CryptoServer

Administrator Guide for CryptoServer Se/CSe in FIPS Mode

http://hsm.utimaco.com

Imprint

Copyright 2017	Utimaco IS GmbH, Germany Germanusstr. 4 52080 Aachen, Germany
Phone	+49 (0)241 / 1696-200
Fax	+49 (0)241 / 1696-199
Internet	http://hsm.utimaco.com
e-mail	hsm@utimaco.com mailto:%20hsm@utimaco.com
Document Number	2011-0002
Document Version	2.2.2
Date	January 11th, 2017
Status	Release
Author	Rainer Herbertz Sven Kaltschmidt Gabriele Wedel Gesa Ott

Table of Contents

1 Introduction 7
2 Fundamentals 8
2.1 Hardware 8
2.1.1 IRAM 9
2.1.2 Key-RAM 9
2.2 Software 10
2.2.1 Boot Loader 10
2.2.2 Firmware Modules 10
2.2.3 Signed Licence File 16
2.2.4 CryptoServer's Boot Process 17
2.2.5 CryptoServer's Operator Modi 21
3 Security Management 23
3.1 Life Cycle and Global States of the CryptoServer 23
3.1.1 State: Initialized 23
3.1.2 State: Defect 24
3.2 User Concept and Access Control 25
3.2.1 Users 25
3.2.2 Authentication 26
3.2.3 Data Types, Configuration Objects and Key Groups 29
3.2.4 Predefined User Roles 30
3.3 Secure Messaging 33
3.4 Auditing 35
3.5 Alarm 37
3.5.1 External Erase 39
3.6 Clear Commands 40
3.7 Behavior of CryptoServer Outside of the Normal Temperature Range 42
3.8 System Keys 43
3.8.1 Default Administrator Key 43
3.8.2 Module Signature Key 45
3.8.3 CryptoServer's Internal Master Key 46
3.9 Backup of Keys and Users 47
3.10 Firmware Module Management 47
3.10.1 Firmware Containers: MTC/MMC, MSC, SYS 47
3.10.2 Package Files 48
4 Administration of CryptoServer with CSADM 49
4.1 CryptoServer Administration Tool CSADM - General 49
4.1.1 System Requirements 49
4.1.2 Installation of CSADM 49
4.1.3 Syntax of CSADM 50
4.1.4 Key Specifiers 52

4.1.5 Password Entry 54
4.1.6 Command Execution with the CSADM Tool 55
4.1.7 Availability of Commands in FIPS Mode 57
4.2 Basic Commands 60
4.2.1 Help 60
4.2.2 PrintError 60
4.2.3 Version 61
4.3 Commands to Set-Up Parameters 62
4.4 Commands to Prepare Firmware Modules 64
4.4.1 MakeMTC 65
4.4.2 RemoveMTC 66
4.4.3 VerifyMTC 66
4.4.4 ModuleInfo 67
4.4.5 RenameToVersion 69
4.4.6 Pack 69
4.4.7 Unpack 70
4.4.8 ListPkg 71
4.4.9 VerifyPkg 72
4.5 CryptoServer Driver Commands 73
4.5.1 Reset 74
4.5.2 ResetToBL 75
4.5.3 Restart 76
4.5.4 GetInfo 77
4.5.5 SetTimeout 77
4.6 Commands for Administration 78
4.6.1 GetState 79
4.6.2 GetBattState 82
4.6.3 StartOS 83
4.6.4 RecoverOS 84
4.6.5 ListFiles 85
4.6.6 LoadFile 87
4.6.7 DeleteFile 89
4.6.8 GetTime 91
4.6.9 SetTime 92
4.6.10 ListModulesActive 93
4.6.11 GetBootLog 95
4.6.12 GetAuditLog 96
4.6.13 ClearAuditLog 96
4.6.14 MemInfo 97
4.6.15 Test 98
4.6.16 LoadPkg 99
4.6.17 CheckPkg 102

4.6.18 Clear ... 103
4.6.19 ResetAlarm ... 105
4.6.20 GetAuditConfig ... 106
4.7 User Management ... 107
4.7.1 ListUser ... 109
4.7.2 AddUserRSASign ... 110
4.7.3 ChangeUserRSASign ... 111
4.7.4 AddUserHMACPwd ... 112
4.7.5 ChangeUserHMACPwd ... 114
4.7.6 DeleteUser ... 115
4.7.7 BackupUser ... 115
4.7.8 RestoreUser ... 116
4.7.9 SetMaxAuthFails ... 116
4.7.10 GetMaxAuthFails ... 117
4.8 User Key Management ... 118
4.8.1 GenKey ... 118
4.8.2 SaveKey ... 120
4.8.3 BackupKey ... 121
4.8.4 CopyBackupCard ... 122
4.8.5 GetCardInfo ... 122
4.8.6 ChangePassword ... 123
4.8.7 ChangePIN ... 124
4.9 Management of Master Backup Keys ... 125
4.9.1 MBKListKeys ... 126
4.9.2 MBKGenerateKey ... 127
4.9.3 MBKImportKey ... 128
4.9.4 MBKCopyKey ... 129
4.9.5 MBKCardInfo ... 130
4.9.6 MBKCardCopy ... 131
4.9.7 MBKPINChange ... 131
4.10 Command Authentication ... 132
4.10.1 LogonSign ... 133
4.10.2 LogonPass ... 134
4.10.3 AuthRSASign ... 135
4.10.4 AuthHMACPwd ... 136
4.10.5 ShowAuthState ... 137
4.11 Secure Messaging ... 138
4.11.1 SessionDH ... 139
4.12 Administration of the CryptoServer LAN ... 140
4.12.1 CSLGetConnections ... 140
4.12.2 CSLScanDevices ... 141
4.12.3 CSLGetVersion ... 143

4.12.4 CSLGetLogFile 143
4.12.5 CSLGetConfigFile 144
4.12.6 CSLPutConfigFile 145
4.12.7 CSLSetTracelevel 146
4.12.8 CSLShutdown 147
4.12.9 CSLReboot 147
4.12.10 CSLGetTime 148
4.12.11 CSLSetTime 148
4.12.12 CSLGetSerial 149
4.12.13 CSLGetLoad 149
4.12.14 CSLListPPApps (ListPINPadApps) 150
4.13 Miscellaneous Commands 151
4.13.1 Cmd 151
4.13.2 CmdFile 152
4.13.3 CSTerm 153
4.13.4 Sleep 153
5 Configuration of CryptoServer via 'p11tool2' 154
6 Batteries of the CryptoServer 158
6.1 Check State of the Batteries 159
7 Typical Administration Tasks 160
7.1 How to Install the CryptoServer 160
7.2 Generate User's Authentication Token 161
7.3 How to Enter FIPS-Mode: Set-Up and First Personalization of a New CryptoServer . 163
7.4 How to Generate a Master Backup Key 166
7.5 How to Get State Indicators 167
7.6 How to Quit an Error State 171
7.7 How to Clear the CryptoServer 172
8 Troubleshooting 174
8.1 Check Operativeness and State of CryptoServer 174
8.2 Alarm Treatment 177
9 Built-in Elliptic Curves 178
10 Appendix: FIPS Validated Firmware Package 179
11 References 181

1 Introduction

This document gives comprehensive guidelines on the administration of Utimaco's hardware security module CryptoServer (**CryptoServer Se** version 3.0.2.0 or **CryptoServer CSe** version v4.0.3.0**)** if run in FIPS-mode with the administration tools CSADM and p11tool2. Here, FIPS mode means FIPS approved mode of operation according to [FIPS140-2].

It should be read carefully by all persons who are allowed to assume the role of an *Administrator* or *Security Officer* (*Cryptographic Officer* in [FIPS140-2]) for the CryptoServer.

The cryptographic services that are offered by the CryptoServer are not described in this document: detailed command descriptions e. g. for encryption or decrypting services, hashing, key generation and key management can not be found in this Administrator's Guide. Moreover, cryptographic services can not be performed by any CryptoServer's Administrator or Security Officer but only by persons who are allowed to assume the Key Manager or (Cryptographic) User role.
A Guide for Cryptographic Users and Key Managers of the CryptoServer is provided by document [CSFIPS-UserGuide].

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

Main task of the **CryptoServer's Security Officers** is the 'local' (slot or Key Group specific) configuration and user management.

CryptoServer can be administrated with the help of the CryptoServer Administration tool CSADM or CryptoServer PKCS#11 Administration Tool p11tool2 (both command line utilities provided by Utimaco).

Instructions will be given for certain typical administrative situations, see section 7 *Typical Administration Tasks*. The detailed command descriptions of CSADM in chapter 4 should be used for the execution of commands in practice. Chapters 2 and 3 give the necessary background information for e.g. the security mechanisms of the CryptoServer.

A detailed command description of the PKCS#11 Administration Tool 'p11tool2' can be found in chapter 5.

How to check the state of the CryptoServer's batteries will be explained in chapter 6. This is vital because in case that the carrier battery is not replaced early enough, all data inside the CryptoServer will be deleted.

Chapter 8 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.

2 Fundamentals

The CryptoServer is an encapsulated, protected security module which is realized as multi-chip embedded cryptographic module in the sense of FIPS 140-2 [FIPS140-2]. Both CryptoServer Se and CryptoServer CSe meet FIPS 140-2 overall level 3 requirements; CryptoServer CSe additionally meets FIPS 140-2 level 4 requirements in section "Physical Security". The primary purpose for this module is to provide secure cryptographic services like encryption or decryption (for various cryptographic algorithms like Triple-DES, RSA and AES), hashing, signing and verification of data (RSA, DSA[1], ECDSA), random number generation, on-board secure key generation, key storage and further key management functions in a tamper-protected environment.

In FIPS mode, the module offers a general purpose API with FIPS Approved algorithms for the above mentioned cryptographic services, as well as an administrative interface. A Secure Messaging concept protects the communication to and from the module by message encryption and MAC authentication.In this chapter the fundamentals of the hardware security module CryptoServer (and its environment) will be described:

2.1 Hardware

Utimaco's hardware security modules CryptoServer are physically protected specialized computer units designed to perform sensitive cryptographic tasks and to securely manage cryptographic keys.

CryptoServer Se is designed for most typical security requirements in commercial environments. All security-relevant hardware components are encapsulated and completely covered by potting material (epoxy resin). This hard, opaque enclosure protects the sensitive CryptoServer hardware components from physical attacks on sensitive keys and data. This mechanism meets the requirements of FIPS 140-2 level 3 in section "Physical Security".
CryptoServer Se may be delivered with a hardware accelerator chip which provides highest performance for RSA operations (models CryptoServer Se1000/Se400).

CryptoServer CSe is designed for highest requirements regarding physical security, as e.g. required in banking and governmental environments. Its extensive sensory mechanism, which includes the usage of a sensory-protection foil, reacts on all kind of mechanical, chemical and physical attacks and erases sensitive keys and data actively and within shortest time from CryptoServer's internal memory. This mechanism meets the requirements of FIPS 140-2 highest level 4 in section "Physical Security", and is certified according to this security standard.

Either CryptoServer series come with the same software architecture, in this document they will henceforward only be named *CryptoServer*.

All models of the CryptoServer Se and CSe series are available in two variants:

- CryptoServer as an expansion card with a PCI Express bus. This is referred to below as the **CryptoServer PCIe**.

[1] CSe only

- CryptoServer LAN as a network component (appliance), which can be easily integrated into a network. This is referred to below as **CryptoServer LAN.**

On the CryptoServer expansion card there are two memory areas which are highly relevant for CryptoServer's security architecture, because they can be used to store sensitive data in clear text:

2.1.1 IRAM

CryptoServer's CPU is equipped with internal RAM (IRAM) for code, data and cache.

The *IRAM* can be used to hold sensitive data in plain on runtime. It is guaranteed that the IRAM is erased actively within a very short time in case of an alarm triggered by the sensory (each memory cell of the IRAM will be overwritten), see subsection 3.5. If the CryptoServer goes down on power-off, the IRAM will be erased, too.

2.1.2 Key-RAM

The *Key-RAM* is a non-volatile RAM which is protected by the sensory: In case that an *alarm* is registered the Key-RAM will be actively erased within less than two milliseconds. Within this timeframe the Key-RAM will be overwritten five times, alternately with 00_h and FF_h patterns.

Since the sensory-protection holds on runtime and in retention (i.e. if power supply is switched off) the *Key-RAM* is used to store the clear CryptoServer's internal *Master Key* (which is needed to encrypt data inside of the CryptoServer, see 3.8.3). Independently from the mode of operation the mechanism to detect an attack is active: An internal power supply will in any case provide enough energy to erase the *Key-RAM* and therefore the CryptoServer *Master Key* in case of alarm.

2.2 Software

During the boot process and the life cycle of a CryptoServer several parts of its software come into action. This chapter only lists these different software components and gives a rough description of their functionality.

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

- Boot loader,
- Operating system (SMOS) with firmware modules.

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

2.2.1 Boot Loader

The *boot loader* is an independent software running before the real operating system and the firmware modules are started. The boot loader is the first software started inside the CryptoServer after a reboot.

If during the boot process the boot loader finds itself *initialized* (i. e. the CryptoServer's public Production Key has been found) and the operating system module SMOS is present in the CryptoServer, the latter will be started by the boot loader. A more detailed description of the boot process follows in section 2.2.4.

2.2.2 Firmware Modules

This chapter describes the firmware that is normally running – SMOS and the firmware modules. Only a subset of these firmware modules is necessary to operate a CryptoServer in FIPS-mode; see chapter 10 for a comprehensive list of all firmware modules belonging to the FIPS firmware package and their necessary approved versions.

Software module or firmware module in the context of this documentation denotes an encapsulated software part running inside 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 following illustration gives a general overview of the various software parts (including the software running on the host PC), how they are divided into modules and – in a rough way – where the communication paths pass.

CryptoServer Se - Software Architecture

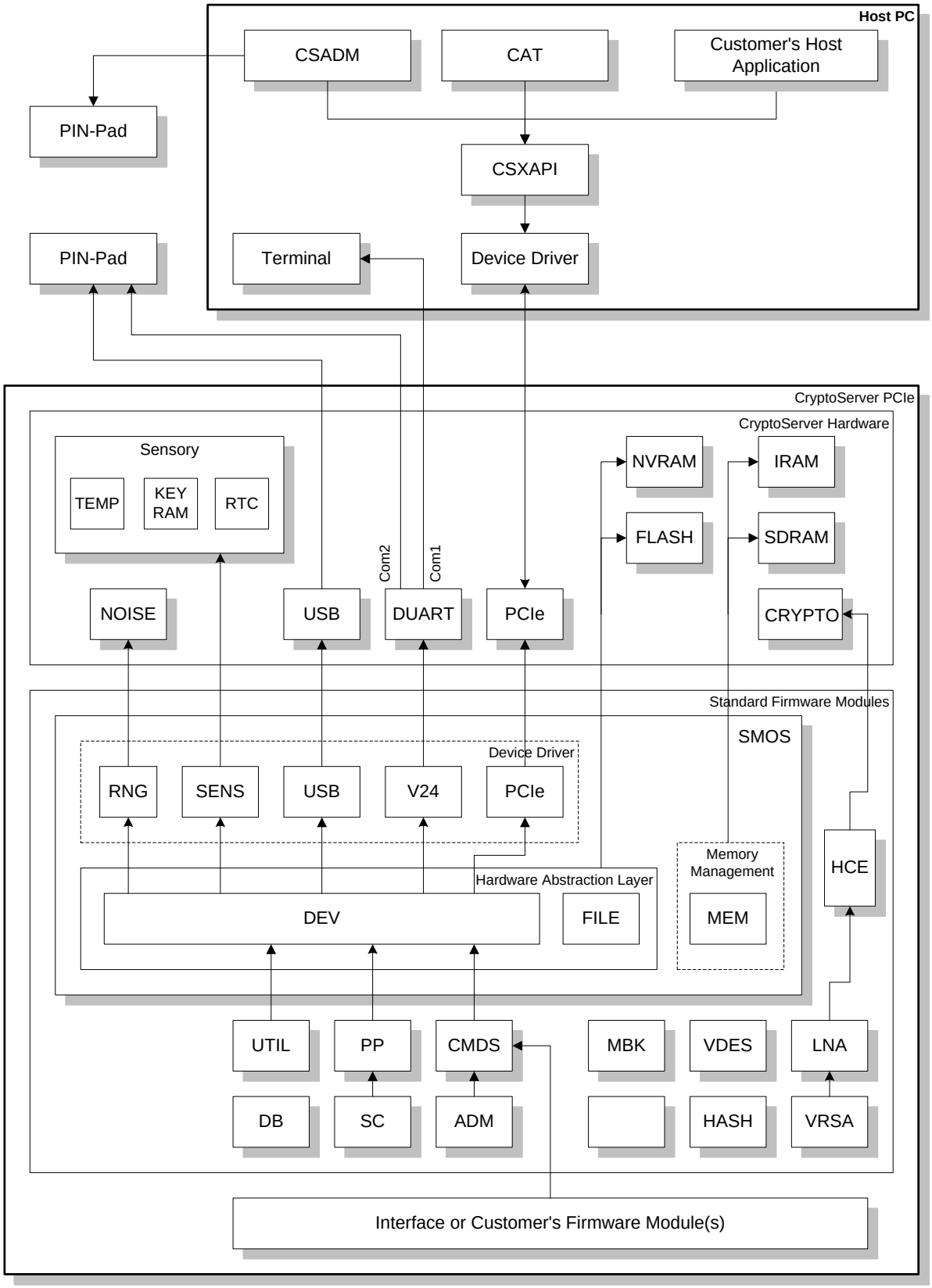


Illustration 2-2: Overview of the CryptoServer Se software architecture

CryptoServer CSe - Software Architecture

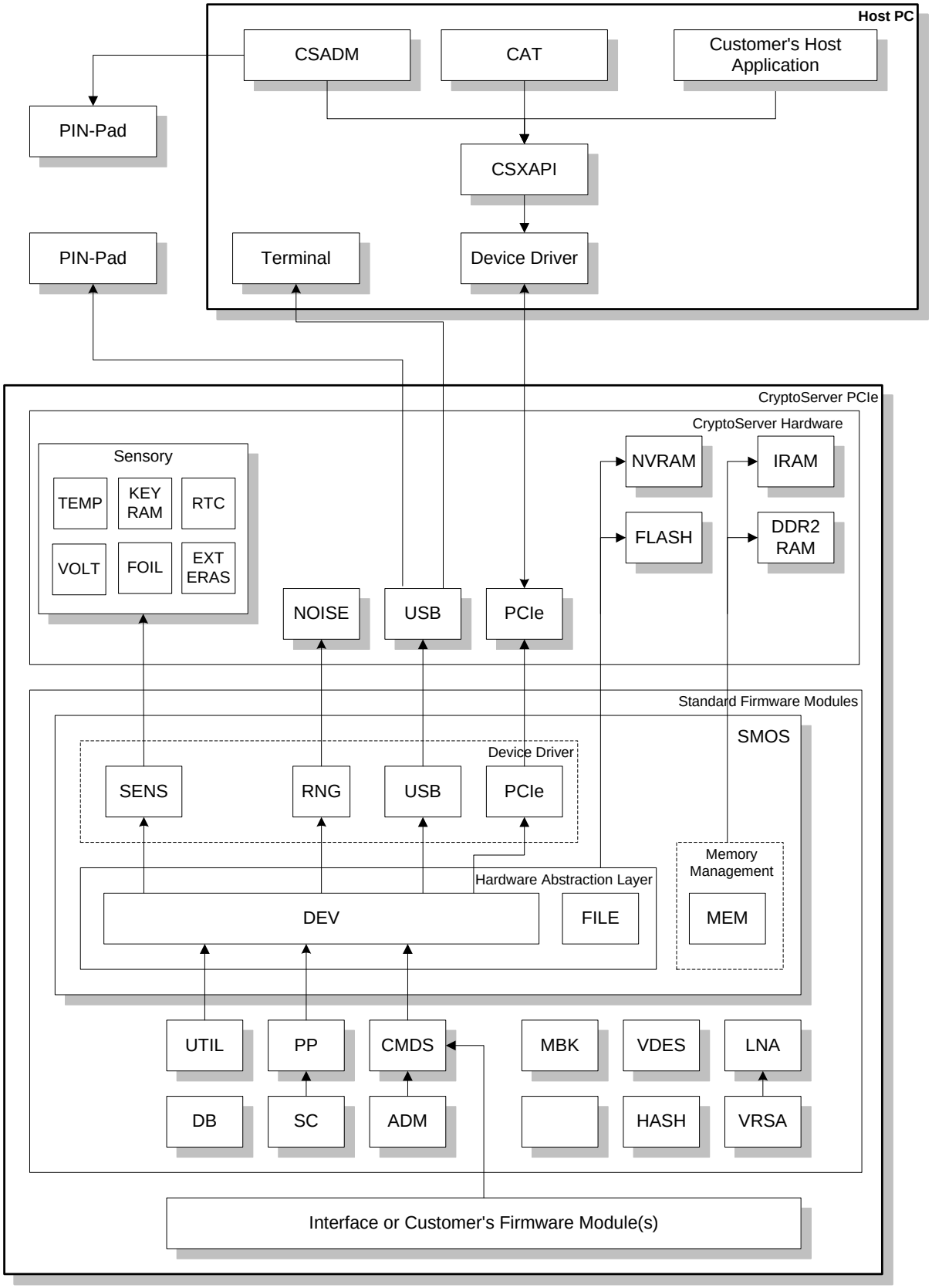


Illustration 2-3: Overview of the CryptoServer CSe software architecture

The above illustration gives an overview of the software modular parts of a CryptoServer system, including the software on the host PC like CryptoServer's Administration Tool CSADM, the PCI Express driver and CryptoServer's Application Programming Interface CSXAPI.

The modular software parts inside the CryptoServer are called *firmware modules*. The firmware modules can be divided into several classes:

- operating system (SMOS)
- further standard firmware modules (CMDS, UTIL, ADM, VDES, ...) provided by Utimaco (which are realized in order to get a running CryptoServer with basic functionality e. g. for administration, cryptographic routines and data management) and
- other application firmware modules (cryptographic interfaces like PKCS#11 or CXI provided by Utimaco, or as developed by customer).

Customer firmware modules may use all functionality that is CryptoServer-internally provided by Utimaco's standard firmware modules.

In FIPS mode customer firmware modules are not allowed.

The role and functionality of these standard firmware modules will be explained in the following subsections.

2.2.2.1 Operating System – SMOS

The CryptoServer's operating system SMOS (**S**mall **M**ultitasking **O**perating **S**ystem) provides mechanisms for task handling, inter process communication, memory management and file handling.

Furthermore SMOS realizes access to CryptoServer's internal hardware components like memory areas, RTC, physical interfaces etc. and offers appropriate access command interfaces to the other firmware modules. In particular based on the included device drivers SMOS provides through its *hardware abstraction layer* a high level access to the physical interfaces of the CryptoServer device (PCI express and USB; Se only: and V.24), for read and write operations on the devices.

2.2.2.2 Further Standard Firmware Modules

The following firmware modules are necessary to operate a CryptoServer in FIPS-mode. Their respective functionality is described in the list below. If one of these firmware modules is not loaded or can not be started, the CryptoServer is either not in FIPS-mode or it will enter FIPS error state. See chapter 10 for a comprehensive list of all necessary firmware modules and their necessary approved versions.

With the exception of the command scheduler module CMDS, the administration module ADM, the cryptographic interface module CXI and the Master Backup Key management module MBK these modules have no external interface and can only be accessed by other firmware modules via an internal public C interface.

- **SMOS** (Small Multitasking Operating System): operating system
- **CMDS** (Command Scheduler): command processing incl. protocol stack, user management
- **ADM** (Administration): general administration of CryptoServer, incl. firmware module management and management of audit logfile

- **UTIL** (Utilities): internal access to random number generator and real time clock
- **VDES**: DES algorithm (key generation, encryption/decryption, MAC)
- **AES**: AES algorithm (key generation, encryption/decryption, MAC)
- **VRSA** and **LNA** (Long Number Arithmetic): RSA algorithm (incl. key pair generation)
- **ECDSA** and **ECA**: elliptic curve algorithm (incl. ECDSA signature generation/verification, key pair generation)
- **HASH**: various hash algorithms
- **ASN1**: ASN.1 decoding and encoding
- **DB**: database management (e.g. for secure storage of keys and other sensitive data)
- **MBK**: management of the Master Backup Key (MBK), incl. generation, export and import of MBK
- **HCE** (Hardware Crypto Engine; Se only:): driver for the RSA crypto accelerator chip[2]
- **DSA** (CSe only)**:** DSA algorithm (incl. DSA domain parameter generation and verification, signature generation/verification, key pair generation)
- **CXI**: Utimaco's proprietary cryptographic HSM interface which is used by the host libraries for the CXI API, CSP/CNG, JCE, OpenSSL
- **FIPS140:** The FIPS Mode Control Module is responsible to check if all firmware modules that are necessary for FIPS-mode are present, have correctly been started and that none of the power-up self-tests has failed.

CryptoServer is usually delivered with a standard set of firmware modules loaded (firmware package of the SecurityServer which is not identical with the FIPS firmware package), and stored as *.msc files.

Additionally, during production at the manufacturer's site, a subset of these firmware modules will be loaded a second time into the CryptoServer as so-called *system firmware modules*.[3] The purpose of these system firmware modules is to provide a backup-copy of every system-relevant firmware module.

*CryptoServer generally stores a firmware module in form of a **MSC (module storage container)**. The MSC format is for CryptoServer-internal use only: It stores the module code together with certain module information and the check value for the module code's integrity (SHA-512 hash value over the module code which will be verified at every start-up).*

*Firmware modules that are downloaded during an early phase of production, at the manufacturer's site, will be stored as **SYS** files (**CryptoServer system files**), which is exactly the same format as a MSC, just with another extension *.sys. These **system firmware modules cannot be deleted** and thereby offer anytime a fallback possibility for firmware.*

[2] The HCE module will only start-up if the hardware crypto accelerator chip is built into the CryptoServer Se. If this is not the case, the initialization state of HCE will be set to INACTIVE and no services of HCE are available.

[3] For this a specialized *LoadFile* boot loader command will be used which stores the firmware modules as "*.sys" files (system firmware modules).The boot loader command *LoadFile* is not available for any CryptoServer after delivery, i. e. at the customer's site.

The following non-standard firmware module belongs to the FIPS firmware package:

FIPS140	**FIPS Mode Control Module** This module is only active in *Power-Up Initialization and Self Test 2* state during the CryptoServer's power-up phase. It is responsible to check if all firmware modules that are necessary for FIPS mode are present, have correctly been started and that none of the power-up self-tests has failed. FIPS140 offers no external interface.

During the process of setup and personalization of the CryptoServer (in order to get a working CryptoServer in FIPS-mode) all firmware modules belonging to the FIPS Firmware Package have to be loaded. See chapter 7.3 for detailed guidance through the CryptoServer's setup and personalization process.

If one of the firmware modules belonging to the FIPS firmware package is not loaded or can not be started, or if any further module is loaded the CryptoServer is either not in FIPS-mode or it will enter FIPS error state.

The modules that will be downloaded as system firmware modules are those that are needed to do all necessary administration tasks to set-up the CryptoServer (e. g. download, replacement and deletion of further firmware modules, administration functions). The following modules are **system firmware modules**:

- **SMOS, UTIL, CMDS, ADM, VDES, AES, VRSA, DSA[4], LNA, ECDSA, ECA, HASH, DB**

These modules are always loaded inside the CryptoServer and can be used as back-up system of firmware modules for emergency cases. *Even if the OS or other parts of the firmware is later on deleted or replaced by future versions, or if the CryptoServer is cleared by the customer, all system firmware modules ('*.sys' files) will be preserved.* Therefore, e. g. in case of buggy or incompatible new firmware modules, a fallback to this first set of system firmware modules can always be made in order to reconfigure the whole system.

The functional design of each standard firmware module and its interface is specified in more detail in the respective module specific documentation.

[4] CSe only

2.2.3 Signed Licence File

The CryptoServer uses a Signed Licence File concept to regulate the availablity of certain features and performances (for instance the speed of certain cryptographic algorithms) in a customer-specific way.
These regulations do in no case produce security-relevant changes to the software!

A Signed Licence File (SLF) is a customer-specific licence file "*.slf" which is generated by Utimaco and signed by Utimaco's *Module Signature Key* (RSA signature, see 3.8.2).

The SLF can be loaded with the *LoadFile* command. During download the signature will be verified, and a check value for the SLF's integrity (SHA-512 hash value over the SLF) will be calculated. The SLF will be stored together with this check value which will replace the signature. With every start-up the operating system SMOS will, during its boot process, search for the SLF and verify its hash value.

If a SLF is missing or its integrity cannot be verified, the CryptoServer's firmware modules always use their default values concerning the execution of certain functions. These are in any case the most restrictive options, i. e. without SLF the CryptoServer will for instance perform certain algorithms only with lowest speed.

2.2.4 CryptoServer's Boot Process

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 7.7).

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

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

2.2.4.1 Boot Phase Controlled by Boot Loader

After any reset (power-up or hardware reset) the CryptoServer starts with the program code that is stored in the *BL code area,* which is located in the flash device.[5] This is the boot loader firmware code. The boot loader will do the first necessary start procedures including self-tests and offers some basic commands like status queries:

At the end of the boot loader-controlled boot phase, after the initialization of the PCIe interface, in personalization mode the bootloader provides an open time window for command.

If the boot loader will not receive any command within a defined time window (3 seconds), and if the operating system module SMOS is loaded and its integrity can be verified (SHA-512 hash value), the boot loader will start SMOS automatically. If this was successful the CryptoServer is considered to be in *operational mode* (or maintenance mode) and the boot loader terminates itself.

2.2.4.2 Start OS

Then CryptroServer's operating system (firmware module SMOS) can be started in two ways – either in the regular way or by starting the failsafe back-up copy of SMOS:

At the end of the boot loader-controlled phase of the boot procedure the boot loader first always tries to search for and start the OS as 'smos.msc'. Only if this fails it tries to start the 'smos.sys' system module.

Both modalities to start the OS can also be explicitly chosen by the user if the respective external CryptoServer command is performed:

[5] On a CryptoServer CSe, the CryptoServer starts with the First Stage BL which is stored in the FPGA and which itself starts the (second stage) Boot Code as stored in the BL code area.

- The command *StartOS* (see 4.6.3) corresponds to the usual way of starting SMOS (i. e. starting ‘smos.msc’) whereas
- the command *RecoverOS* (see 4.6.4) corresponds to the start of the back-up copy of SMOS (i. e. start of the system module ‘smos.sys’).

If the OS start fails in both modalities the CryptoServer will remain in *bootloader mode*. The boot loader remains active and further boot loader commands (like e. g. status queries) can be performed.

If the start of the SMOS is successful the CryptoServer enters *operational mode* or *maintenance mode* (see below) and the boot loader terminates.

2.2.4.3 Boot Phase Controlled by Operating System SMOS

After the boot loader has passed the control to the operating system, SMOS will roughly perform the following steps:

- initialization of the memory
- initialization of the flash file system
- initialization of all hardware peripherals (including serial (Se only), PCIe and USB interfaces)
- searching for firmware modules in the flash file system (depending on its own starting mode: If SMOS itself has been started as ‘smos.msc’, and if there is no alarm present, it will start all ‘*.msc’ firmware modules. If SMOS is started as ‘smos.sys’ module, or if an alarm is present, it will start all ‘*.sys’ system firmware modules).
- verifying the integrity of all firmware modules (see 3.10.1)
- starting all firmware modules.

2.2.4.4 Watching the Boot Process and Analyzing Errors

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

If no fatal error occurs during the boot phase (i. e. if all basic firmware modules can be started successfully) the log messages can be retrieved later using the administration command *GetBootLog* (see 4.6.11).

2.2.4.5 Start Back-up Copies of Firmware

If the CryptoServer hangs up itself during the boot phase, it cannot be accessed anymore over the command interface. This may happen for example if the administrator has deleted one of the base firmware modules or downloaded buggy or incompatible software.

In such a situation of hang-up the back-up copies of the base firmware modules (i. e. all system firmware modules ‘*.sys’) can be started to put the CryptoServer in operational mode again. This has to be done by issuing the command ResetToBL followed by the command RecoverOS (see 4.5.2 and 4.6.4).

After that only the system firmware modules are running which is sufficient to administrate the CryptoServer. The bad / missing firmware can now be replaced / loaded using the normal administration command (*LoadFile* command, see 4.6.6). After that the CryptoServer can be put again in normal operational mode (running all firmware modules '*.msc') with a *Restart* command.

2.2.4.6 Boot Process in FIPS Mode

If the CryptoServer is in FIPS-mode, the boot process is similar to that described above. The most important differences are the following:

1. During the boot loader phase of the boot process, there is no time window for command: Various tests and initialization steps will be performed and, in error free case, at the end of this process the operating system SMOS will be started automatically. Thus no boot loader command can be performed as long as no error has occurred.
2. If during this phase a *FIPS error* is found (if for instance the boot loader configuration file can't be read, or if the Known Answer Test for the boot loader's SHA-512 algorithm fails), the CryptoServer enters ***Boot Loader Error* state**: The boot loader remains active and a time window for command is opened, but only commands for status requests and loggin information can be performed. See 4.6.
3. When SMOS has started successfully, it performes further tests and initialization steps. In particular it starts and initializes all firmware modules that are found in the flash file. If this was successful, the CryptoServer is considered to be in (FIPS mode and) **normal operational state**, i. e. not in any FIPS error state.
4. If during this phase any *FIPS error* is found (failure of one of the power-up self tests), the CryptoServer enters ***OS Error* state**. Examples for FIPS errors include: any cryptographic algorithm test has failed (e.g. DES or RSA Known Answer Test), 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 some non-security relevant commands (e.g. status requests) can be performed. See 4.6.
5. The possibilities to watch the boot process and analyze errors in FIPS mode are described in 4.6.11.
6. The possibilities to recover the back-up copies of the base firmware (see 2.2.4.5 above) are not available in FIPS mode but only in personalization mode.

2.2.4.7 Different Commands to Reset the CryptoServer

There are three different driver commands that reset the CryptoServer and start the boot process:

- The *Reset* command (see 4.5.1) causes a hardware reset. The boot process described in the previous sections is performed. For a CryptoServer in personalization mode, this includes the 3 second time window for command of the boot loader; if the CryptoServer is in FIPS mode, this 3 second time window is skipped.
- The *Restart* command (see 4.5.3) causes a hardware reset, waits for the beginning of the time window for boot loader commands and immediately issues a *StartOS* command (which will be ignored in FIPS mode). Like this the boot process is sped up by skipping the 3 second time window of the boot loader. Additionally, the *Restart*

command waits for all firmware modules to be started by the operating system SMOS. After the *Restart* command has successfully finished, the CryptoServer is in operational mode and ready to accept commands.

- The *ResetToBL* command (see 4.5.2) causes a hardware reset, waits for the beginning of the time window of the boot loader and issues a *GetState* command. The boot process is stopped and the CryptoServer remains in boot loader mode (see 2.2.5). After the *ResetToBL* command has successfully finished, the CryptoServer is immediately ready to accept boot loader commands.

For normal purposes it is recommended to use the Restart command. After a Restart the CryptoServer is in operational mode.
Only if boot loader commands should be performed (with CryptoServer being in boot loader mode), use the ResetToBL command.
It is strongly recommended not to use the Reset command.

If the CryptoServer is in FIPS mode, the ResetToBL command is not available. In FIPS mode, the Reset command and Restart command can be used and show the same result (putting the CryptoServer into operational mode without time delay).

If the CryptoServer is in personalization mode, the commands for resetting the CryptoServer should be used according to the following rule:
If boot loader commands shall be performed, use the ResetToBL command.
If operational commands shall be performed and/or the OS shall be started, use the Restart command.

2.2.5 CryptoServer's Operator Modi

2.2.5.1 Boot Loader Mode, Operational Mode, Maintenance Mode

Depending on the loaded firmware modules, a CryptoServer can either be in FIPS-mode (if the respective FIPS validated 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.

Basically the CryptoServer can be (if the respective software is loaded) in four different operator modi, depending on which firmware is actually active:

The CryptoServer is in boot loader mode if the boot loader is active, i. e. the boot loader is powered up and running 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 and further firmware modules ('*.msc' modules) could have been started successfully and are active.
The CryptoServer is in maintenance mode if its back-up system firmware modules (SYS modules: '*.sys' files) have been started successfully and are active.
A CryptoServer in FIPS mode will never reach maintenance mode.
The CryptoServer is in 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.

If the CryptoServer is in the *operational* mode or *maintenance* mode, further download, replacement and deletion of firmware modules and other administrative work can be done with help of the respective administrative commands (see 4.6).

- With the *ResetToBL* command the CryptoServer can be set to boot loader mode. This may be useful for diagnostic purposes.
- With the *Restart* command or (if the CryptoServer is in boot loader mode) the *StartOS* command the CryptoServer can be set to *operational mode* (if the respective firmware modules are loaded and not defect).
- With the *ResetToBL* followed by the *RecoverOS* command the CryptoServer can be set to maintenance mode.

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 = Maintenance Mode (not available in FIPS mode) or*
- *mode = Boot Loader Mode*

(as well as the CryptoServer's state) is part of the answer to the command

3 Security Management

In this chapter the various CryptoServer's security mechanisms, like e.g. user authentication, secure messaging, or alarm treatment will be described.

3.1 Life Cycle and Global States of the CryptoServer

In error-free operation throughout its life cycle the CryptoServer runs through the following states:

BLANK -> MANUFACTURED -> INITIALIZED

Additionally the CryptoServer can be in **DEFECT** state.

For a CryptoServer in *blank* state there is no housing present yet and the CryptoServer board is not yet covered with potting material (Se) or wrapped and tamper protected (CSe), no software is loaded. If the CryptoServer is in *manufactured* state the boot loader code is loaded, the mechanical housing of the CryptoServer is closed. For a CSe the tamper foil is wrapped around the housing, the CryptoServer is potted and mounted on the PCIe carrier card, and the sensory mechanism the CryptoServer is activated.

At the customer's site, only CryptoServer devices in state initialized or defect will normally appear. The states blank and manufactured are exclusively relevant for the production process and for maintenance work done by the manufacturer. Utimaco will never deliver the CryptoServer in one of these states.

If the CryptoServer is found to be in defect state, or in any other state but initialized, the manufacturer/Utimaco IS GmbH has to be contacted.

Information about CryptoServer's state can be retrieved via the *GetState* command.

Both the *Initialized* and the *defect state* can occur both in *FIPS mode* and in *personalization mode.* If a CryptoServer in *FIPS mode* is in the *defect state*, this implies that it is in *FIPS error state (BL error state).*

3.1.1 State: Initialized

Any CryptoServer leaving the manufacturer's secure environment and being delivered to the customer is in *initialized* state.

As a minimum, in initialized state the boot loader program code and all standard firmware modules are loaded as '*.sys'-files, as well as all (manufacturer-owned) system keys (public parts) like *Module Signature Key*, *Default Administrator Key* and the *Production Key*.

Any CryptoServer being at the customer's site should be in state initialized.

3.1.2 State: Defect

If the boot loader's hardware self test fails during the starting phase, the CryptoServer will be in the *defect* state. This test is always run at the beginning of the boot process.

In defect state the CryptoServer exclusively accepts the commands *GetState* and *GetBootLog* (if yet technically possible), which are not to be authenticated. The alarm mechanism of the CryptoServer remains unchanged.

If the CryptoServer is in the defect state, please contact the manufacturer/Utimaco.

3.2 User Concept and Access Control

Security-relevant external commands that are sent to the CryptoServer may only be performed if the sender has authenticated *the command. Such command authentication can only be done by so-called* ***users*** *which have to be registered at the CryptoServer device.*

To control and restrict the access to security-relevant commands, the CryptoServer implements the concept of **users**: Authentication of commands can only be done by registered *users* who are equipped with the relevant *permissions*. Only those users are allowed to access security-relevant commands.

3.2.1 Users

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

- *Name* (which serves as a unique identifier for the user), 8 bytes long.
- *Long Name* (optional, which serves as a unique identifier for the user), up to 255 bytes long.
- *Permissions* of the user. The structure of these user permissions corresponds to the structure of the authentication state (as explained in the section 3.2.2.1 below), i. e. it consists of eight values in the range from 0-3.
- *Flags* that determine if static login or secure messaging is 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 3.2.2.4 below).
- *Authentication data* like cryptographic keys or passwords, see below.
- *User Attributes* (optional).

For a more detailed description of the possible user data see 4.7.

The CryptoServer provides also the appropriate commands to create or delete a user or to change his/her authentication token/data, see 4.7.

For every authentication to be performed, the user name and the authentication mode (static login or single command authentication, see 3.2.2.3 below) have to be specified. Further necessary data depend from the authentication mechanism of the user.

After a successful authentication the CryptoServer's *authentication state* will be augmented by the permissions of the user. The authentication concept will be explained in the following chapter.

3.2.2 Authentication

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

Inside a CryptoServer, the firmware module CMDS is responsible for managing the authentication of commands:

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 CMDS (including verification of the authentication), and, depending on the result, CMDS will increment the authentication state. The firmware module which was addressed by the command will later check this authentication state.

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

3.2.2.1 Authentication State and User Permissions

The CMDS firmware module internally stores an **authentication state**. 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:

The authentication state consists of eight values, each in the range from 0 to 3. These eight values represent eight different **user groups** (user group 0 to 7). Each value, called **authentication level**, gives the height (sum) of the rights/permissions gained through the authentications of various users from this specific user group. Each authentication level can vary from 0 (no authentication) to 3 (highest level of permission).

Example:

	Eight values of example authentication state							
Value of Authentication state:	2	0	0	1	0	3	0	1
means authentication level for user group ...	7	6	5	4	3	2	1	0

In this example, the CryptoServer has reached authentication level 2 in user group 7, level 0 in user group 6, (...), and authentication level 1 in user group 0.

After a user has successfully authenticated towards the CryptoServer the authentication state will be augmented by the ***permissions* of the user** (see 3.2.1):

Example:

Authentication state before user's login:	01000000
Permissions of the user:	01000001
Authentication state after user's login:	02000001

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). 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.

3.2.2.2 User Groups and Access to Commands

Therefore, depending on the application, the eight different user groups/permissions can be used to control the access to sensitive commands.

Using these access control mechanisms, and if it is wanted for security reasons, it is for example possible to determine that *a specific command is only available after an authentication following the **2-persons-rule***: If for instance the necessary authentication state for this command demands the authentication level 2 in some user group 'x', and if for this group 'x' only users with permission level 1 are registered, then the specific command must be authenticated by two users of this group 'x'. Thus the command is only accessible if the 2-persons-rule is obeyed.

So the implementation of security rules for the authentication of commands within a specific Crypto¬Server application is dependent upon the user management:

With the concrete implementation of application firmware modules the framework for the security rules for command authentication will be determined. In particular it will be fixed which commands have to be authenticated and which not.

Within this framework, with setting rules for user administration, the customer-individual security rules for command authentication can be realized. This can be done by setting the specific permissions and authentication mechanisms for any user when the user is created.

3.2.2.3 Authentication Mode: Single Command Authentication

For the authentication, in FIPS mode only**Single Command Authentication** is possible. This means that the authentication holds only for a single command and must therefore be done together with this command. After the execution of the command the authentication state will be automatically set back to the previous value.

Several users can authenticate/login one after the other or within one command. Doing this, mixed authentication mechanisms are allowed.

3.2.2.4 Authentication Mechanisms

In FIPS mode the following different authentication mechanisms are possible:

HMAC Password Authentication:

For this mechanism a password of arbitrary length will be used.
First the host demands an 8 bytes random value from the CryptoServer. Then the host calculates the HMAC value over this random value and the command data block using the password as the HMAC key (hash algorithm for the HMAC calculation is SHA-256). It transfers this hash value to the CryptoServer which recalculates and checks the hash with the help of the password stored in the user database.
For the following reasons this mechanism is also qualified for the communication via Ethernet (CryptoServer LAN):

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

RSA Signature Authentication:

For this mechanism first the host demands an 8 bytes random value from the CryptoServer. Then the host (or RSA smartcard) calculates a RSA signature over this random value and the command data block with the user's private RSA key (PKCS#1 signature format). 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. The hash algorithm for the signature calculation is SHA-1.

This mechanism is particularly qualified for communication via Ethernet (CryptoServer LAN) and therefore recommended for secure applications.

For the syntax which has to be used in this context with the CSADM tool, see section 4.10.

3.2.2.5 Consecutive Failed Authentication Attempts

For every user the number of consecutive failed authentication attempts is internally counted and stored as a user attribute (counter for failed authentication attempts, *AuthenticationFailureCounter*). This counter can be retrieved with command *ListUser* (attribute 'Z', see 4.7.1). If the authentication

of the user fails the user's *AuthenticationFailureCounter* is incremented by one, if the authentication of the user is successful the user's *AuthenticationFailureCounter* is reset to zero.

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

If a user is locked no further authentication is possible for this user, consequently this user cannot perform any command which has to be authenticated. Only a user with permission for user management (level 2 in user group 7) is allowed to unlock another user by setting the users *AuthenticationFailureCounter* back to zero (see *Change User* commands in section 4.7.3 and 4.7.5).

The value of *MaxAuthFailures* can be set and retrieved with external commands *Set Maximum Authentication Failures* and *Get Maximum Authentication Failures (sections* 4.7.9 and 4.7.10).

3.2.3 Data Types, Configuration Objects and Key Groups

The generic data type to be used in the cryptographic CXI command interface is called ***CXI Object***: A CXI Object denotes a *Key*, a *Configuration Object* or a *Storage Object*:

- A ***Key*** is a cryptographic key, key pair or generic secret to be used with a specific cryptographic algorithm (DES, AES, RSA, ECDSA/ECDH, DSA/DH/DH_PKCS[6], Generic Secrets to be used for HMAC calculation),
- A ***Storage Object*** may contain a certificate, domain parameters or other data which shall be stored in the CryptoServer's key table. Storage Objects can only be used to store information securely within the CryptoServer; they cannot be used for any evaluation or calculation within the CryptoServer.
- A ***Configuration Object*** contains a list of configuration properties. The following properties exist:

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

[6] CSe only

Property	***Description***
	and has to be greater or equal than the strength of the wrapped key. If *false* the strength of the wrapping key is not checked (default: *false*).
AUTH_KEYM	Permission mask of the *Key Manager* (default: 00000002). May be set to '00000020'.

If the configuration property ALLOW_GROUPS is set to *'yes'*, each operator as well as every CXI Object may be assigned to a ***Key Group*** (called '*Slot'* in PKCS#11):

- A CXI Object is called ***Global*** if assigned to no Key Group; it is called ***Local*** if it is assigned to a Key Group. In general Local CXI Objects can only be accessed by users which are assigned to the same Key Group; Global Objects can be seen by all users.
- A Key or Storage Object is called ***Assigned*** to a user if both belong to the same Key Group, or if the Object is Global.
- Operators may be assigned to multiple Key Groups by using wildcards within the 'Key Group' attribute.
- A Local Configuration Object only applies to the assigned Key Group; the attributes of the Global Configuration Object apply to all Global Objects, and to all Local Objects with no Local configuration value defined.

The property 'ALLOW_GROUPS' only exists in the Global Configuration Object; it does not exist in a Local Configuration Object.

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

3.2.4 Predefined User Roles

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

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

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

   To execute the administration commands as provided by the PKCS#11 administration tool *p11toolv2* for global configuration management and PKCS#11 specific user management (e.g. *SetGlobalConfig* and*InitToken*, see chapter 5*)* the *user permission* '2000000' is sufficient.

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

   These users must have the *user permission* '00000200' (or higher). *Security Officers* are allowed to perform Key Group specific user management and can work on Local Configuration Objects (Configuration Object whose Key Group matches the *Security Officer*'s Key Group), and they can set the TRUSTED attribute of every Assigned wrapping key.

All commands as provided by the PKCS#11 administration tool p11toolv2 for Key Group specific configuration and PKCS#11 typical user management (e.g. *SetSlotConfig* and *InitToken*, see chapter 5) can only be performed after a *Security Officer's* authentication, i. e. when the authentication state '00000200' (or higher) has been reached.

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

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

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

   Key Managers can perform key management services (like generation, deletion, backup and restore) for Assigned Keys and Storage Objects. These users must have the *user permission* which is defined by the configuration value 'AUTH_KEYM' (see section 3.2.2.5. The default setting is '00000002' which means that the *User* role and the *Key Manager* role is the same (also called '*Cryptographic Users'*) and both *Users* and *Key Managers* are allowed to perform all cryptographic and key management services.

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

   To execute the key management commands as provided by the PKCS#11 administration tool *p11toolv2* (e.g. *generateKey* or *importP12*, see chapter 5*)* the *user permission* as specified by 'AUTH_KEYM' is sufficient.

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

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

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

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

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

Upon correct authentication, the user's *AuthenticationFailureCounter* is reset to zero and the (command-individual or session-individual) authentication state will be augmented by the permissions of the user and thus the role (*Administrator, Security Officer, Key*

Manager, User or *Cryptographic User*) is selected based on the user name of the operator.

3.2.4.1 Default Administrator User ADMIN

On delivery of a CryptoServer from the manufacturer, in order to provide to the customer the possibility to do first administration tasks, CryptoServer contains exactly one user called "ADMIN". ADMIN is allowed to perform all standard administration and user management commands. In FIPS mode ADMIN is thus allowed to assume the Administrator role.

The authentication mechanism of ADMIN is 'RSA Signature Authentication'. The user permission of ADMIN is '22000000', i.e. authentication level 2 for user groups 6 and 7. A first private *Default Administrator Key* for user ADMIN is delivered with the CryptoServer (see 3.8.1), enabling ADMIN to authenticate towards the CryptoServer. Since this *Default Administrator Key* is not customer-individual, it should either be changed by the operator, or the ADMIN user should be replaced by other users with administrator rights:

After arrival of the CryptoServer device at the customer's site, user ADMIN should as soon as possible perform one of the following alternative actions:
Replace his Default Administrator Key by an individual authentication token, i. e. by an individual RSA key.
In case that e.g. the 2-persons-rule is wanted for the administration tasks: Replace the user ADMIN by two (or more) individul users with the necessary user permissions.
Details of this procedure are described in chapter 7.3.

If the default administrator user ADMIN shall be replaced, the sum of the permissions for user group 7 of all those of the newly created users who use RSA Signature Authentication mechanism has to be at least 2. Only if this permission level is reached, CryptoServer allows the deletion of user ADMIN.

Reason for this precaution is that in any situation the CryptoServer must remain administrable: Whenever an alarm occurs to the CryptoServer, or when the *Clear* command is performed (see 3.5 and 3.6; *Clear* is only available in personalization mode), all users with HMAC Password Authentication mechanism will be deleted from the user database. But the remaining users should be able to setup the CryptoServer again, in particular there must be enough users left who are able to authenticate the commands to reset the alarm and to create new users.

Additional administrator users with HMAC Password mechanism may be created, but they will be deleted in case of an alarm:

Any user with HMAC Password Authentication mechanism will be deleted in case of an alarm.

If a firmware update makes the CryptoServer enter or leave FIPS mode all sensitive data (including all users in the user database) are erased from the CryptoServer during next restart!

Also at any later date, the CryptoServer will only allow to delete a user with Administrator rights if the sum of the permissions of all remaining users who use RSA key as authentication token is still above the minimum level of '21000000' (which is the necessary condition to retain an administrable CryptoServer). Otherwise the command for user deletion will be rejected with an error code.

In emergency case when the administration key is lost (e.g. smartcard(s) with private part of customer-inidividual administrator key is lost), the CryptoServer can be reset by performing an external erase and executing the "Clear to Factory Defaults" command. In this case, the CryptoServer leaves FIPS mode, deletes the user database and creates a new one that contains the default ADMIN user described above. See 3.6 for details.

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 AES encrypted and integrity-protected with an AES MAC. In FIPS mode, Secure Messaging must be used for every command which has to be authenticated.

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

1. **Generate a session key.**

   First the external CryptoServer function *GetSessionKey* is called to generate an AES session key.

   The session key is negotiated with the *Diffie-Hellman* key establishment protocol (see [PKCS#3]). In FIPS mode the command to generate a session key must be authenticated by a valid user.

2. **Send encrypted commands.**

   The host can now send commands to the CryptoServer that are AES-encrypted and integrity protected with an AES MAC. 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 AES encryption and MAC calculation and is incremented after every command to prevent unauthorized replays of the commands.

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

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 command (i. e. a command using the 'secure messaging' layer of CryptoServer's protocol stack), 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.

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 the permission of these user(s) will be granted to the whole session. All commands that are encrypted with this session key inherit the permission of these users, without any extra authentication being necessary. But outside of 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 one of the following secure messaging commands:

- the *SessionDH* command (for Diffie-Hellman key agreement, usable by any user, see 4.11.1),
- one of the authentication commands *LogonSign* or *LogonPass,* which open an authenticated secure messaging session in a simplified way, see 4.10.1 and 4.10.2.

One of these secure messaging commands has to be used together in one command line with the command(s) for which secure messaging should be used. Then CSADM will automatically open the session (by getting the session key), perform the specific command(s) with secure messaging (i. e. encrypted and MAC-secured) and close the session on the CryptoServer again.

For details on the usage and syntax of secure messaging with the CSADM tool see section 4.11.

3.4 Auditing

The CryptoServer offers extensive audit functionality.

Some events will always and automatically be audited. Other events will optionally be audited if the CryptoServer is accordingly configured.

If an auditable event occurs, the operating system SMOS will automatically add an entry to the audit log file. The log entry always includes:

- a timestamp with date and time of the event (if the RTC is running),
- user name of all users who authenticated the audited command (if appropriate),
- function code (FC) and subfunction code (SFC) of the audited command (if appropriate),
- error code which indicates if the action has been successfully performed or if an error had occurred (if appropriate).

Additionally event-individual information may be stored, too.

The following events will be *always* and automatically logged:

Event	**audit entry contains additional information about:**
alarm	alarm reason
extreme high temperature	measured temperature
Clear command performed	
ResetAlarm command performed	

Additionally many more events may be logged if the CryptoServer's audit system is accordingly configured. In the last column it is indicated if the listed event is audited by a CryptoServer in default configuration:

Auditable event	**Message Class No.**	**Will be audited by default?**
Firmware and file management (e.g. commands to load or delete a file/firmware module)	1	yes
User management actions (e.g. commands to add or delete a user or to change the user's authentication token)	2	yes
Setting of CryptoServer's system clock	3	yes
CryptoServer Start-Up	4	yes
Deletion of Audit Log file	5	yes
Master Backup Key management (e.g. create or import a Master Backup Key)	6	yes

Auditable event	Message Class No.	Will be audited by default?
Key management (e.g. key management functions of firmware module CXI)	7	no
Successful authentications/logins	8	no
Failed authentications/logins	9	yes

Auditing may also be configured that way that only some (or none) of the above listed "default" events, or some of the "non-default" events will be logged: For this purpose a configuration file named "`audit.cfg`" can be loaded. This configuration file has to contain a list with all audit event classes that shall be logged (identified by their message class no.).

Additionally the following parameters of CryptoServer's audit functionality may be configured:

- Write logfiles rotatingly or not?[7] (default: write rotatingly),
- Number of audit log files (2 – 10) (default: 3)
- Length of one logfile (4,000 ≤ length ≤ 240,000 bytes) (default: 200,000 bytes).

For the format of the audit configuration file see [CSSMOS] or ask Utimaco for help.

- *CryptoServer's audit logfile can any time and in any mode be read with command* GetAuditLog, *see 4.6.12.*
- *CryptoServer's audit configuration can be read with command* GetAuditConfig at any time except in FIPS Error State, *see 4.6.20.*
- *CryptoServer's audit logfile can be deleted with command* ClearAuditLog *(if appropriately authenticated), see 4.6.13.*

The audit log files are stored in CryptoServer's flash memory. Regarding the configuration of auditing, it should always be kept in mind that a too extensive auditing may result in a great wear and tear of the flash memory and therefore in a decreased lifetime of the CryptoServer.

[7] The audit logfile is organized in several files. If the audit logfiles are written *rotatingly*, this means that in case that the last log file is full CryptoServer will start to overwrite the first logfile etc. If the audit logfile is configured to *not rotating*, this means that in case that the last logfile is full CryptoServer will refuse to execute any auditable functions except for deleting the audit logfile. If the audit logfile is deleted, this will be logged as first new entry.

3.5 Alarm

Alarm state is caused by certain extraordinary physical circumstances. It has not to be mixed up with the above mentioned global states (initialized, defect) the CryptoServer can be in.

Anytime a physical alarm happens to the CryptoServer, it will immediately be detected by the sensory which triggers the defined alarm mechanism (see below for details). Part of this mechanism is a restart of the CryptoServer. *Alarm* is given if the operating system SMOS detects an alarm condition in the alarm status register during the boot process. All sensitive data will be deleted and the CryptoServer will leave FIPS mode. The CryptoServer responds with the current alarm condition and a *ResetAlarm* command is required to reset the alarm status register. The alarm reason can be output via the *GetState* command.

After an alarm, the CryptoServer will leave FIPS mode and enter personalization mode.
To enter FIPS mode again, the whole process to set-up and personalize the CryptoServer has to be performed, see 8.2 and 7.3.

Possible reasons for alarm are:

Abbreviation	Explanation
Temp_low	Temperature too low (see also 3.7)
Temp_high	Temperature too high (see also 3.7)
Pow_high	Voltage / tension too high (CryptoServer-internal battery)
Pow_low	Voltage / tension too low (CryptoServer-internal battery)
ext_Erase	External deleting / clearing done
inval_MK	Invalid (corrupted) device-internal *Master Key* (reason for this is usually an empty battery, see below)
Power failed	Sensory controller without power.
Sensory Controller failed	No reaction from sensory controller

The following alarms are generated by the CryptoServer CSe only:

Inner foil damaged	Chemical or mechanical attack has damaged the inner foil
Outer foil damaged	Chemical or mechanical attack has damaged the outer foil

Whenever a physical alarm occurs (noticed by the sensory), a dedicated *Alarm-Bit* in the alarm status sensory register will be set immediately, together with bits which indicate the alarm reason.

The CryptoServer's internal *Master Key* will be deleted immediately (i. e. the Key-RAM will be actively erased by the sensory controller) and the CryptoServer will be restarted.

Independently from the occurrence of an alarm, the CPU's internal RAM (IRAM) will be erased at the very beginning of the boot procedure. Clearing of Key-RAM and IRAM will be finished within less than 4 msec after the occurrence of the alarm.

If during the (following) boot process the operating system SMOS finds an alarm in the alarm status register which is not yet logged (i. e. the *Alarm-Bit* is still set), first all security-relevant data inside the CryptoServer will be deleted. Only some specific data like firmware and the system keys remain.

In case of an alarm, all security-relevant secret data inside the CryptoServer will be deleted. In particular all data that is stored encrypted with CryptoServer's Master Key will be deleted.
The only data that remain stored are the following:

- *all firmware modules, except for the FIPS control module FIPS140*
- *the user database, except for all users who use the HMAC Password Authentication mechanism (i. e. in particular all passwords have been deleted)*
- *all system keys (like Module Signature Key, Default Administrator Key, Production Key)*
- *the audit logfile 'audit.log' and its configuration file 'audit.cfg'*
- *boot loader configuration file 'bl.cfg', including the EID (Extended ID)*
- *any signed licence files '*.slf'.*

In particular, CryptoServer's internal Master Key is erased.

After the deletion of all other data SMOS continues to start all system firmware modules (*.sys), which means that in alarm state the CryptoServer is running in *maintenance mode:* only the back-up copies '*.sys' of the basis firmware modules are running. No secret or private key is left. Since the firmware module DB for database management is blocked without the *Master Key*, no further keys can be loaded in alarm state. Most CryptoServer functionality is blocked:

Commands that are available in alarm state *comprise only all administrative commands which do not have to be authenticated (like all commands which return non-sensitive information), plus the commands* ResetAlarm *and* SetTime*.*

If possible, please try to eliminate the physical cause of the alarm.

After that, to regain full functionality a ***ResetAlarm command*** *has to be performed and the CryptoServer has to be restarted. Not until then CryptoServer will react again on further commands.*

Thus each alarm must explicitly be reset by the user using the command *ResetAlarm* before the CryptoServer accepts further sensitive commands. See section 4.6.19.

As part of *ResetAlarm*, the alarm will be logged (audit logfile 'audit.log'), including date and time (up to seconds) and alarm reason. This logfile can anytime and in any mode be read out with the help of the *GetAuditLog* command. With *ResetAlarm* the *Alarm-Bit* will be set back (to demonstrate that the alarm has already been logged) and a new *Master Key* will be generated.

Since the *ResetAlarm* command has to be authenticated by a user with administrative rights, it is ensured that for any alarm at least one responsible person has been aware about the occurence of the alarm. Additionally, *ResetAlarm* will add an entry to

CryptoServer's audit log file which contains the user name of the user who authenticated the command.

In case of a physical alarm which is still present (e. g. foil damages or temperature still too high) the Alarm-Bit will immediately be set again. The Key-RAM will be erased and the CryptoServer restarted. Thus the procedure described above will repeat if the cause of the alarm is not eliminated. Therefore in case of a permanent alarm for which the reason can't be eliminated (e.g. foil damages) the CryptoServer has to be sent back to the manufacturer. Please contact Utimaco.

If the CryptoServer is stored for a long period of time without voltage supply and the battery gets empty, the CryptoServer-internal Master Key will be deleted according to the alarm mechanism. However, information about this alarm state will be lost, too, as the content of the sensors register is also lost in the absence of voltage. Therefore the boot loader additionally checks at each boot process the integrity of the loaded *Master Key*. If the verification fails, the boot loader will then activate a 'virtual' alarm. This alarm is displayed as 'Inval_MK', see above.

For the accurate procedure "What to do?" in case of an alarm see chapter 8.2.

3.5.1 External Erase

To offer the possibility for deliberate and manual erasure of all sensitive data inside the CryptoServer and to leave FIPS error state, an alarm may artificially be triggered by performing an *External Erase*.

The execution of an External Erase is the precondition for a ClearToFactoryDefaults *to be performed, see 3.6.*

For a CryptoServer PCIe-card, External Erase has to be done manually by a short-circuit of the 'External Erase' pins on the PCIe-card.

For a CryptoServer LAN, an External Erase will be performed by pressing the sealed delete switch inside the battery compartment of the device (see chapter 4 in [CSLAN-InstallManual]

).

3.6 Clear Commands

CryptoServer provides with its *Clear* command the possibility to erase the CryptoServer to a previous state.

For various purposes there are two variants of the *Clear* command implemented (see 4.6.18):

The ClearToFactoryDefaults command will set the CryptoServer back to the factory default setting in which it usually is delivered. In particular all customer-individual data, including firmware, keys and customer-individual users will be deleted.
This command may be used by the customer in case that the customer-individual administrator keys have been lost, because it offers the possibility to get an administrable CryptoServer again. But the usage of the command should be restricted to this emergency case only!
The normal Clear command gives a similar result than the ClearToFactorySetting command, with the exception that all users in the user database that use an RSA key as authentication token will remain (i.e. only users with HMAC Password mechanism will be erased).

Both commands are only available in personalization mode, to be used during the process of set-up and personalization of the CryptoServer before FIPS mode is entered:

The Clear and ClearToFactoryDefaults commands are not available in FIPS mode!
They can only be performed in personalization mode.

The normal *Clear* command (syntax ‘csadm Clear’, see 4.6.18) will

- erase the CryptoServer’s internal *Master Key*,
- erase all firmware modules (except for the boot loader code and system firmware modules ‘*.sys’),
- erase all data except for
  - the public parts of the *Production Key*, *Default Administrator Key* and *Module Signature Key*,
  - the alarm state file (alarm.sens)
  - the audit log file,
  - users with RSA key as authentication token (as stored in user database)
  - and the boot loader configuration file ‘bl.cfg’.

Afterwards a reboot must be performed.

This command has to be authenticated by a user with administrative rights.

If the customer has lost his customer-individual administrator keys, the *ClearToFactoryDefaults* command has to be used instead.

The ***ClearToFactoryDefaults*** command (syntax ‘csadm Clear=DEFAULT’, see 4.6.18) will

- erase all data like the normal *Clear* command does, see above, and will additionally
- erase *all* users from the user database, including users with RSA key as authentication token.

*After a **ClearToFactoryDefaults** command all users have been erased from the user database, except for the default administrator user ADMIN who will be restored with the Default Administrator's Key as authentication token.*

This *ClearToFactoryDefaults* command does not need to be authenticated, but it can be executed only immediately after an *External Erase* was performed, see 3.5.1. This assures that only persons with physical access to the CryptoServer hardware have the possibility to clear the CryptoServer from all non-default users.

Then a reboot must be performed, and the alarm (emerging from the *External Erase*) has to be reset (*ResetAlarm* command). The CryptoServer is now in the factory default settings in which it usually is delivered (and with all basis firmware modules loaded as *.sys-files only, and with the default administrator ADMIN as the only user). In particular the CryptoServer is in personalization mode.

*The **ClearToFactoryDefaults** command has to be used by the customer in case that the customer-individual administrator keys have been lost, or in case of FIPS error state, because it offers the possibility to get an administrable CryptoServer again. But all customer's data are erased.*

The usage of this command should be restricted to these two emergency cases only!

3.7 Behavior of CryptoServer Outside of the Normal Temperature Range

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

Temperature	Behavior of the CryptoServer
below –13°C	Alarm is triggered (all sensitive data on the security module will be cleared) and the security module is restarted. After the restart CryptoServer's processor is suspended, commands will not be executed any longer.
–13°C to 62°C	Normal operation.
62°C to 66°C	CryptoServer's processor is suspended, commands will not be executed any longer.
above 66°C	An alarm is triggered (all sensitive data on the security module will be cleared) and the security module is restarted. After the restart CryptoServer's processor is suspended, commands will not be executed any longer.

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

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

Once the CryptoServer's processor is suspended (i.e. outside of the operational temperature range), it does not respond to any request. Any attempt to access the CryptoServer will usually result in a kind of timeout error from the device driver.

Before the CryptoServer's processor is suspended the boot loader writes an entry into the audit log file which can be read out with the administration command *GetAuditLog*.

The only way to get the CryptoServer out of the suspend mode is to reset it using the *Restart* command.

Resetting the CryptoServer has no effect if the temperature is still outside of the normal operational range. In case of CryptoServer's suspend mode caused by high temperature, it is recommended to switch the supply power off for some time in order to cool the CryptoServer down.

3.8 System Keys

This chapter will describe the system keys used in and around a CryptoServer, how they are generated, who will be responsible for storing them, and the way they are handled and used.

The user keys (user's authentication token) and the Master Backup Key (MBK) are not described herein.

3.8.1 Default Administrator Key

As soon as possible after shipping, the Default Administrator Key ought to be used to perform first administrative tasks, including to set-up a customer-individual CryptoServer.

In case that the customer has lost his customer-individual administrator authentication token(s), the Default Administrator Key offers a fall-back possibility to get an administrable CryptoServer again.

You will find the key (incl. private part) on the Product CD, key file 'init_dev_prv.key', and on the delivered smartcards.

type:	The *Default Administrator Key* is an RSA key of 1024 bits size.
generation:	The generation will be done by the manufacturer, outside of the CryptoServer. The key is manufacturer-specific, but it is a default key and therefore neither customer-individual nor CryptoServer-individual. As part of CryptoServer shipment, the manufacturer will hand over the key (public and private part) to the customer.
usage:	The key is needed to offer first administration possibilities: The key *can* be used by default administrator user ADMIN to authenticate the security relevant administration commands (commands e. g. for firmware and user management, reset alarm, set CryptoServer time) as long as the CryptoServer is still in *factory default setting*. The key *ought to be used* immediately after shipping to set-up a customer-individual CryptoServer, either by replacing the default authentication token of user ADMIN (who by default uses the *Default Administrator Key* as authentication token) by a customer-individual key, or by replacing the default administrator user ADMIN by customer-individual users with sufficient administrator rights.
life cycle:	The key is imported in the CryptoServer by the manufacturer as part of the production process. One copy of the public part of the *Default Administrator Key* is stored in the user database as default authentication token of the

	administrator user ADMIN. This copy can be overwritten (*ChangeUser* command, which has to be authenticated by ADMIN himself) or deleted (command *DeleteUser*, which can only be performed after sufficent further users with administrator rights have been created before, see 3.2.4.1). One of these options should be done by the customer as soon as possible after receiving the CryptoServer. A second copy of the public key is stored in flash device (file 'init.key'). This copy can neither be changed nor deleted, but it will be used to restore the default administrator user ADMIN with his *Default Administrator Key* as authentication token (using the *ClearToFactoryDefaults* command, see 3.6). Also in case of an alarm this copy of the key will not be deleted.

3.8.2 Module Signature Key

The Module Signature Key is used by the CryptoServer to verify the integrity and authenticity of the firmware modules, i. e. its origin by the manufacturer.

The Module Signature Key will not be used by the customer.

type:	The *Module Signature Key* is a RSA key of 4096 bits size.
generation:	The generation will be done by the manufacturer, outside of the CryptoServer.
life cycle:	The private part of the *Module Signature Key* will be stored in a safe environment at the manufacturer's site where only authorized personnel has access. The public part of the key will be imported into the CryptoServer by the manufacturer as part of the production process. It will be stored in the file system area of the CryptoServer flash file (file 'mdlsig.key'). The *Module Signature Key* can neither be exported nor changed nor deleted by the customer. Also in case of an alarm, or a *Clear* command being executed, the key will not be deleted.
usage:	The private part of the *Module Signature Key* is used by the manufacturer for the signature of any MMC (Module Manufacturer Container, see 3.10.1). A MMC contains the executable of a CryptoServer firmware module. The signature is used to protect the integrity and ensure authenticity of the MMC during download, i. e. its origin by the manufacturer. The public part of the *Module Signature Key* is stored inside the CryptoServer. During firmware download, this key or the customer-specific *Alternative Module Signature Key* (which is not available in FIPS mode) will be used to verify the MMC signature. The private part of the *Module Signature Key* will as well be used by the manufacturer for the signature of any Signed Licence File (SLF, see 2.2.3 for details). Whenever a '*.slf' is downloaded, the public part of the *Module Signature Key* will be used to verify the SLF signature, and thus to verify the integrity and authenticity of the SLF, i. e. its origin by the manufacturer.

3.8.3 CryptoServer's Internal Master Key

The customer will never use CryptoServer's internal Master Key directly. The CryptoServer security module itself is completely responsible for generation, usage and erasure (in case of alarm) of the Master Key.
The Master Key is CryptoServer-individual and will never leave the CryptoServer.

type:	CryptoServer's internal *Master Key* is a 32 bytes AES key.
generation:	Generation of the key will be done automatically inside of the CryptoServer, as part of the production process. The CryptoServer security module itself is responsible for usage and life cycle of this key.
usage:	The internal *Master Key* is needed to encrypt data inside of the CryptoServer.[8] For this purpose it will be used internally by other firmware modules. The database firmware module (module DB) facilitates the usage of the key. This DB module provides an internal public interface for secure data storage to other application firmware modules.
life cycle:	The *Master Key* cannot be imported but will be generated inside of the CryptoServer. After generation, the key is stored inside of the CryptoServer in (sensory-protected) Key-RAM (see 2.1.2). Key export is not possible. The *Master Key* will never be available outside of the CryptoServer. The life cycle of this key ends when an alarm occurs to the CryptoServer. If any alarm cause (temperature, voltage, …) is detected by CryptoServer's sensory, the memory area in which the key is stored will be automatically erased. A new *Master Key* is generated during the next normal boot process (after successful performance of the *ResetAlarm* command). The key is also erased with all *Clear* commands, see 3.6: Any Clear command erases the *Master Key* by generating and storing a new *Master Key* (overwriting the old one).

[8] This is necessary in order to be able to store sensitive data also in memory areas which are not sensory-protected, i. e. which will not be erased within a very short time after an alarm has occurred. These are e. g. the flash device, NV-RAM and SD-RAM(Se)/DDR2-RAM (CSe).

3.9 Backup of Keys and Users

In many applications, for security reasons or to setup a redundant CryptoServer with identical data, there is a need to create a backup of the cryptographic keys that are stored in a hardware security module, and/or to create a backup of the registered users and their user data including their authentication data (public key or password). To support backup functionality, CryptoServer provides the possibility to use *Master Backup Keys*:

A Master Backup Key may basically be used to encrypt secret data which is exported from the CryptoServer (e.g. create a key back-up), or to decrypt data which is imported into the CryptoServer (e.g. restore a key back-up). This will e.g. be done by Utimaco's application firmware module CXI. It can also be used to backup and restore a user and his user data (see commands *BackupUser* and *RestoreUser*, 4.7.7 and 4.7.8).

CryptoServer is able to store up to four Master Backup Keys (slot 0..3) to be used by various applications. In FIPS mode an AES Master Backup Key (MBK) in slot 3 has to be used (16, 24 or 32 bytes).

The handling of such Master Backup Keys is implemented over the CryptoServer firmware module MBK. The CSADM commands for MBK management will be described in chapter 4.8.

3.10 Firmware Module Management

In this section the management of CryptoServer's firmware modules is described from a security point of view.

With the aim to link the firmware with administrative information (e. g. module name and version number) and to add check values for the authenticity and integrity of the firmware every firmware module is enveloped into a so called *container*. These containers allow also the download and storage of encrypted firmware.

For easier handling, Utimaco offers also so-called *package files* in which a set of firmware modules (the containers described below) is bundled, ready for download. See 3.10.2 for the concept and usage of these firmware packages.

3.10.1 Firmware Containers: MTC/MMC, MSC, SYS

A raw *firmware module* (RFM) is the executable module compiled for the CryptoServer platform, i. e. ready for interpretation of the CryptoServer operating system SMOS. This can be COFF (*.out), as used by the CryptoServer hardware, or DLL (*.dll), as used by the CryptoServer SDK (Software Development Kit).

For various purposes (firmware transport, download, and storage inside the CryptoServer) this RFM will be enveloped in so-called firmware module containers:

- For firmware download the RFM has to be enveloped into a **MTC** (*module transport container*). This MTC contains a **MMC** (*module manufacturer container*) which has mandatorily to be signed.
- For secure and authentic transport the MTC may optionally be signed.
- After download, the CryptoServer will unwrap the RFM from the MMC/MTC and store it in form of a **MSC** (*module storage container*).

The **MMC** adds a header with module information to the RFM and a signature that guarantees for the integrity and authenticity of the module. The signature has to be calculated with Utimaco's *Module Signature Key*, it will be verified during firmware download.

The **MTC** adds a second header to the MMC (which contains additional module transport information) and optionally a second signature which is calculated over the complete MMC. The MTC structure and signature can be used to secure the transport of the firmware module from the developer to the customer (to guarantee for the authenticity of the module during delivery).

The **MSC** format is for CryptoServer-internal use only: It stores the module code together with certain module information and the check value for the module code's integrity (SHA-512 hash value over the module code).

Backup-copies of all system-relevant firmware modules are always stored in every CryptoServer as **SYS** modules (system firmware modules). SYS files have the same format than MSC, but will be stored with file extension *.sys. It is impossible to delete, load or change a SYS file.

3.10.2 Package Files

In order to simplify firmware management Utimaco offers firmware also in so-called *package files*: A package file "*.mpkg" can contain several firmware modules (e. g. MTCs) as well as other files in a packed form.

Package files are intended to give the user a facile possibility to download or update a set of several firmware modules and/or files into the CryptoServer in only one step.

For this purpose the CryptoServer's administration tool CSADM offers the *LoadPkg* command (see 4.6.16) which can replace a series of succeeding CSADM commands: The *LoadPkg* command loads the contents of the given package file "*.mpkg" into the CryptoServer, adding to or replacing existing firmware.

Furthermore the CSADM tool offers commands to create a package file from a collection of MTCs or to list its contents, to retrieve the contained firmware modules from a "*.mpkg" file, and to compare the contents of a package file with the firmware that is already stored in the CryptoServer (commands *Pack* 4.4.6, *ListPkg* 4.4.8, *Unpack* 4.4.7, *CheckPkg* 4.6.17). Additionally you can also check the MMC signature of each firmware module contained in a package (*VerifyPkg* 4.4.9).

For the package file for FIPS Approved mode of operation see chapter 10.

4 Administration of CryptoServer with CSADM

The *CryptoServer Administration Tool* (CSADM) is a command line utility designed for being called from the command line or in a batch file. It offers functions either to execute administrative commands on the CryptoServer or the CryptoServer LAN. In addition it contains utility functions which will be processed without a connection to a CryptoServer (e. g. preparation of firmware modules).

4.1 CryptoServer Administration Tool CSADM - General

4.1.1 System Requirements

The following requirements have to be made for the CSADM:

PC-Hardware:

- no special requirements as far as memory and CPU performance is concerned.
- network interface card to access CryptoServer LAN
- a free USB or serial port to connect the PIN pad (smartcard reader with keyboard and display).

PC-Software (OS):

- Microsoft Windows (2000, XP, W2K3, Vista or W2K8)
- Linux (X86 or PPC)
- Solaris (Sparc, X86)
- AIX (PPC)

4.1.2 Installation of CSADM

The installation of the Administration Tool CSADM is quite simple.

Installation on a computer with a Windows operating system:

1. Copy the file ‘csadm.exe’ to a well-chosen directory.
2. Add this directory to the ‘PATH’ environment variable to be able to call the Administration Tool from any other directory.
3. 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 4.1.3).

Installation on a computer with a Unix like operating system (Linux , Solaris, AIX):

1. Copy the executable file ‘csadm’ to a well-chosen bin directory.
2. 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 4.1.3).

4.1.3 Syntax of 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 squared brackets [ ] are optional.

In the syntax describing line of the individual command descriptions all parameters are shown in angle brackets < > and are explained later.

Some parameters or commands require an assigned value ('=...'), some do not.

Some commands use a default value if none is given ('[=...]').

Examples:

- **csadm dev=PCI:0 GetState**
- **csadm dev=TCP:192.168.1.98 LogonSign=ADMIN,d:\keys\cs2\init_dev_prv.key …LoadFile=c:\firmware\exmp.mtc**
- **csadm dev=192.168.1.1 LogonPass=paul,ask …AddUser=bob,00000002,no_login+sma,ask**
- **csadm dev=TCP:288@154.54.2.23 LogonSign=ADMIN,:cs2:cyb:COM2 ...SetTime=20020602111532**

Every command running on CryptoServer or CryptoServer LAN requires the 'Dev=' parameter (device parameter, see also 4.3) which sets CryptoServer's or CryptoServer LAN's address. Commands running locally without using a CryptoServer (like module preparation commands, see 4.4) do not need this parameter. Possible values are:

Address	Description
`PCI:/dev/cryptoserver0`	local CryptoServer No. 1 in a Linux system.
`PCI:1`	local CryptoServer No. 2 in a Windows system.
`TCP:288@194.168.4.107`	IP-Address and port number of a CryptoServer LAN
`TCP:194.168.4.107`	IP-Address of a CryptoServer LAN (default: port=288)
`194.168.4.107`	IP-Address of a CryptoServer LAN (default: protocol=TCP, port=288)
`TCP:288@cslan01`	host name and port number of a CryptoServer LAN (using DNS request to resolve host name)
`TCP:cslan01`	host name of a CryptoServer LAN (using DNS request to resolve host name, default: port=288)

Address	Description
`cslan01`	host name of a CryptoServer LAN (using DNS request to resolve host name, default: protocol=TCP, port=288)
`TCP:3001@127.0.0.1`	local CryptoServer simulator

If the environment variable CRYPTOSERVER is set according to the above mentioned syntax, the 'Dev=' parameter can be skipped (see also 4.1.2). Using the 'Dev=' parameter overrides the CRYPTOSERVER environment variable in any case for the specific command.

4.1.4 Key Specifiers

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

(1) Authentication keys stored in a file '`*.key`' (as plain text).

(2) Authentication keys stored in an encrypted key file '`*.key`' (private key part stored encrypted, public key part stored as plain text).

(3) Authentication keys stored on a smartcard.

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

For security reasons private signature keys should normally be used only from smartcards or encrypted key files. In a test environment, where the private key does not need to be kept secret, it may be useful to store the keys in (plaintext) files.

Encrypted RSA signature key files are protected by a 168 bit Triple-DES key that is derived from a password with the SHA-256 hashing algorithm. The password can be changed with the *ChangePassword* command (see 4.8.1). With the same command a plaintext key file can be changed in an encrypted key file and vice versa (by omitting the old respectively the new password).

Apart from test environments, it is strongly recommended to store private keys only on smartcards! Only in this case the private key will never leave the secure token and not be used for calculations on the host.

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

- *a filename of a key file ('`*.key`') or*
- *a smartcard specifier.*

Key File Specifiers:

In case that the private part of the key is needed for the command and is given in an encrypted key file, the password of the key file has to be given in the command syntax, too. This can be done either by giving a separate password parameter (Example 1) or by appending the password directly after the filename, separated by a '**#**' (Example 2):

Example 1: **csadm Password=silence LogonSign=ADMIN,c:\keys\init_prv.key ResetAlarm**
Example 2: **csadm LogonSign=ADMIN,c:\keys\init_prv.key#silence ResetAlarm**

For some commands only the second syntax variant is possible (e.g. if several key specifiers are involved), as described in the relevant command syntax description.

In both cases hidden password entry can be used (see 4.1.5):

Example 1: **csadm Password=ask LogonSign=ADMIN,c:\keys\init_prv.key ResetAlarm**
Example 2: **csadm LogonSign=ADMIN,c:\keys\init_prv.key#ask ResetAlarm**

Smartcard Specifiers:

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

- The first string identifies the type of the smartcard.
- The second string identifies the type of the smartcard reader (PIN pad).
- The last string is the name of the serial device the reader is connected to.

Smartcard types supported at the moment are the following types:

Identifier	Smartcard type
`cs2`	CryptoServer smartcard (using TCOS)
`cos`	CardOS smartcard
`nkey`	Telesec NetKey Card

At the moment supported reader types are:

Identifier	Smartcard reader type (PIN pad)
`cyb`	Reiner SCT cyberJack
`cp8`	Ingenico/Bull SafePad
`acr`	Advanced Card System ACR80
`cm8`	Omnikey CardMan 8630

Key specifier examples:

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

4.1.5 Password Entry

To authenticate a security-relevant administration command, an operator who uses the HMAC Password authentication mechanism has to enter his user name and password to the CSADM tool (see 4.10). At this and other opportunities 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, CSADM offers the possibility for hidden password entry.

For hidden password entry, instead of the password first the string ‘ask’ has to be entered. Then CSADM will, before starting to process the authentication (and the rest of the command), return and prompt for the password separately.

Example:

csadm LogonPass=paul,ask SetTime=GMT

Enter Passphrase:

If now the password will be entered over the keyboard, it will not be reflected in clear on the monitor, but be hidden by the display of default characters.

The hidden password entry mechanism can also be used to protect the password of an encrypted key file (see 4.1.4) or the root password of the CryptoServer LAN from being displayed on the monitor.

The usage of hidden password entry is strongly recommended!

4.1.6 Command Execution with the CSADM Tool

If the CryptoServer is installed on the local computer (as PCIe card) commands will be sent from the host to the CryptoServer via the PCIe interface. The CryptoServer processes the command and sends the answer (answer data or error code) back to the host.

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

The following illustration shows how commands are executed on a local CryptoServer PCIe or a remote CryptoServer LAN:

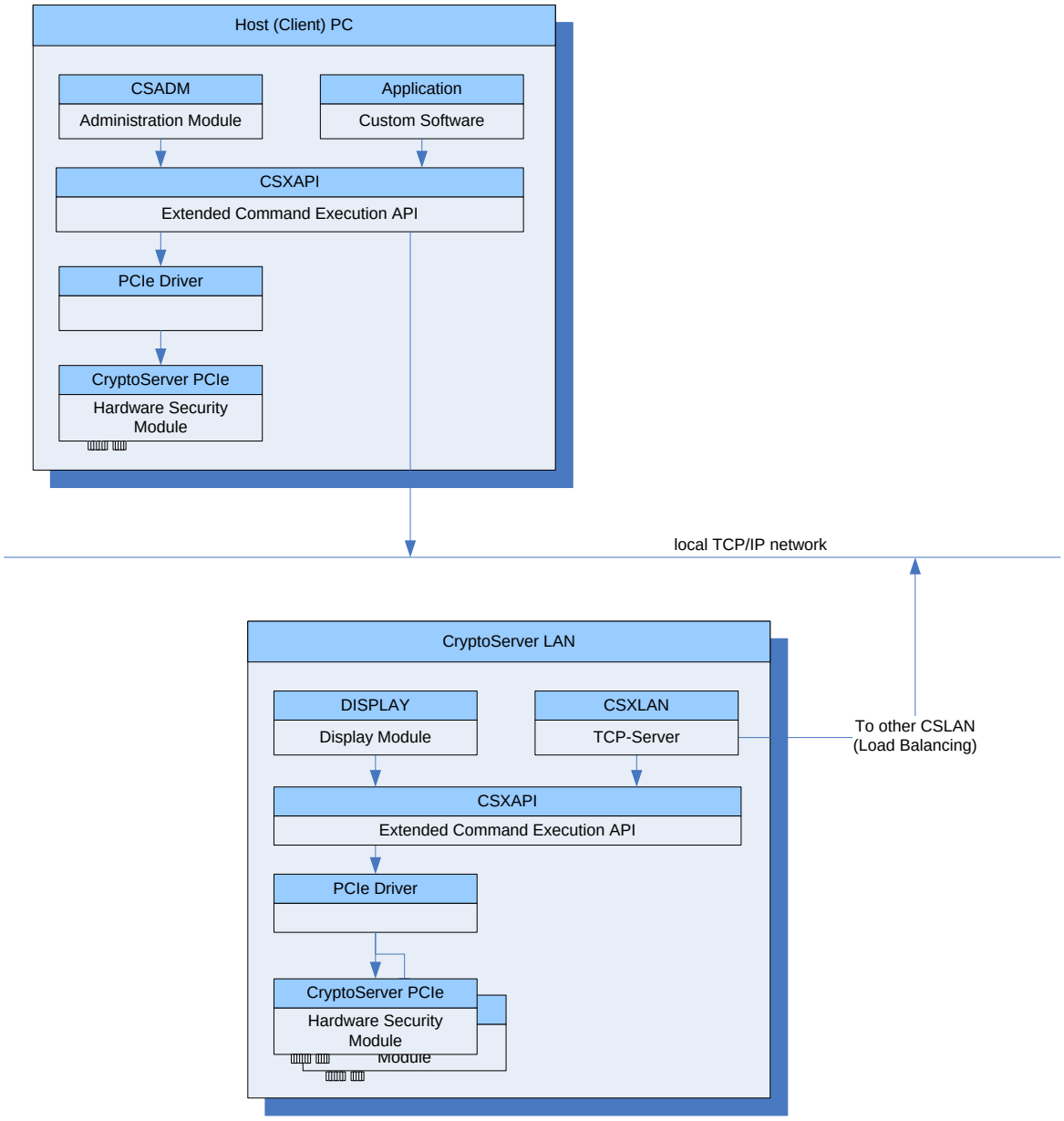


Illustration 4-1: Command Execution

An application on the host PC can use the *extended command execution library* CSXAPI to execute any command on a CryptoServer PCIe or CryptoServer LAN.[9]

An application on the host PC can use the FIPS-specific library for cryptographic services (CXI library, which also contains the *extended command execution library* CSXAPI) to execute any of the cryptographic commands which are offered by the CryptoServer if run in FIPS mode. (Users who are allowed to assume the role *Cryptographic User* have the right to perform these cryptographic services.)

As a standard application the CryptoServer *Administration Tool* CSADM is available to provide any kind of basic administration like file download or deletion, setting of the CryptoServer's clock, user management a. s. f (see the following subsections).

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

Commands can be divided into the following classes:

Command Destination	Counterpart on the CryptoServer	Command Group
CryptoServer	Command Scheduler Module (CMDS)	authentication, user management
	Administration Module (ADM)	administration of CryptoServer
	MBK Management Module (MBK)	management of Master Backup Keys
	CXI Module (CXI)	cryptographic services (not realized in CSADM)
TCP Server of the CryptoServer LAN	Control module of the TCP Server (daemon) *csxlan*	administration (configuration) of the TCP Server ('daemon') *csxlan*

[9] CSXAPI (CryptoServer Extended Application Programming Interface) is a software running on the host (see page 14f, Illustrations 2-2 and 2-3: Overview of the CryptoServer Se/CSe software architecture) which has the job to generate a C-interface out of the external CryptoServer byte stream interface.

4.1.7 Availability of Commands in FIPS Mode

The following chapters describe CSADM commands which are available in personalization mode. In FIPS mode the CryptoServer blocks some of these commands. The following table gives an overview about all relevant CSADM commands and indicates if these commands are available in FIPS mode:

chapter		Available in FIPS mode	Available in FIPS Error State
4.2.1	Help	X	X
4.2.2	PrintError	X	X
4.2.3	Version	X	X
4.4	Commands to Prepare Firmware Modules		
4.4.1	MakeMTC	X	X
4.4.2	RemoveMTC	X	X
4.4.3	VerifyMTC	X	X
4.4.4	ModuleInfo	X	X
4.4.5	RenameToVersion	X	X
4.4.6	Pack	X	X
4.4.7	Unpack	X	X
4.4.8	ListPkg	X	X
4.4.9	VerifyPkg	X	X
4.5	CryptoServer Driver Commands		
4.5.1	Reset	X	X
4.5.2	ResetToBL	not available	not available
4.5.3	Restart	X	X
4.5.4	GetInfo	X	X
4.5.5	SetTimeout	X	X
4.6	Commands for Administration		
4.6.1	GetState	X	X
4.6.2	GetBattState	X	X
4.6.3	StartOS	X[10]	X[10]
4.6.4	RecoverOS	X[10]	X[10]
4.6.5	ListFiles	X	X
4.6.6	LoadFile	X	blocked
4.6.7	DeleteFile	X	blocked
4.6.8	GetTime	X	X
4.6.9	SetTime	X	blocked
4.6.10	ListModulesActive	X	X
4.6.11	GetBootLog	X	X
4.6.12	GetAuditLog	X	X
4.6.13	ClearAuditLog	X	blocked
-4.6.20	GetAuditConfig	CSe: X Se: blocked	blocked
-	SetAuditConfig	blocked	blocked

[10] If the CryptoServer is not in boot loader mode, this command is ignored.

chapter		Available in FIPS mode	Available in FIPS Error State
4.6.14	MemInfo	X	X
4.6.15	Test	X	X
4.6.16	LoadPkg	X	blocked
4.6.17	CheckPkg	X	X
-	LoadFWDecKey	blocked	blocked
-	LoadPubKey	blocked	blocked
4.6.18	Clear	blocked	blocked
4.6.19	ResetAlarm	blocked	blocked
-	BackupDatabase	blocked	blocked
-	RestoreDatabase	blocked	blocked
4.7	User Management		
4.7.1	ListUser	X	blocked
4.7.2	AddUserRSASign	X	blocked
4.7.3	ChangeUserRSASign	X	blocked
-	AddUserClrPwd	blocked	blocked
-	ChangeUserClrPwd	blocked	blocked
-	AddUserSHA1Pwd	blocked	blocked
-	ChangeUserSHA1Pwd	blocked	blocked
-	AddUserRSASC	blocked	blocked
-	ChangeUserRSASC	blocked	blocked
4.7.4	AddUserHMACPwd	X	blocked
4.7.5	ChangeUserHMACPwd	X	blocked
-	AddUserECDSA	blocked	blocked
-	ChangeUserECDSA	blocked	blocked
4.7.6	DeleteUser	X	blocked
4.7.7	BackupUser	X	blocked
4.7.8	RestoreUser	X	blocked
4.7.9	SetMaxAuthFails	X	blocked
4.7.10	GetMaxAuthFails	X	blocked
4.8	User Key Management		
4.8.1	GenKey	X	X
4.8.2	SaveKey	X	X
4.8.3	BackupKey	X	X
4.8.4	CopyBackupCard	X	X
4.8.5	GetCardInfo	X	X
4.8.6	ChangePassword	X	X
4.8.7	ChangePIN	X	X
4.9	Management of Master Backup Keys		
4.9.1	MBKListKeys	X	blocked
4.9.2	MBKGenerateKey	X	blocked
4.9.3	MBKImportKey	X	blocked
4.9.4	MBKCopyKey	X	X
4.9.5	MBKCardInfo	X	X
4.9.6	MBKCardCopy	X	X
4.9.7	MBKPINChange	X	X
4.10	Command Authentication		
4.10.1	LogonSign	X	blocked

chapter		Available in FIPS mode	Available in FIPS Error State
4.10.2	LogonPass	X	blocked
4.10.3	AuthRSASign	X	blocked
-	AuthClrPwd	blocked	blocked
-	AuthSha1Pwd	blocked	blocked
-	AuthRSASC	blocked	blocked
4.10.4	AuthHMACPwd	X	blocked
-	AuthECDSA	blocked	blocked
4.10.5	ShowAuthState	X	blocked
-	Login	blocked	blocked
-	Logoff	blocked	blocked
4.11	Secure Messaging		
-	SessionRSA	blocked	blocked
-	SessionEC	blocked	blocked
-	SessionPwd	blocked	blocked
-	SessionHMAC	blocked	blocked
4.11.1	SessionDH	X	blocked
-	SessionECDH	blocked	blocked
4.12	Administration of the CryptoServer LAN		
4.12.1	CSLGetConnections	X	X
4.12.2	CSLScanDevices	X	X
4.12.3	CSLGetVersion	X	X
4.12.4	CSLGetLogFile	X	X
4.12.5	CSLGetConfigFile	X	X
4.12.6	CSLPutConfigFile	X	X
4.12.7	CSLSetTracelevel	X	X
4.12.8	CSLShutdown	X	X
4.12.9	CSLReboot	X	X
4.12.10	CSLGetTime	X	X
4.12.11	CSLSetTime	X	X
4.12.12	CSLGetSerial	X	X
4.12.13	CSLGetLoad	X	X
4.12.14	CSLListPPApps	blocked	blocked
4.13	Miscellaneous Commands		
4.13.1	Cmd	X	X
4.13.2	CmdFile	X	X
4.13.3	CSTerm		
4.13.4	Sleep		

In the following chapters all CSADM commands that are relevant for usage of CryptoServer in FIPS mode will be described in detail. This includes several commands which are not available in FIPS mode but which have e.g. to be used for the process to set-up CryptoServer in FIPS mode or for error handling.

4.2 Basic Commands

These basic commands are CSADM internal functions.
No connection to a CryptoServer will be established.

4.2.1 Help

If called without any parameter, this command shows a list of all available CSADM commands. If a 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]**

4.2.2 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, CryptoServer LAN, PCIe-Driver and Host-APIs.

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**

4.2.3 Version

This command shows the version numbers of the CSADM tool and the included libraries.

Syntax	csadm Version
Output	**CryptoServer Administration Utility 1.6.0** **csadm_lib (ADM section) 1.1.19** **csadm_lib (AUDIT section) 1.1.0** **csadm_lib (CSL section) 1.1.2** **csadm_lib (MMC section) 2.2.5** **csadm_lib (MTC section) 2.1.3** **csadm_lib (SLF section) 1.0.3** **csadm_lib (AUTH/SM section) 1.2.4** **csadm_lib (BL section) 1.1.0** **csadm_lib (BL3 section) 1.0.5** **csadm_lib (UTL section) 1.3.4** **csadm_lib (PKG section) 1.3.7** **csadm_lib (MBK section) 1.3.1** **csadm_lib (SMC section) 1.0.6** **csadm_lib (CLONE section) 1.0.2** **csxapi 1.6.13** **pp_api 1.2.2** **sl 1.1.0** **yacl 1.7.3** **yacl (VRSA section) 1.3.3** **yacl (HASH section) 1.2.0** **yacl (DES section) 1.0.2** **yacl (ECC section) 1.1.1** **yacl (AES section) 1.1.0**

4.3 Commands to Set-Up Parameters

Most commands require one or more parameters which have to be set up prior to the command execution (and will be evaluated during command execution).

The following table shows all available parameters:

Command	Parameter	Used e. g. by command(s)
`Dev=` `Device=`	address of CryptoServer or CryptoServer LAN (see 4.1.3)	nearly all
`MMCSignKey=`	key specifier of private key.	`MakeMTC`
`MMCPubKey=`	key specifier of public key.	`VerifyMTC`
`SignType=`	Hardware type specifier; needed to determine correct signature format: *'C86' for MMC/MTC creation for CryptoServer Se series* *'C57' for MMC/MTC creation for CryptoServer CSe series*	`MakeMTC`
`Password=`	password of encrypted key file	many commands, like e. g. `MakeMTC`, `LogonSign`, `AuthRSASign`, `ChangePassword, ...`
`NewPassword=`	new password of encrypted key file	`ChangePassword`, `GenKey`
`Pause=`	"1" or "0", to switch on or off	
`Key=`	key specifier for MBK key shares	`MBKGenerateKey`, `MBKImportKey`, `MBKCopyKey`
`OldKey=`	key specifier of old MBK	`MBKImportKey`
`FWEncKey=`	key specifier for firmware encryption key (public RSA key).	`MakeMTC`
`FWDecKey=`	key specifier for firmware decryption key (private RSA key).	`SaveKey`

`KeyType=`	Type of specified key (RSA, EC)	`GenKey`
		`ChangePassword`
		`SaveKey`
		`BackupKey`
		`CopyBackupCard`

If the environment variable CRYPTOSERVER is set, the 'Dev=' parameter can be skipped (see also 4.1.2).
Using the 'Dev=' parameter overrides the CRYPTOSERVER environment variable in any case for the specific command.

4.4 Commands to Prepare Firmware Modules

These commands are used for the load preparation of firmware modules, in particular the making, verification and removing of the *Module Transport Container* (MTC).

Furthermore commands are offered to create a firmware package file "*.mpkg" or to see its contents, or to retrieve the contained modules from a "*.mpkg" file.

More details about the concepts of MTCs and package files can also be found in chapter 3.9.

All firmware modules are packed into a signed data container before they are loaded into the CryptoServer:

Module Transport Container – MTC			
MTC Header	MMC Header	Binary (*.out, *.dll)	MMC Signature (*Module Signature Key*)

The Module Transport Container is signed with the private part of the Module Signature Key. This signature is mandatory and is intended to be created by the author of the firmware module. The CryptoServer - and anyone else - is able to check the authenticity of the firmware module's origin.

When a firmware module is loaded into the CryptoServer the operating system (SMOS) tries to verify the signature with the public part of the *Manufacturer's Module Signature Key,* which is stored in every CryptoServer.

The structure and signature of a firmware module container can be checked manually (outside the CryptoServer) using the command *VerifyMTC* (see chapter 4.4.3).

The following command group contains internal functions of CSADM. No connection to a CryptoServer will be established.

4.4.1 MakeMTC

This command builds the *Module Transport Container* file from a raw, binary firmware module (*.out, *.dll).

Firmware modules created by Utimaco are always signed with the private part of the Manufacturer's Module Signature Key. As the corresponding public key part is loaded into every CryptoServer, these firmware modules can be loaded without additional measures.

(not available in FIPS mode) Firmware modules created (programmed) by the customer have to be signed with an Alternative Module Signature Key, which is usually also created by the customer. The public part of the Alternative Module Signature Key has to be loaded into the CryptoServer in order to be able to load a self-signed firmware module into the CryptoServer.

Syntax	csadm SignType=<type>] MMCSignKey=<keyspec> …[Password=<password>] [FWEncKey=<keyspec>] …MakeMTC=<file>
Parameters	<keyspec> MMCSignKey: private part of the *Module Signature Key* FWEncKey: public part of firmware encryption key (not available in FIPS mode) *(see 4.1.4 for possible key specifiers)* <password> If an encrypted key file is used as *Firmware Encryption Key:* password of that encrypted key file *(see 4.1.4 for encrypted key files)*: for hidden password entry: string 'ask' (see 4.1.5; hidden password entry is strongly recommended); otherwise: password of key file <file> firmware module (*.out, *.dll, *.mmc) <type> hardware type (determines signature type): • 'c57': for CryptoServer CSe-Series • 'c86': for CryptoServer Se-Series
Example	csadm SignType=c86 MMCSignKey=c:\keys\my_mdl_sign.key …MakeMTC=c:\firmware\exmp.out
Output	**building MTC ... OK** on success, or error message

4.4.2 RemoveMTC

This command removes the header and signature of a given MTC container and restores the contained binary firmware module file (*.out or *.dll).

Syntax	csadm RemoveMTC=<file>
Parameter	<file> firmware module as MTC (*.mtc)
Example	csadm RemoveMTC=c:\firmware\exmp.mtc
Output	**removing MTC ... OK** on success, or error message
Note	• The unpacked file will be stored in the current directory. • CSADM is also creating a temporary file (*.mmc) in the current directory.

Removing the MTC container of a firmware module is only necessary if a firmware module should be re-signed with a customer-generated Module Signature Key and thus is of no use in FIPS mode.

4.4.3 VerifyMTC

This command verifies the signature of the *Module Transport Containers* MTC (*.mtc).

Syntax	csadm MMCPubKey=<keyspec> VerifyMTC=<file>
Parameter	<keyspec> MMCPubKey: public part of the *Module Signature Key* *(see 4.1.4 for possible key specifiers)* <file> firmware module as MTC (*.mtc)
Example	csadm MMCPubKey=:cs2:cyb:COM1 VerifyMTC=c:\firmware\exmp.mtc
Output	**Verifying MTC ...OK** **Removing MTC ...OK** **Verifying MMC ...OK**
Note	While verifying a MTC, CSADM is creating a temporary file (*.mmc) in the current directory.

Should multiple MTCs be verified in one step, the last part of the command (VerifyMTC=<file>) will have to be repeated for each firmware module

4.4.4 ModuleInfo

This command shows the module information of a firmware module. MTC, MMC or raw binary files (COFF) can be given as input file.

Syntax	csadm ModuleInfo=<file>
Parameter	<file> firmware module (e.g. *.mtc, *.mmc, *.out, or *.dll)
Example	csadm ModuleInfo=adm.mtc
Output	**Module** **'ADM'** **ID** **87** **Version** **3.0.4.4** **Name** **'Administration Module'** **Module type** **'CS2-COFF'** **MMC header vers.** **1.1.0.0** **Target CPU type** **'C64x'** **Architecture** **'C86'** **Signatures contained in file: MMC**
Note	• For firmware modules MTC with older versions (architecture = C64 or C62) also the owner and name of the MTC signing key (usually the *Default Administrator Key*) is displayed.

If multiple firmware modules shall be processed in one step, the last part of the command (ModuleInfo=<file>) will have to be repeated for each firmware module.

The meaning of the output information is as follows:

Field	***Description***
Module	Firmware module name
ID	Firmware module ID (FC)
Version	Firmware module version number
Name	Extended firmware module name
Module type	Type of the raw binary file that is contained in MTC/MMC: • 'CS2-COFF' if binary is compiled for CryptoServer hardware, • 'SDK' if binary is compiled for CryptoServer simulator If the input file is already a binary file, 'unknown' will be returned.

Field	***Description***
MMC header vers.	Version of MMC header (will be returned only for MTC or MMC file): • 1.1.0.0: current version (Note: SMOS version ≥ 2.1.0.0 is needed for interpretation of MMC header) • 1.0.0.0: older version (encrypted firmware modules not yet supported by MMC format)
Target CPU type	CPU type of target platform: • 'C64x' for CryptoServer Se series • 'C64+' for CryptoServer CSe series • 'SDK' for CryptoServer simulator
Architecture	Hardware type / Software architecture type: • 'C86' for CryptoServer Se series • 'C57' for CryptoServer CSe series

4.4.5 RenameToVersion

This command expands the filename of a firmware module by inserting the version number of the firmware module.

On loading onto the CryptoServer this added version number will be automatically removed.

Syntax	csadm RenameToVersion=<file>
Parameter	<file> firmware module (*.mtc, *.out, *.dll)
Example	csadm RenameToVersion=C:\modules\vdes.mtc
Output	none on success, or error message
Note	In the example above, the file `C:\modules\vdes.mtc` is renamed to e. g. `C:\modules\vdes_1.0.4.0.mtc`

4.4.6 Pack

This command creates a CryptoServer package file (archive "*.mpkg", see 3.10.2) from the content of the given directory. All files contained in the given directory will be added to the package file (which has the same name as the given directory, plus extension ".mpkg"). This package file can later be used to load a set of several firmware modules and files into the CryptoServer within one command (*LoadPkg* command, see 4.6.16).

Syntax	csadm Pack=<directory>
Parameter	<directory> Input directory containing CryptoServer firmware modules and files
Example	csadm Pack=firmware
Output	none on success, or error message
Note	• The resulting package file "firmware.mpkg" will be stored in the same directory as the given input directory 'firmware'. • Any existing file with the same name will be overwritten.

4.4.7 Unpack

This command extracts all files contained in a CryptoServer package file "*.mpkg" (see 3.10.2). The files will be extracted in a new directory which will have the same name as the given package file but without the extension ".mpkg".

Syntax	csadm Unpack=<package>
Parameter	<package> Input package file containing CryptoServer firmware modules and files (filename must have extension ".mpkg")
Example	csadm Unpack=firmware.mpkg
Output	none on success, or error message
Note	The resulting directory 'firmware' containing the extracted firmware modules and files will be created in the same directory where the given package file "firmware.mpkg" is located. If a directory of that name already exists, the extraction is denied.

4.4.8 ListPkg

This command lists the content of a CryptoServer package file (archive "*.mpkg", see 3.10.2), i. e. the filenames of all contained files. If firmware modules are contained in the package the module's version numbers and long names are also listed.

Syntax	csadm ListPkg=<package>
Parameter	<package> Input package file containing CryptoServer firmware modules and files (filename must have the extension ".mpkg")
Example	csadm ListPkg=firmware.mpkg
Output	Example output: **Archive firmware.mpkg contains:** **1. - adm_1.0.1.1.mtc (Administration Module version 1.0.1.1)** **2. - asn1_1.0.1.2.mtc (ASN1 Module version 1.0.1.2)** **3 - cmds_1.0.6.0.mtc (Command Scheduler version 1.0.6.0)** **4. - db_1.0.1.1.mtc (Database module version 1.0.1.1)** **5. - hash_1.0.1.0.mtc (Hash Module version 1.0.1.0)** **6. - lna_1.0.3.0.mtc (Long Number Arithmetic version 1.0.3.0)** **7. - smos_1.0.3.5.mtc (SMOS Operating System version 1.0.3.5)** **8. - util_1.0.5.0.mtc (Utility Module version 1.0.5.0)** **9. - vdes_1.0.4.0.mtc (DES Module version 1.0.4.0)** **10 - vrsa_1.0.4.0.mtc (RSA Module version 1.0.4.0)** **.**

4.4.9 VerifyPkg

This command verifies the signature of the *Module Transport Container* MTC (*.mtc) of each firmware module that is contained in the given package file (archive "*.mpkg", see 3.10.2).

This command can be performed without any access to a CryptoServer.

Syntax	csadm [MMCPubKey=<keyspec>] MTCPubKey=<keyspec> VerifyPkg=<package>
Parameters	<keyspec> MMCPubKey: public part of the *Module Signature Key* *(see 4.1.4 for possible key specifiers)* <package> input package file containing CryptoServer firmware modules and files (filename must have extension ".mpkg")
Example	csadm MMCPubKey=:cs2:cyb:COM1 VerifyPkg=c:\firmware\fw-1.0.0.mpkg
Output	Information about the firmware module signature (verification results, see example below).

Example:

csadm MMCPubKey=:cs2:cyb:COM1 MTCPubKey=c:\keys\init_pub.key
VerifyPkg=c:\firmware\fw-1.0.0.mpkg

```
I: Verifying db_1.1.0.0_c64.mtc          MTC OK - MMC OK
I: Verifying vdes_1.0.1.1_c64.mtc        MTC OK - MMC OK
I: Verifying aes_1.0.3.0_c64.mtc         MTC OK - MMC OK
I: Verifying asn1_1.0.3.3_c64.mtc        MTC OK - MMC OK
I: Verifying smos_2.0.0.6_c64.mtc        MTC OK - MMC OK
I: Verifying hash_1.0.5.0_c64.mtc        MTC OK - MMC OK
I: Verifying eca_1.1.0.4_c64.mtc         MTC OK - MMC OK
I: Verifying ecdsa_1.0.1.0_c64.mtc       MTC OK - MMC OK
I: Verifying dsa_1.0.0.0_c64.mtc         MTC OK - MMC OK[11]
I: Verifying util_2.1.0.0_c64.mtc        MTC OK - MMC OK
I: Verifying adm_2.0.0.3_c64.mtc         MTC OK - MMC OK
I: Verifying cmds_1.1.1.0_c64.mtc        MTC OK - MMC OK
I: Verifying pkcs11_1.0.5.1_c64.mtc      MTC OK - MMC OK
I: Verifying vrsa_1.0.6.0_c64.mtc        MTC OK - MMC OK
I: Verifying lna_1.0.4.3_c64.mtc         MTC OK - MMC OK
Package c:\firmware\fw-1.0.0.mpkg successfully verified
```

[11] CSe only

4.5 CryptoServer Driver Commands

The commands in this section are directed to the PCI/PCIe driver of the CryptoServer card. If the addressed CryptoServer is part of a CryptoServer LAN, these commands are executed remotely via TCP.

The CryptoServer may be in any mode (see 3.1) to execute the driver commands. No authentication towards the CryptoServer is required.

If the addressed CryptoServer is part of a CryptoServer LAN it might be necessary to authenticate the driver commands Reset, Restart and ResetToBL towards the CryptoServer LAN with the root password of the LAN box.

This depends on the configuration of the LAN box: if the 'AuthReset' variable is set in the configuration file, password authentication of these commands is mandatory (see [CSLANAdminManual] section 5.2.5 *The Configuration File csxlan.conf*). This password protection can be useful e. g. to avoid a denial-of-service attack.

4.5.1 Reset

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

If the CryptoServer is not in FIPS mode *(i. e. the CryptoServer is in personalization mode, e. g. during the initial setup phase), after executing Reset it takes up to 15 seconds until the boot loader window is timed out and the operating system is restarted again (see chapter 2.2.4).*

In this case it is therefore recommended NOT to use the Reset *command but instead ...*

- *to use Restart (see 4.5.3) to restart the CryptoServer as fast as possible (3 – 5 sec) or*
- *to use ResetToBL (see 4.5.2) to reset the CryptoServer and to get into boot loader mode.*

If the CryptoServer is in FIPS mode *this boot loader time window is skipped, the Reset and* Restart *commands give the same results.*

Syntax	csadm [Dev=<device>] Reset[=<password>]
Parameters	<device> device specifier (see 4.1.3) <password> root password of CryptoServer LAN (see below): • for hidden password entry: string ‘ask’ (see 4.1.5; hidden password entry is strongly recommended!); • otherwise: root password This parameter is only relevant if the CryptoServer LAN is used, and if the command is password protected (see below).
Output	none on success, or error message
Note	In case that the CryptoServer LAN is used and the driver commands *Reset*, *Restart* and *ResetToBL* are protected by a root password (i. e. depending on the settings in the LAN box configuration file), the ‘password’ parameter is mandatory. See page 73 above.

4.5.2 ResetToBL

If the CryptoServer is in FIPS-mode, it is not possible to set the CryptoServer into boot loader mode. The command will instead perform a reset and restart the CryptoServer into operational mode. It is recommended to use the Restart command instead. the ResetToBL command is not available!
If the CryptoServer is not in FIPS-mode but in personalization mode, this command can be used for error diagnostics

This command will reset the CryptoServer and get it into *boot loader mode* (see 2.2.5):
First a reset of the CryptoServer is performed in the same way as using the *Reset* command. Immediately after the *Reset* command a dummy command is sent to the CryptoServer (which will only be accepted if the CryptoServer is not in FIPS-mode). The boot loader now remains active and will not start the operating system SMOS. Thus the CryptoServer is in boot loader mode now.

If the CryptoServer is in boot loader mode, the operating system (SMOS) and all other firmware modules can be started using the StartOS command (see 4.6.3). This way the CryptoServer can be put into operational mode again.
The RecoverOS command (see 4.6.4) on the other hand starts the recovery copies of the firmware modules, which were loaded into the CryptoServer during production.
A CryptoServer in FIPS mode can only be in boot loader mode if it is in FIPS error state.

Syntax	csadm [Dev=<device>] ResetToBL[=<password>]
Parameters	<device> device specifier (see 4.1.3) <password> root password of CryptoServer LAN (see below): • for hidden password entry: string ‘ask’ (see 4.1.5; hidden password entry is strongly recommended!); • otherwise: root password This parameter is only relevant if the CryptoServer LAN is used, and if the command is password protected (see below).
Output	none on success, or error message
Note	• In boot loader mode any command sent to the CryptoServer is responded by the boot loader. • The boot loader will remain active until the CryptoServer is reset again or the *StartOS*-command is sent to the CryptoServer. • In case that the CryptoServer LAN is used and the driver commands *Reset, Restart* and *ResetToBL* are protected by a root password (i. e. depending on the settings in the LAN box configuration file), the ‘password’ parameter is mandatory. See page 73 above.

4.5.3 Restart

This command will reset/restart the CryptoServer and get it into *operational mode* (see 2.2.5):

In a first step a reset of the CryptoServer is performed in the same way as using the *Reset* command. In addition to the *Reset* command the *StartOS* command (which will be ignored in FIPS-mode) is sent to the CryptoServer immediately.

Using the Restart command is the fastest way to restart the operating system. If it is successful, the CryptoServer is in operational mode afterwards.
If the Restart command is performed as last step of the personalization process, and in particular if all firmware modules that are mandatory for FIPS mode are loaded, the CryptoServer will remain in FIPS mode afterwards.

Syntax	csadm [Dev=<device>] Restart[=<password>]
Parameters	<device> device specifier (see 4.1.3) <password> root password of CryptoServer LAN (see below): • for hidden password entry: string ‘ask’ (see 4.1.5; hidden password entry is strongly recommended!); • otherwise: root password This parameter is only relevant if the CryptoServer LAN is used, and if the command is password protected (see below).
Output	none on success, or error message
Note	• As a general rule, the CryptoServer has to be restarted after the loading (or replacement) of a new firmware module: The firmware module only becomes active after the CryptoServer having been restarted. • In case that the CryptoServer LAN is used and the driver commands Reset, Restart and ResetToBL are protected by a root password (i. e. depending on the settings in the LAN box configuration file), the ‘password’ parameter is mandatory. See page 73 above.

4.5.4 GetInfo

This command retrieves information from the PCIe driver of the CryptoServer. It shows the driver version, PCIe slot number, interrupt number, and the state of the error correction.

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 requests 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 4.1.3)
Output	**vers 3.0.1** **slot 0000:02:00.0** **model 2** **state idle** **spm 00008000** **count 0 0** **crc 0 0** **irq f 1 f f** **txlen 00000000 00000000** **rxlen 00000000 80000000** **fwver 1.0.0.3** **...**
Note	The command is used for diagnostic purposes in error case!

4.5.5 SetTimeout

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

Syntax	csadm [Dev=<device>] SetTimeout=<timeout>
Parameters	<device> device specifier (see 4.1.3) <timeout> new timeout to be set in milliseconds
Output	old timeout: 600000 ms new timeout: 10000 ms
Note	The new timeout is valid for the current connection. Next time csadm is executed, it will use the default timeout setting.

4.6 Commands for Administration

In this chapter the commands for the administration of the CryptoServer device are described, like status requests, firmware management and audit management.

The CryptoServer has to be either in *Operational Mode* or in *Maintenance Mode*.

Not all commands can be executed in FIPS mode, and in FIPS error state administrative commands that are security-relevant can not be performed at all. See chapter 4.1.7 for details.

Some of the commands have to be authenticated (e. g. *LoadFile*), some do not (e. g. *ListFiles*).

Command Authentication is based on a sophisticated user management, which allows to create various users with different permissions, authentication mechanisms and other properties. For a detailed description of the possible authentication mechanisms and the user concept see chapter 3.2.2. The commands for user management can be found in chapter 4.6.18.

Although command authentication is implemented in a very generic way, the way a command has to be authenticated depends on the user, i. e. on his/her special authentication mechanism. Therefore (in this and the following sections) the description of the command syntax for commands in operational mode which have to be authenticated *does not specify each possibility of authentication*. Instead a placeholder (<Authentication>) is inserted, and the command example then shows one possibility of authenticating the command.

If in the syntax of one of the commands described in the following chapters the placeholder <Authentication> is found, it has to be replaced by one or more authentication commands according to the required authentication state and depending on the specific user's (permissions and) authentication mechanis.m The exact syntax of the various authentication commands (LogonSign, LogonPass, AuthRSASign, AuthHMACPwd,) can be found in chapter 4.10.

4.6.1 GetState

This command returns the status of the CryptoServer (see 3.1), its temperature, alarm state, hardware information and set-up information.

The GetState command is responded by the CryptoServer in any mode (boot loader mode as well as operational mode, 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.
See also 7.5 for further interpretation of status information in FIPS mode.

Please prepare the output of GetState in case of a support request.

Syntax	csadm [Dev=<device>] GetState
Parameter	<device> device specifier (see 4.1.3)
Authentication	none
Output	**example for CryptoServer in FIPS-mode and normal operational state:**
	mode = Operational Mode **state = INITIALIZED (0x00140004)** **FIPS mode = ON** **temp = 47.5 [C]** **alarm = OFF** **bl_ver = 3.00.3.0 (Model: Se-Series)** **UID = 28000011 0c82f401** **adm1 = 53653530 20202020 43533838 38303232 | Se50 CS888022** **adm2 = 43757374 6f6d6572 20585900 00000000 | Customer XY** **adm3 = 496e6974 2d446576 2d312d4b 65790000 | Installed**
	example for CryptoServer not in FIPS-mode but Maintenance Mode, alarm is on:
	mode = Maintenance Mode **state = INITIALIZED (0x000a7f84)** **temp = 44.6 [C]** **alarm = ON** **sens = 027f** **- Alarm has occurred** **- external Erase is executed** **bl_ver = 3.00.3.0** **UID = 28000011 0c82f401** **adm1 = 5554494d 41434f20 43533431 30303036 | UTIMACO CS410006** **adm2 = 43757374 6f6d6572 20585900 00000000 | Customer XY** **adm3 = 496e6974 2d446576 2d312d4b 65790000 | Installed**

The displayed fields have the following meaning:

field	***description***
state	state of the CryptoServer (should be *initialized* - see also 3.1): • Manufactured — only relevant during production process • Initialized — firmware modules and all system keys (*Production Key, Module Signature Key, Default Administrator Key*) are correctly loaded. • Defect — CryptoServer defect – please contact manufacturer/Utimaco
mode	• Operational — regular set of firmware modules (*.msc) is started. • Maintenance — back-up set of firmware modules (*.sys) is started (e.g. in alarm state) • Boot Loader — boot loader is running (and operating system and regular set of firmware modules have not been started yet)
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 restrictions applied (Se only)	Se only: If the FIPS Mode Control Module is loaded but the other loaded firmware modules are not identical with the FIPS validated firmware modules (see chapter 10) the CryptoServer is NOT in validated FIPS mode, but most of the restrictions implemented for FIPS mode are nevertheless applied. In particular all restrictions for the usage of non-approved cryptographic algorithms and key lengths are applied. This mode is indicated as “FIPS restrictions applied”.
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. 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 3.7): < -13°C — sensory triggers a temperature alarm => all user data on the CryptoServer will be cleared! -13°C - 62°C — normal temperature range > 62°C — a new entry is put into the audit log file is made and the CryptoServer is set into power down mode (see 2.2.5). Commands will not be executed any longer and the CryptoServer needs a Restart-command to be reset into normal operation after it has cooled down again. > 66°C — sensory triggers a temperature alarm => all sensitive data on the CryptoServer will be cleared!

field	***description***
alarm	either ON or OFF, if ON (see also 3.5) the following reasons are possible and will be shown in case of an alarm state: • Power is too low (i.e. empty battery) • Power is too high • Temperature too high (> 66°C) • Temperature too low (< -13 °C) • External Erase is executed (manually by a short-circuit of the corresponding pins on the PCIe-card) • Invalid / Corrupted Master Key • Communication to Sensory Controller failed • Outer Foil Damaged (CSe only) • Inner Foil Damaged (CSe only) In addition it is shown if the alarm reason is still present or if it has been removed in the meantime (e. g. empty battery has been replaced).
bl_ver	version of the boot loader
UID	Unique ID of the CryptoServer (as a hardware property)
adm1	Unique serial number of the CryptoServer PCIe card which is assigned during the production process by Utimaco IS GmbH.
adm2	Field is assigned during the production process by manufacturer. This field may be empty.
adm3	Field is assigned during the production process by manufacturer. This field may be empty.

4.6.2 GetBattState

This command can be used to show the state of the two batteries, the carrier battery and the external battery. The state “low” indicates that the battery should be renewed as soon as possible to avoid a power low alarm.

Syntax	csadm [Dev=<device>] GetBattState
Parameter	<device> device specifier (see 4.1.3)
Authentication	none
Output	**Carrier Battery: ok (3.055 V)** **External Battery: absence**
Note	• The external battery is only relevant for a Cryptoserver LAN device. For a CryptoServer PCI/PCIe card the external battery is usually not present, and thus the state ‘absence’ for the external battery is no reason to worry.

4.6.3 StartOS

The regular set of firmware modules (*.msc) is started. These firmware modules are maintained (loaded, updated or deleted) by the customer.

See 2.2.4.2 for more detailed information about this process.

On start-up of the CryptoServer (power-on, or after the Reset command) the StartOS command is automatically executed by the boot loader if it doesn't receive any command within 3 seconds (see chapter 2.2.4.1)

Syntax	csadm [Dev=<device>] StartOS
Parameter	<device> device specifier (see 4.1.3)
Authentication	none
Output	none on success, or error message
Note	• If no operating system SMOS is present (SMOS has not been loaded onto CryptoServer before), an error message will be displayed. • SMOS checks the integrity of every firmware module, a firmware module will only be started after successful verification. • If an error occurs during the start-up of a firmware module, the corresponding error message will be added to the boot log file (see GetBootLog, section 4.6.11).

This command is intended to be performed in boot loader mode. After it is successfully performed, the CryptoServer is in Operational Mode (see 2.2.5). If the CryptoServer is NOT in BL mode the StartOS command will be ignored (no output).

4.6.4 RecoverOS

The recovery set of firmware modules (*.sys) is started. These set of base firmware modules has been loaded during the production process and can't be modified or deleted by the customer.

See 2.2.4.5 for more detailed information.

The recovery set of the firmware modules have to be started explicitly by the user, using this RecoverOS command.
This may be necessary if it is not longer possible to start the regular set of firmware modules (*.msc files), for example because one or more indispensable modules (e. g. SMOS, CMDS, ADM, UTIL) have been deleted by mistake, or a defect firmware module has been loaded.

Syntax	csadm [Dev=<device>] RecoverOS
Parameter	<device> device specifier (see 4.1.3)
Authentication	none
Output	none on success or error message
Note	• If no operating system SMOS is present (SMOS has not been loaded onto CryptoServer before), an error message will be displayed. • SMOS checks the integrity of every firmware module, a firmware module will only be started after successful verification. • If an error occurs during the start-up of a firmware module, the corresponding error message will be added to the boot log file (see GetBootLog, section 4.6.11).

This command is intended to be performed in boot loader mode. After it is successfully performed, the CryptoServer is in Maintenance Mode (see 2.2.5).If the CryptoServer is NOT in BL mode the StartOS command will be ignored (no output).

4.6.5 ListFiles

This command lists all files stored on the CryptoServer. The following directories are available on the different storage devices:

Device	Directory Name	Size	Description
FLASH	\FLASH	32 MBytes	Every firmware module has read and write access to the working directory. The regular set of firmware modules and any other kind of application data (databases, configuration or log files, ...) is stored here.
NVRAM (non volatile RAM)	\NVRAM	512 kBytes	Every firmware module has read and write access to this directory. Often changing data, which should be stored non-volatile, is stored here.

Don't use the \FLASH directory to store often changing data (e. g. counter). This will result in a great wear and tear of the flash-component and therefore in a decreased lifetime of the CryptoServer.
Use the \NVRAM directory instead to store this data non-volatile!

Syntax	csadm [Dev=<device>] ListFiles[=<mode>]
Parameters	<device> device specifier (see 4.1.3) <mode> <none> - show all non-hidden files SYS - only hidden files in the \FLASH directory are listed NVRAM - only files in the \NVRAM directory are listed
Examples	• csadm ListFiles • csadm ListFiles=NVRAM • csadm ListFiles=SYS
Authentication	none

Output

```
file name           size     module: ID    type version  name
------------------------------------------------------------------------------
FLASH\CXIKEY.db     202      -
FLASH\VMBK1.db      97       -
FLASH\adm.msc       68204    ADM     0x087 C64  3.0.11.4 Administration Module
FLASH\aes.msc       72380    AES     0x08b C64  1.3.1.1  AES Module
FLASH\asn1.msc      15436    ASN1    0x091 C64  1.0.3.3  Asn1 Module
FLASH\audit_00.log  7667     -
FLASH\cmds.msc      99644    CMDS    0x083 C64  3.1.1.2  Command Scheduler
FLASH\cxi.msc       241452   CXI     0x068 C64  2.1.3.2  Crypto. eXtended Interf
FLASH\db.msc        34700    DB      0x088 C64  1.1.2.5  Database module
FLASH\eca.msc       108396   ECA     0x08f C64  1.1.3.0  Elliptic Curve Arith.
FLASH\ecdsa.msc     33068    ECDSA   0x09c C64  1.1.2.0  ECDSA Module
FLASH\fips140.msc   27388    FIPS140 0x001 C64  3.0.1.0  FIPS140 Module
FLASH\hash.msc      40236    HASH    0x089 C64  1.0.8.0  Hash Module
FLASH\hce.msc       20044    HCE     0x00a C64  1.0.1.1  Hardware Crypto Engine
FLASH\lna.msc       79564    LNA     0x08e C64  1.2.0.1  Long Number Arithmetic
FLASH\mbk.msc       63260    MBK     0x096 C64  2.2.4.0  Master Backup Key Mod
FLASH\se50.slf      200      -
FLASH\smos.msc      132156   SMOS    0x000 C64  3.1.2.1  SMOS Operating System
FLASH\user.db       183      -
FLASH\util.msc      17612    UTIL    0x086 C64  3.0.2.0  Utility Module
FLASH\vdes.msc      42284    VDES    0x081 C64  1.0.5.0  DES Module
FLASH\vrsa.msc      96204    VRSA    0x084 C64  1.1.2.0  RSA Module

                22 files   1200377 bytes, 31143936 bytes available
```

Note

- If no mode parameter is given, all non-hidden files are listed.
- 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)
  - CPU target type (should be ‘C64’ for CryptoServer Se series, ‘C64+’ for CSe series, and ‘SDK’ for CryptoServer simulator)
  - version number
  - long name

The CryptoServer’s file system is case sensitive!

4.6.6 LoadFile

This command loads a file (firmware module or other file) onto the working directory (\FLASH) or the non-volatile directory (\NVRAM) of the CryptoServer.

If the file is a firmware module ('*.mtc') the MMC signature of the firmware module will be checked. If it can't be verified successfully the loading of the module will be rejected.

In order not to violate security restrictions the following files or file types are generally not allowed to be loaded:

- prod*.key
- init*.key
- mdlsig*.key
- alarm.sens
- auth_fails.max
- fips.mode
- *.mt_
- *.sl_
- *.s__
- *.sc_
- *.c__
- *.msc
- *.sys
- *.emc
- *.db
- *.log

If the given file already exists, it will be replaced.
A loaded/replaced firmware module does not become active until the CryptoServer has been restarted (see Restart command, chapter 4.5.3).

Syntax	csadm [Dev=<device>] <*Authentication*> LoadFile=<file>[,<dir>]
Parameter	<device> device specifier (see 4.1.3) <file> filename (eventually with path) <dir> CryptoServer directory where the file should be stored (FLASH or NVRAM). If omitted, the FLASH directory is used as default.
Examples	csadm LogonSign=ADMIN,d:\keys\init_prv.key … … LoadFile=exmp_dbg_1.0.2.3.mtc csadm LogonPass=paul,swordfish LoadFile=usage.cnt,NVRAM

Authentication	The command must be authenticated with level 2 in user group 6. If the audit configuration file 'audit.cfg' is loaded, the command must be authenticated with level 2 in user groups 6 and 7.
Output	none on success or error message
Note	• If the filename of a firmware module contains an underscore character ('_') the rest of the name up to the file extension will be stripped before loading (e. g. exmp_dbg.mtc -> exmp.mtc, adm_1.0.0.1.mtc -> adm.mtc). • The length of the file name (without directory) may not exceed 15 characters (here only characters count that will not be stripped, i.e. characters before any underscore character '_'). • If no path is given with the <file> parameter, the file will be taken from the current directory. If the file to be loaded is stored on another directory (e. g. on an USB stick), the respective path must be given. • The CryptoServer's file system is case sensitive! • Filenames of firmware modules (with extension '.mtc') are converted to lower case letter before being loaded onto the CryptoServer. • If several files should be loaded in one step, the last part of the command ('LoadFile=...') has to be repeated for every wanted file.

Firmare modules should not be loaded using function 'LoadFile' directly but only using service 'LoadPackage' (section 4.6.16) with option 'ForceClear' as explained in section 7.3.

If the FIPS Mode Control Module FIPS140 is loaded but the other loaded firmware modules are not identical with the FIPS certified firmware modules (as listed in chapter 10)

- ***the CryptoServer Se leaves validated FIPS mode.***
- ***the CryptoServer CSe goes into FIPS error state. This state can only be left by executing an External Erase which deletes all sensitive data!***
  If the FIPS Mode Control Module FIPS140 is loaded in non-FIPS-mode FIPS mode is entered during next restart and all sensitive data (including all users in the user database) are erased from the CryptoServer!

4.6.7 DeleteFile

With this command a file or a group of files can be deleted from the CryptoServer's working directory (\FLASH) or non-volatile directory (\NVRAM).

The following files or file types are generally not allowed to be deleted due to security restrictions:

- prod*.key
- init*.key
- mdlsig*.key
- alarm.sens
- auth_fails.max
- fips.mode
- *.sys
- *.log
- *.scf

This command can also be used to delete firmware modules (*.msc files), but the backup set of the basis firmware modules (system firmware modules '*.sys') can not be deleted!
Even if already deleted from the working directory, a firmware module is still running (Se: in SD-RAM memory; CSe: DDR2 RAM) and will only become inactive after the CryptoServer has been restarted.

In FIPS mode firmare modules should not be deleted using function 'DeleteFile' directly but only using service 'LoadPackage' (section 4.6.16) with option 'ForceClear' as explained in section 7.3.

If the FIPS Mode Control Module FIPS140 is loaded but the other loaded firmware modules are not identical with the FIPS certified firmware modules (as listed in chapter 10) the CryptoServer goes into FIPS error state. This state can only be left by executing an External Erase which deletes all sensitive data.

If the FIPS Mode Control Module FIPS140 is deleted FIPS mode is left during next restart and all sensitive data (including all users in the user database) are erased from the CryptoServer!

Syntax	csadm [Dev=<device>] <*Authentication*> DeleteFile=[<dir>\]<file>
Parameters	<device> device specifier (see 4.1.3) <file> filename <dir> CryptoServer directory (FLASH or NVRAM). If omitted, the working directory (\FLASH) is used as default.

Examples	• csadm LogonPass=paul,swordfish DeleteFile=exmp.msc • csadm LogonSign=sm,:usa:cm8:COM2 … … DeleteFile=NVRAM\usage.cnt
Authentication	The command must be authenticated with level 2 in user group 6 (see chapters 3.2.2 and 4.10).
Output	none on success or error message
Note	The CryptoServer file system is case sensitive!

4.6.8 GetTime

This command returns the CryptoServer's system time.

Syntax	csadm [Dev=<device>] GetTime
Parameter	<device> device specifier (see 4.1.3)
Authentication	none
Output	**Date: 06.11.2008 Time: 15:35:49.400**
Note	• The system clock of the CryptoServer has a resolution of 1/1000 second (one millisecond). • Date and time are given in the time zone of the host PC's system time. • The necessary transformation from the CryptoServer's internal system time (which runs in Greenwich Mean Time, GMT) to the host PC's system time zone is done automatically by CSADM.

4.6.9 SetTime

This command sets the CryptoServer's system clock.

Syntax	csadm [Dev=<device>] <*Authentication*> SetTime=<time>
Parameters	<device> device specifier (see 4.1.3) <time> • use given digit string ('YYYYMMDDHHMMSS'). The time will be converted to GMT time zone automatically. • GMT – use the system time of the host PC to set the CryptoServer's time (recommended). The time will be converted to GMT time zone automatically.
Examples	• csadm LogonSign=ADMIN,:cm8:usa:COM1 SetTime=GMT • csadm LogonPass=paul,swordfish SetTime=20020602115500
Authentication	The command must be authenticated with level 2 in user group 6 (see chapters 3.2.2 and 4.10).
Output	none on success or error message
Note	• The CryptoServer's system clock is assumed to run in Greenwich Mean Time (GMT). Whenever SetTime is performed, the necessary transformation from the host PC's system time zone to GMT is done automatically by CSADM. • Any changing of the clock is taken down on the audit log file. The new entry contains the old time as well as the new time value. The GetAuditLog command (see chapter 4.6.12) can be used in any mode to view this file.

The command is not sent to the CryptoServer until authentication is done. If this requires a lot of time (e. g. insertion of a smartcard and PIN input) there is a gap between the given time and the actual time. If the CryptoServer's time has to be set very precisely you can enter a future time and perform the last step (e. g. pressing 'OK' after PIN input) when this time is reached

4.6.10 ListModulesActive

This function returns a list with information about all firmware modules which are currently running, giving their module ID, abbreviated name, CPU type of target ('C64' for CryptoServer Se series, 'C64+' for CSe series, 'SDK' for CryptoServer simulator), version number and their respective module initialization level.

If a module cannot be started or fully initialized, the GetBootLog command (see next chapter 4.6.11) provides more detailed information about the reason.

Syntax	csadm [Dev=<device>] ListModulesActive
Parameters	<device> device specifier (see 4.1.3)
Authentication	none
Output	ID name type version initialization level --- 0 SMOS C64 3.1.2.1 INIT_OK 1 FIPS140 C64 3.0.1.0 INIT_OK a HCE C64 1.0.1.1 INIT_OK 68 CXI C64 2.1.3.2 INIT_OK 81 VDES C64 1.0.5.0 INIT_OK 83 CMDS C64 3.1.1.2 INIT_OK 84 VRSA C64 1.1.2.0 INIT_OK 86 UTIL C64 3.0.2.0 INIT_OK 87 ADM C64 3.0.11.4 INIT_OK 88 DB C64 1.1.2.5 INIT_OK 89 HASH C64 1.0.8.0 INIT_OK 8b AES C64 1.3.1.1 INIT_OK 8e LNA C64 1.2.0.1 INIT_OK 8f ECA C64 1.1.3.0 INIT_OK 91 ASN1 C64 1.0.3.3 INIT_OK 96 MBK C64 2.2.4.0 INIT_OK 9c ECDSA C64 1.1.2.0 INIT_OK
Note	• For the meaning of the possible module initialization levels see below. • If for a firmware module no entry is found, the module is not loaded. • After loading or replacing a firmware module, the CryptoServer has to be restarted (see Restart command chapter 4.5.3) before the firmware module becomes active.

During the CryptoServer's boot process 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 (files '*.msc', or in maintenance mode files '*.sys') and started by SMOS 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 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 (CSe: DDR2 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 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 SC (Smartcard) module requires the PP (PIN pad) 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). Possible calls to services from other modules are done successfully.
MDL_INIT_FAILED	**Module initialization failed**. Services provided by this module are not available.
MDL_INACTIVE	Module is loaded but cannot supply services, e.g. because of not-installed hardware components. This initialization level is currently only used by firmware module HCE (Se only; unused on a CryptoServer CSe).
MDL_SUSPENDED	Module was shutdown and cannot supply services any more.

4.6.11 GetBootLog

GetBootLog retrieves a log file which contains log messages made by the operating system and other firmware modules (or by the boot loader if the command is called in boot loader mode) during the 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.

Syntax	csadm [Dev=<device>] GetBootLog
Parameter	<device> device specifier (see 4.1.3)
Authentication	none
Output	**SMOS Ver. 3.0.1.0 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 MBK can't find module DB (00000000)** **FATAL: module 0x96 (MBK) initialization failed (err = b096fe01)**

4.6.12 GetAuditLog

The function “GetAuditLog” shows the contents of all audit logfiles. The function can be called in any mode and needs no authentication.

Syntax	csadm [Dev=<device>] GetAuditLog
Parameters	<device> device specifier (see 4.1.3)
Authentication	none
Output	**__.__.__ __:__:__ new ALARM detected 2: power fail** **__.__.__ __:__:__ [ADMIN] Reset Alarm [0]** **27.07.10 09:27:37 [ADMIN] Set Time from 0Z [0]** **27.07.10 09:29:01 [ADMIN] Load File 'FLASH\smos.msc' [0]** **27.07.10 09:29:01 [ADMIN] Load File 'FLASH\util.msc' [0]** **27.07.10 09:29:02 [ADMIN] Load File 'FLASH\cmds.msc' [0]** **27.07.10 09:29:02 [ADMIN] Load File 'FLASH\adm.msc' [0]**

4.6.13 ClearAuditLog

This function erases the contents of audit logfiles. For a continuous auditing the newest logfile(s) can be kept. See also 3.4.

Syntax	csadm [Dev=<device>] <*Authentication*> ClearAuditLog[=<n>]
Parameters	<device> device specifier (see 4.1.3) <n> number of (the newest) logfiles not to be deleted
Example	csadm LogonSign=ADMIN,:cm8:usa:COM1 ClearAuditLog=2
Authentication	The command must be authenticated with level 2 in user group 7 or with level 2 in user group 6 (see chapters 3.2.2 and 4.10).
Output	none on success or error message

4.6.14 MemInfo

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

The desired directory (‘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 4.1.3) <dir> directory on the CryptoServer (‘FLASH’ or ‘NVRAM’)
Example	csadm MemInfo=FLASH
Authentication	none
Output	**FLASH** **max_size = 33292288** **used_size = 1832960** **free_size = 30753280** **available_size = 31459328**
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

4.6.15 Test

With this command a communication test with the CryptoServer is executed. It sends series of random 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 4.1.3) <datalength> length [bytes] of the used pattern. <loopcount> number of execution cycles.
Examples	csadm Test=256,10000
Authentication	none
Output	**data type : random** **data length (min) : 256** **data length (max) : 256** **increment : 1** **loop count : 10000** **len count** **256 10000** **execution time : 329 ms** **data throughput : 7781155 bytes/s** **transaction time : 0.032900 ms**
Note	• 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 ;-)

4.6.16 LoadPkg

This command gives the user a comfortable and easy-to-use possibility to load or update a set of several firmware modules and/or files into the CryptoServer in only one step. It can replace a series of succeeding CSADM commands (see example below).

The *LoadPkg* command loads the contents of the given package file (archive "*.mpkg", see also 3.10.2) into the CryptoServer. Depending on the given command line flags the load procedure can also perform a *Clear* if needed.

Syntax	csadm [Dev=<device>] <*Authentication*> … … LoadPkg=<package>[,<flags>[,<rootpassword>]]	
Parameters	<device>	device specifier (see 4.1.3)
	<package>	input package file containing CryptoServer firmware modules and files (filename must have extension ".mpkg")
	<flags>	See below. Flags can be added (+) if this makes sense.
	<rootpassword>	root password of CryptoServer LAN (see note below): • for hidden password entry: string 'ask' (see 4.1.5; hidden password entry is strongly recommended!); • otherwise: root password This parameter is only relevant if the CryptoServer LAN is used, and if the driver commands *Reset*, *Restart* and *ResetToBL* are password protected (see below).
Example	csadm LogonSign=ADMIN,:cs2:cyb:COM1 … … LoadPkg=firmware.mpkg,NoClear+NoDelete	
Authentication	This command has to be authenticated with level 2 in user group 6 (see chapters 3.2.2 and 4.10).	
Output	information about all actually performed steps and changes in status, mode or firmware module versions (see examples below)	
Note	In case that the CryptoServer LAN is used and the driver commands *Reset, Restart* and *ResetToBL* are protected by a root password (i. e. depending on the settings in the LAN box configuration file), the 'password' parameter is mandatory. See section 4.5.	

Modules that are stored inside the CryptoServer but not contained in the given package file will remain on the CryptoServer. To delete them, the option 'ForceClear' has to be used.

If a firmware update makes the CryptoServer enter or leave FIPS mode all sensitive data (including all users in the user database) are erased from the CryptoServer during next restart!

flag	meaning
`NoClear`	CryptoServer will not be cleared prior to package loading (default).
`AllowClear`	CryptoServer will only be cleared prior to package loading if necessary. This option is not available in FIPS mode.
`ForceClear`	CryptoServer will be cleared prior to package loading in any case. All sensitive data inside the CryptoServer will be deleted before the contents of the package file are loaded. This option is not available in FIPS mode.
`SwapComCreate`	Creates the file FLASH\swap.com in the CryptoServer
`SwapComDelete`	Deletes the file FLASH\swap.com in the CryptoServer (if available)
`SwapComDetect`	If the addressed CryptoServer is a CryptoServer LAN, the file FLASH\swap.com will be deleted. For all other CryptoServer this file will be created.

Example 1:

The firmware which is currently stored in the CryptoServer shall be completely replaced by the firmware (and other files) contained in a CryptoServer package file “cxi_1.3.1.3.mpkg”. In this case a *Clear* should be performed as part of the *LoadPkg* command, i. e. the ‘ForceClear’-flag has to be set. During command execution, information about all currently performed steps and the CryptoServer’s state and mode will be displayed:

```
csadm LogonSign=ADMIN,:cs2:cyb:COM3 LoadPKG=d:\temp\cxi_1.3.1.3.mpkg,ForceClear
I: Reading package...
I: Perform authentication and create session
I: Going to delete all files/modules inside the CryptoServer (perform Clear)
I: Load file smos_3.0.1.0_c86.mtc
I: Load file util_3.0.0.1_c86.mtc
I: Load file cmds_3.0.0.2_c86.mtc
I: Load file adm_3.0.0.2_c86.mtc
I: Load file db_1.1.2.2_c86.mtc
I: Load file pp_1.2.3.0_c86.mtc
I: Load file sc_1.2.0.0_c86.mtc
I: Load file asn1_1.0.3.3_c86.mtc
I: Load file hash_1.0.6.0_c86.mtc
I: Load file aes_1.0.4.1_c86.mtc
I: Load file vdes_1.0.3.0_c86.mtc
I: Load file lna_1.1.0.0_c86.mtc
I: Load file vrsa_1.1.0.1_c86.mtc
```

```
I: Load file eca_1.1.1.0_c86.mtc
I: Load file ecdsa_1.0.1.0_c86.mtc
I: Load file mbk_2.1.2.3_c86.mtc
I: Load file hce_1.0.0.0_c86.mtc
I: Load file cxi_1.3.1.3_c86.mtc
I: Restarting CryptoServer
Package d:\temp\cxi_1.3.1.3.mpkg successfully loaded
```

Example 2:

In the following example only some firmware modules will be upgraded (AES, CMDS), modules with higher version on the CryptoServer will not be touched and others (that have not been yet available on the CryptoServer) will be loaded.

```
csadm LogonSign=ADMIN,init_dev_prv.key LoadPkg=fw1.mpkg
I: Reading package...
I: Perform authentication and create session
I: Retrieving file list from CryptoServer
I: Load file exmp_1.0.1.0.mtc
I: Load file pkcs11_1.0.6.1.mtc
I: Load file cmds_3.0.0.2_c86.mtc
I: Load file aes_1.0.5.0_c86.mtc
I: Restarting CryptoServer
Package fw1.mpkg successfully loaded
```

4.6.17 CheckPkg

This command compares the firmware modules contained in a given package file (archive "*.mpkg", see 3.10.2) against the firmware currently loaded on a CryptoServer. It announces if the package file contents differ from the loaded firmware and gives detailed information about which modules had to be loaded or deleted (in case the package file is loaded) and about differences in firmware module versions. If a *Clear* command would be unavoidable in case the given package file is loaded, this would be announced, too.

Information about differences concerning files other than firmware is not given.

Syntax	csadm [Dev=<device>] CheckPkg=<package>[,<password>]
Parameters	<device> device specifier (see 4.1.3) <package> input package file containing CryptoServer firmware modules and files (filename must have extension ".mpkg") <password> root password of CryptoServer LAN (see below): • for hidden password entry: string 'ask' (see 4.1.5; hidden password entry is strongly recommended!); • otherwise: root password This parameter is only relevant if the CryptoServer LAN is used, and if the driver commands *Reset, Restart* and *ResetToBL* are password protected (see below).
Example	csadm CheckPkg=firmware.mpkg
Authentication	none
Output	information about differences between loaded firmware and firmware contained in given package file (see example below)
Note	In case that a CryptoServer LAN is used and the driver commands *Reset*, *Restart* and *ResetToBL* are protected by a root password (i. e. depending on the settings in the LAN box configuration file), the 'password' parameter is mandatory. See section 4.5.

```
Example:
I: Reading package...
I: Retrieving file list from CryptoServer
Package d:\temp\cxi_1.3.1.3.mpkg successfully checked
```

4.6.18 Clear

The Clear command erases all sensitive data from the CryptoServer. Only the data of users with public key authentication mechanisms may remain in the user database, depending on the used variant of the command. For a detailed description in which case this command should be used, please also refer to chapter 3.6.

The following table gives an overview about the required authentication and the action taken depending on the given command flag:

Clear =	**required authentication**	**action**
INIT	user with administrative rights	The CryptoServer's internal *Master Key* and other data on the CryptoServer are erased. The following files are not deleted: • the boot loader code and the system firmware modules ('*.sys') • the alarm state file (alarm.sens) • the audit log file (audit.log) • the audit configuration file (audit.cfg) Only users with a password authentication mechanism are deleted from the user database, users with a public key remain stored in the user database.
DEFAULT	no authentication, but an external erase must have been initiated just before	This command restores the factory default settings and resets the CryptoServer into delivery state. The CryptoServer's internal *Master Key* and every other data on the CryptoServer is erased, except for the recovery set of firmware modules (*.sys). The user database will be deleted and recreated with the default user (ADMIN) and the corresponding *Default Administrator Key*.

Syntax	csadm [Dev=<device>] [<*Authentication*>] Clear[=<flag>]
Parameter	<device> device specifier (see 4.1.3) <flag> command flag, can be INIT or DEFAULT. If omitted, the INIT version of the command is used by default. For details and required authentication see table above.
Examples	csadm LogonSign=ADMIN,c:\keys\init_prv.key Clear=INIT csadm Clear=DEFAULT
Authentication	The command with flag=INIT must be authenticated with level 2 in user group 6 (see chapters 3.2.2 and 4.10).

Output	none on success or error message

With the command Clear=DEFAULT the CryptoServer is reset to factory defaults.
By this means the CryptoServer can be restored in case the user has locked out himself from administration (e.g. lost smartcard, forgotten password, …).
Before the command can be executed the user has to prove physical presence by triggering an external erase (which can only be performed locally, i.e. directly at the CryptoServer device).

None of the variants of the Clear command is available in FIPS mode.

See 7.7 for how to clear the CryptoServer if it is in FIPS mode. By clearing the CryptoServer it will leave FIPS mode.

4.6.19 ResetAlarm

This command manually resets the *alarm state* if the physical reason for the alarm is no longer present.

The ResetAlarm command is only available (and makes sense) directly after an alarm. In particular it is not available in FIPS mode because the CryptoServer automatically leaves FIPS mode after the occurrence of an alarm.

If an alarm occurs on the CryptoServer, the internal *Master Key* and other sensitive data will be erased, FIPS mode is left (by deleting the FIPS Mode Control Module) and an entry into the audit log file will be made (see chapter 3.5 *Alarm*). The occurrence of the alarm will be stored as a special *alarm state*. Even if the physical reason for an alarm has been removed (e. g. a low battery was exchanged), the alarm state is still active and has to be manually reset by an authenticated user with *ResetAlarm*. With *ResetAlarm* also a new internal *Master Key* will be generated.

In alarm state the CryptoServer is running in *maintenance mode* (i. e. only the back-up set of the base firmware modules is running) and there is no secret or private key left. Since the firmware module DB for database management is blocked, no further keys can be loaded in alarm state. Most CryptoServer functionality is blocked.

On reset of a pending alarm a new *Master Key* will be generated and the CryptoServer will be restarted. Afterwards the CryptoServer is running in *operational mode* again.

If the reason for the alarm is still pending (e. g. temperature still too high), any attempt to reset the alarm state will cause the automatic execution of a reset of the CryptoServer (in the same way the Reset command does).

See 8.2 *Alarm Treatment* for more details about what to do in case of an alarm.

Syntax	csadm [Dev=<device>] [<Authentication>] ResetAlarm
Parameters	<device> device specifier (see 4.1.3)
Example	csadm LogonSign=ADMIN,:cs2:cyb_usb0 ResetAlarm
Authentication	The command must be authenticated with level 2 in user group 6 or with level 2 in user group 7 (see chapters 3.2.2 and 4.10).
Output	none on success or error message
Note	• If the (physical) alarm reason is still present, an error message will be returned and the CryptoServer does not leave the alarm state. • Any new alarm is taken down on the 'audit.log' file, together with the date and time when the alarm happened. The *GetAuditLog* command (see chapter 4.6.12) can be used to view this file.

4.6.20 GetAuditConfig

This command shows the configuration setting for the audit functionality of the CryptoServer. See chapter 3.4 for details on audit configuration.

Syntax	`csadm [Dev=<device>] GetAuditConfig`
Parameter	<device> device address (see chapter 4.1.3)
Example	`csadm GetAuditConfig`
Authentication	none

Output	`Audit log configuration parameters:`
	`Number of log files: 5`
	`Rotate logfiles: yes`
	`Max. filesize 200000`
	`Events: 0x00000007 (Bits 1:2:3)`

The audit log configuration parameters have the following meaning (see also chapter 3.4):

Audit configuration parameter	***Meaning***
`Number of log files`	Maximal number n of audit log files (2 ≤ n ≤ 10 allowed)
`Rotate logfiles`	*yes* *Audit log files will be written rotatingly.* *no* *Audit log files will NOT be written rotatingly.*
`Max. filesize`	Maximum length of one audit log file in bytes
`Events`	Class mask of the configured audit message classes: An event with audit message class no. n will be audited if and only if the bit n is set. In the example above (class mask 0x00000007, i. e. bits 1, 2 and 3 set), all events from the message classes firmware/file management, user management and time management will be audited. See chapter 3.4 for a list of all audit message class numbers.

4.7 User Management

The CryptoServer provides a sophisticated user management in order to maintain different users with different authentication mechanisms and permissions.

In FIPS mode, security relevant commands have to be authenticated by a user who has assumed the ***Administrator, SecurityOfficer, Key Manager, User*** or ***Cryptographic User*** role (see chapter 3.2.1 for the details). In dependency on the command the user therefore has to be fitted out with a specific permission.

User's credentials are stored in the user database 'user.db', which is hosted on the CryptoServer.

The respective commands to create, change or delete users are described in this chapter. Within this user management, the following properties can be assigned to each user:

Property	***Description***
Name	user name, up to 255 printable characters (must not contain a '~')
Permission	A user's permission consists of 8 different figures (enumerated downward 7 .. 0) with values between 0 and 3. 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). All commands for global CryptoServer administration, configuration and user management (like *LoadFile*, *AddUser*, *SetTime* ..., see this chapter and 4.6) can only be performed after an *Administrator's* authentication. 2) Users who shall be allowed to assume the ***Security Officer*** role must have the **user permission '00000200'** (or higher). All external functions realized by the firmware module CXI which offer Key Group specific configuration management (like *getKeyProperty, setKeyProperty,…*) and user management (*AddGroupUser*, *DelGroupUser*) or which allow to set the TRUSTED property of a wrapping key (SetKeyProperty) can only be performed after a *Security Officer's* authentication. 3) Users who shall be allowed to assume the ***User*** role must have the **user permission '00000002'** (or higher). All external functions realized by the firmware module CXI which offer cryptographic services (like encryption/decryption, MAC calculation, hashing, ...) can only be performed after a *User's* authentication. 4) Users who shall be allowed to assume the ***Key Manager*** role must have the **user permission** which is defined by the configuration value 'CXI_PROP_CFG_AUTH_KEYM' (or higher; see chapter). The default setting is '00000002' which means that the *User* role and the *Key Manager* role is the same (also called '*Cryptographic Users*') and both *Users* and *Key Managers* are allowed to perform all cryptographic and key management services. All external functions realized by the firmware module CXI which offer key management functions (like key generation, key import/export, ...) can only be performed after a *Key Manager's* authentication. 5) Additionally it is possible to create users which can assume both *Administrator* and *User* role, i. e. users with user permission

	'22000002' (or higher). 7) Users who are allowed to assume the *User* role AND the *Key Manager* role are called ***'Cryptographic Users'***. By default the required *Key Manager* permission and the required *User* permission are the same ('00000002') which means that the *User* role, the *Key Manager* role and the *Cryptographic User* role is identical. When the *Key Manager* Permission mask is configured to '00000020' the Cryptographic User role is split into two different sub-roles *Key Manager* and *User*. Cryptographic Users must then have the permission mask '00000022' to be able to perform all cryptographic AND key management services.
Mechanism	• RSA signature • HMAC password See 3.2.2 for more details about the authentication mechanisms.
Flags	no_login: user has to authenticate each (sensitive) command separately; static login is not allowed in FIPS mode
	sma: user may open a secure messaging session if he/she authenticates the command (see 4.11).

Attributes	A string containing a list of assignments with syntax:
	`<AttrName>=<AttrValue>,<AttrName>=<AttrValue>,…`
	Example:
	`P11Slot=005,P11User=SO`
	The meaning of the Attributes depends on the application loaded into the CryptoServer.

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

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

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

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

4.7.1 ListUser

This command lists all existing users from the ‘user.db’ database.

Syntax	csadm [Dev=<device>] ListUser
Parameter	<device> device specifier (see 4.1.3)
Authentication	none
Output	**Name Permission Mechanism Flags Attributes** **ADMIN 22000000 RSA sign no_login + sma** **sm 00000022 HMAC passwd no _login + sma A[CXI_GROUP=test]** **paul 22000022 HMAC passwd no _login + sma** **test 21000011 HMAC passwd no _login + sma A[P11Slot=007]**
Note	• The initial user ‘ADMIN’ will be displayed even if the database ‘user.db’ is not yet present. • For a description of the properties (name, permission, mechanism, flags, attributes) see the previous pages or 3.2.

4.7.2 AddUserRSASign

With this command a new user is added to the user database using 'RSA-Signature' as authentication mechanism. Therefore the user needs a RSA key pair either on a smartcard or as – optionally encrypted - key file.

During the execution of this command, the public part of the RSA key is read from the smartcard or key file and stored in the CryptoServer's user database.

The CryptoServer can check later the authenticity of commands by verifying the RSA command signature: this RSA signature is calculated by the host over a random value (a challenge value retrieved from the CryptoServer in a prior step) and the command data block with the private part of the user's key (PKCS#1 signature format). This signature will then be transmitted to the CryptoServer who will verify it with the help of the RSA key's public part which is stored in the user database.

Syntax	csadm [Dev=<device>] <*Authentication*> … ... AddUserRSASign=<user>,<permission>,<flag1>[+<flag2>],<keyspec>
Parameters	<device> device specifier (see 4.1.3) <user> user name <permission> user's permissions (see 3.2.2): 8 digits 'XXXXXXXX', each representing a permission level in a specific user group. A list of attributes, put into curly braces { }, can be appended to the permission digits, see table with user properties in 4.7 above and example below. <flag1> 'no_login' / 'allow_login' (‚no_login' mandatory in FIPS mode) <flag2> 'sm' / 'sma' (‚sma' mandatory in FIPS mode) <keyspec> public part of the user's RSA key (see 4.1.4)
Examples	csadm LogonSign=ADMIN,:cs2:cyb:COM1 … … AddUserRSASign=Paul,01000000,no_login+sma,:cs2:cyb:COM1 csadm LogonSign=ADMIN,:cs2:cyb:COM1 … … AddUserRSASign=Eve,010{P11Slot=007},no_login+sma,key1.key
Authentication	The command must be authenticated with level 2 in user group 7 (see chapters 3.2.2 and 4.10).
Output	none on success or error message
Note	If the authentication of this command requires a smartcard and the source of the new user's public key is a smartcard, too, watch the PIN pad's display and 1. insert the smartcard containing the new user's RSA key first, 2. insert the smartcard for command authentication after that.

4.7.3 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 smartcard 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>] <Authentication> … ..... ChangeUserRSASign=<user>,<keyspec>
Parameter	<device> device specifier (see 4.1.3) <user> existing user name <keyspec> public part of the user’s new RSA key (see 4.1.4)
Example	csadm LogonSign=paul,:cs2:cyb:COM1 ChangeUserRSASign=paul,:cs2:cyb:COM1
Authentication	The command must be authenticated by the existing user according to his/her authentication mechanism ‘RSA Signature’ (see chapters 3.2.2 and 4.10).
Output	none on success or error message
Note	If the authentication of this command requires a smartcard and the source of the user’s new public key is a smartcard too, watch the PIN pad’s display and... 1. insert the smartcard containing the user’s new RSA key at first, 2. insert the smartcard for command authentication (old key) after that.

4.7.4 AddUserHMACPwd

With this command a new user is added to the user database using the 'HMAC Password' authentication mechanism.

The entered password is stored in the CryptoServer's user database.

The CryptoServer can check later the command authentication via HMAC value which is going to be added to each command: This hash value has been calculated by the host based on a random value (a challenge value retrieved from the CryptoServer in a prior step) and the command data block using the password as the HMAC key. The CryptoServer can recalculate and check the hash value with the help of the password stored in its user database.

Compared with any clear password authentication, this mechanism has the following advantages:

The password will not be submitted in clear and thus cannot be scanned.

Because of the random value a 'Playback'-Attack of the command becomes impossible.

Syntax	csadm [Dev=<device>] <*Authentication*> … ... AddUserHMACPwd=<user>,<permission>,<flag1>[+<flag2>],<password>	
Parameters	<device>	device specifier (see 4.1.3)
	<user>	user name
	<permission>	user's permissions (see 3.2.2): 8 digits 'XXXXXXXX', each of them representing a permission level in a specific user group. A list of attributes, put into curly braces { }, can be appended to the permission digits, see table with user properties in 4.7 above and example below.
	<flag1>	'no_login' / 'allow_login' (‚no_login' mandatory in FIPS mode)
	<flag2>	'sm' / 'sma' (‚sma' mandatory in FIPS mode)
	<password>	for hidden password entry: string 'ask' (see 4.1.5 and below; hidden password entry is strongly recommended); otherwise: new user's password
Example	csadm LogonSign=ADMIN,:cs2:cyb:COM1 … … AddUserHMACPwd=paul,11000021,no_login+sma,swordfish csadm LogonSign=ADMIN,:cs2:cyb:COM1 … … AddUserHMACPwd=paul,11000021,no_login+sma,ask csadm LogonSign=ADMIN,:cs2:cyb:COM1 … … AddUserHMACPwd=paul,020{P11Slot=004},no_login+sma,enigma	
Authentication	The command must be authenticated with level 2 in user group 7 (see chapters 3.2.2 and 4.10).	

Output	none on success or error message

The usage of hidden password entry is strongly recommended!

If hidden password entry is used, the CSADM will ask for the new user's password separately (i. e. a request 'Enter Passphrase:' follows in the next command line) and hide the entrance on the monitor by the display of default characters.

4.7.5 ChangeUserHMACPwd

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

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

Syntax	csadm [Dev=<device>] <*Authentication*> … ... ChangeUserHMACPwd=<user>,<password>
Parameter	<device> device specifier (see 4.1.3) <user> existing user name <password> for hidden password entry: string 'ask' (see 4.1.5 and below; hidden password entry is strongly recommended); otherwise: user's new password
Examples	csadm LogonPass=paul,swordfish … … ChangeUserHMACPwd=paul,sesame csadm LogonPass=paul,ask ChangeUserHMACPwd=paul,ask
Authentication	The command must be authenticated by the existing user according to his/her authentication mechanism (see chapters 3.2.2 and 4.10).
Output	none on success or error message
Note	The user has to be already present in the user database

The usage of hidden password entry is strongly recommended!

If hidden password entry is used, the CSADM will ask for the user's new password separately (i. e. a request 'Enter New Passphrase:' follows in one of the next command lines) and hide the entrance on the monitor by the display of default characters.

4.7.6 DeleteUser

This function deletes a user from the user database.

In order to prevent the CryptoServer from getting non-administrable, the sum of the permissions of the remaining users who use a signature-based authentication mechanism has to be at least level 2 in user group 7 and level 1 in group 6. If this is not the case, the function will refuse execution but return with an error code.

Syntax	csadm [Dev=<device>] <*Authentication*> DeleteUser=<user>
Parameter	<device> device specifier (see 4.1.3) <user> existing user
Example	csadm LogonSign=ADMIN,:cs2:cyb:COM1 DeleteUser=paul
Authentication	Command must be authenticated with level 2 in user group 7 (see chapters 3.2.2 and 4.10).
Output	none on success or error message

4.7.7 BackupUser

This command backs up all users that are currently set up on the CryptoServer into a file. Each user entry in the created backup file is encrypted with the CryptoServer's Master Backup Key (MBK) and therefore can be handled without additional security measures.

The user 'ADMIN' will not be backed up.

Syntax	csadm [Dev=<device>] <*Authentication*> BackupUser=<file>[,<flags>]
Parameter	<device> device specifier (see 4.1.3) <file> path and file name of the backup file to be created <flags> optional flags: • 'overwrite': backup file will be overwritten if already existing
Example	csadm LogonSign=ADMIN,:cs2:cyb:COM1 ... ... BackupUser=d:\temp\cs123456-20051212.ubk,overwrite
Authentication	The command must be authenticated with level 2 in user group 7 (see chapters 3.2.2 and 4.10).
Output	none on success or error message
Note	If the overwrite flag is not given and the backup file already exists the backup will not be performed and the existing backup file will not be overwritten.

4.7.8 RestoreUser

This command restores the users on the CryptoServer from a backup file (see command *BackupUser*).

The same Master Backup Key (MBK) which has been used to create the backup file has to be stored on the CryptoServer.

Syntax	csadm [Dev=<device>] <*Authentication*> … ... RestoreUser=<file>[,<flags>]
Parameter	<device> device specifier (see 4.1.3) <file> path and file name of the backup file <flags> optional flags: • 'overwrite': if a user with one of the given user names already exists on the CryptoServer it will be overwritten
Example	csadm LogonSign=ADMIN,:cs2:cyb:COM1 ... ... RestoreUser=d:\temp\cs123456-20051212.ubk,overwrite
Authentication	The command must be authenticated with level 2 in user group 7 (see chapters 3.2.2 and 4.10).
Output	none on success or error message
Note	If the overwrite flag is not given and one of the users (identified by its user name) already exists on the CryptoServer the command fails and returns an error code.

4.7.9 SetMaxAuthFails

This command defines the unallowed number (n) of consecutive failed authentication attempts for each user (see section 3.2.2.5).

If this number is reached the user is blocked. A blocked user is not able to authenticate towards the CryptoServer any more.

A blocked user can be unblocked by an authenticated user administrator (permission level 2 in user group 7; 0x2000 0000)

Syntax	`csadm [Dev=<device>] <Authentication> SetMaxAuthFails=<n>`
Parameter	<device> device specifier (see 4.1.3) <n> Defines the unallowed number of consecutive failed authentication attempts for each user. Allowed range is from 0 to 255. A value of 0 means that an unlimited number of failed authentication attempts is

	allowed, no user is blocked. Default value is 0.
Example	*csadm Dev=192.168.1.2 LogonSign=ADMIN,:cs2:cyb:USB0 SetMaxAuthFails=3* `csadm LogonPass=Anita,ask SetMaxAuthFails=4`
Authentication	The command must be authenticated with level 2 in user group 7 (see chapters 3.2.2 and 4.10).
Output	none on success or error message

The configured value is not changed in case of an alarm. It is reset to zero by execution of a clear command.

4.7.10 GetMaxAuthFails

This command displays the unallowed number (n) of consecutive failed authentication attempts for each user (see section 3.2.2.5).

Syntax	`csadm [Dev=<device>] GetMaxAuthFails`
Parameter	<device> device specifier (see 4.1.3)
Example	`csadm Dev=192.168.1.2 GetMaxAuthFails`
Authentication	none
Output	`n` (0…255): the unallowed number (n) of consecutive failed authentication attempts for each user. If this number is reached the user is blocked and unable to authenticate towards the CryptoServer any more.

4.8 User Key Management

In this chapter CSADM commands for generation, administration and backup of user's authentication keys (or tests keys) are described. The generated keys may be stored either on a smartcard or in a key file.

The following command group contains internal functions of CSADM. No connection to a CryptoServer will be established.

4.8.1 GenKey

This command generates a key file '`*.key`' which contains a RSA or ECDSA key pair of given size (for key files see 4.1.4). This can be used e. g. to generate a user key.

It can be chosen if the generated key file is in plaintext or encrypted (see 4.1.4 for encrypted key files).

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 [NewPassword=<password>] … … KeyType=RSA GenKey=<file>[,<bitsize>[,<keyname>]] csadm [NewPassword=<password>] … … KeyType=EC GenKey=<file>,<curve>[,<keyname>]
Parameters	<password> password to protect encrypted key file: • for hidden password entry: string 'ask' (see 4.1.5; hidden password entry is strongly recommended); • otherwise: new password of key file <file> name of the key file (with path, filename extension '*.key') <bitsize> bit length of the generated RSA key (512 <= bitsize <= 8192) (default value: 1024 bit) <curve> name of the elliptic curve to be used for ECDSA key generation (see section 9) <keyname> key name
Examples	• csadm GenKey=C:\my_keys\testkey1.key,2048 • csadm KeyType=EC NewPassword=ask ... …GenKey=C:\my_keys\testkey2.key,secp256k1,key2
Output	none on success or error message

Note	• If the 'NewPassword' parameter is given, the command will generate an encrypted key file, protected by the given password (see 4.1.4 for encrypted key files). If the parameter is omitted, the command will generate a plaintext key file. • As default a RSA key will be generated. If an ECDSA key shall be generated, the parameter 'KeyType=EC' has to be given. • For the RSA key length the size 1024 bits will be taken as default value. • The parameter 'keyname' can be used to assign a name to the RSA or ECDSA key. If the parameter is not given, the key name will be left empty.

The usage of hidden password entry is strongly recommended!

If hidden password entry is used, the CSADM will ask for the password separately (i. e. a request 'Enter Passphrase:' follows in the next command line) and hide the entrance on the monitor by the display of default characters.

4.8.2 SaveKey

This command reads a key from one storage medium and stores it to another storage medium. The key may either be the public or private part of a key pair or one half of a backup key set.

The following table shows the possible combinations of source and target mediums:

Key Type	***Source Medium***	***Target Medium***
Public	key file	key file
	smartcard	key file
Private	key file	key file or smartcard
Backup	key file	key file or smartcard
	smartcard	key file or smartcard

It is neither possible to transfer the private key part out of a smartcard to any other medium, nor to transfer only a public key part to a smartcard.

This command is executed locally without any 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 PubKey=<source> SaveKey=<target> csadm PrvKey=<source> SaveKey=<target> csadm BckKey=<source> SaveKey=<target>
Parameter	<source> filename or key specifier (see 4.1.4) of user's key to be copied <target> filename or key specifier of the target key
Examples	csadm PubKey=:cs2:cyb:COM1 SaveKey=C:\keys\pubkey1.key csadm BckKey=:cs2:cyb:COM1 SaveKey=:cs2:cyb:COM1 csadm PubKey=C:\keys\prvkey1.key SaveKey=C:\keys\pubkey1.key csadm PrvKey=C:\keys\prvkey1.key#ask SaveKey=:cs2:cyb:COM1
Output	none on success, or error message
Note	• If smartcards are used: A PIN pad including smartcard reader has to be connected to a serial line of that computer where the CSADM tool is running. Watch the PIN pad's display for instructions on further command processing.

4.8.3 BackupKey

This command reads a key from a key file, splits it into two 'XOR' parts and saves the parts on two smartcards for back-up matters.

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 PrvKey=<keyfile> BackupKey=<keyspec>
Parameter	<keyfile> file where the key is stored (*.key, see 4.1.4) <keyspec> key specifier of the back-up smartcards (see 4.1.4)
Examples	csadm PrvKey=C:\my_keys\prvkey1.key BackupKey=:cs2:acr:COM1 csadm PrvKey=C:\my_keys\myprvkey.key#mypassword BackupKey=:cs2:acr:COM1
Output	none on success, or error message
Note	A PIN pad including smartcard reader has to be connected to a serial line of that computer where the CSADM tool is running. Watch the PIN pad's display for instructions on further command processing.

4.8.4 CopyBackupCard

This command copies one XOR-half of a key-backup (backup of user's authentication key) from one smartcard to another.

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 CopyBackupCard=<keyspec>
Parameter	<keyspec> key specifier of the backup-smartcards (see 4.1.4), source and target
Examples	csadm CopyBackupCard=:cs2:cyb:COM1
Output	none on success, or error message
Note	A PIN pad including smartcard reader has to be connected to the computer where the CSADM tool is running. Watch the PIN pad's display for instructions on further command processing.

4.8.5 GetCardInfo

The *GetCardInfo* command reads information about keys eventually stored on a smartcard: The info records of the user's authentication key (e.g. *Default Administrator Key*) and back-up key will be scanned and output.

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 GetCardInfo=<keyspec>
Parameter	<keyspec> key specifier of the smartcard (see 4.1.4)
Examples	csadm GetCardInfo=:cs2:cyb:COM1
Output	Example: **RSA-Key: Utimaco IS GmbH / Init-Dev-1-Key** **Key-Backup: (no key)**
Note	A PIN pad including smartcard reader has to be connected to a the computer where the CSADM tool is running. Watch the PIN pad's display for instructions on further command processing.

4.8.6 ChangePassword

This command changes the password of an encrypted key file ('`*.key`', see 4.1.4 for encrypted key files).

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 Password=<password1> NewPassword=<password> ChangePassword=<keyfile>
Parameters	<password1> for hidden password entry: string 'ask' (see 4.1.5 and below; hidden password entry is strongly recommended); otherwise: old password of key file <password2> for hidden password entry: string 'ask' (see 4.1.5 and below; hidden password entry is strongly recommended); otherwise: new password of key file <keyfile> encrypted key file ('*.key')
Examples	• csadm Password=swordfish NewPassword=shark ChangePassword=C:\my_keys\mykey1.key • csadm Password=ask NewPassword=ask ChangePassword=C:\my_keys\privatekey1.key
Output	none on success or error message
Note	The command can also be used to convert a plaintext key file into an encrypted key file and vice versa: • The command can be used to encrypt a key file that has been a plaintext key file before. In this case the parameter 'Password=' has to be omitted and only the 'NewPassword=' parameter has to be given. • The command can also be used to decrypt a key file, i. e. to turn an encrypted key file into a plaintext key file. In this case the parameter 'NewPassword=' has to be omitted and only the 'Password=' parameter has to be given.

The usage of hidden password entry is strongly recommended!

If hidden password entry is used, the CSADM will ask for the key file's old and new password separately (i. e. a request 'Enter Passphrase:' respectively 'Enter New Passphrase:' follows in the next command line) and hide the entrance on the monitor by the display of default characters.

4.8.7 ChangePIN

This command changes the PIN of a given smartcard. A PIN pad including smartcard 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 smartcard type, reader type and serial port (see 4.1.4)
Example	csadm ChangePIN=:cs2:cyb: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.

4.9 Management of Master Backup Keys

In order to provide backup functionality (see 3.9), CryptoServer is able to store up to four Master Backup Keys (in slot 0..3) to be used by various applications. Each Master Backup Key (MBK) can either be a DES key (16 or 24 bytes length) or an AES key (16, 24 or 32 bytes). In FIPS mode only an AES MBK can be used.

An MBK can be generated on the CryptoServer and — split into key parts (key shares) — externally stored on two or more smartcards. The key parts of a MBK can be imported into the CryptoServer either from smartcards or they can be manually keyed in at the smartcard reader (PIN pad). If a new key should be imported into a key slot which already contains an MBK, the old MBK has to be imported too in order to be verified. The CryptoServer compares the existing key with the key to be verified and overwrites the existing key with the new key only if they match.

In addition to the local management of Master Backup Keys (which is not available in FIPS mode), where the PIN pad has to be directly connected to the CryptoServer, the MBK can also be managed remotely without having direct physical access to the CryptoServer (which perhaps is located in a data center). To protect the key parts during transmission from/to the CryptoServer, they are encrypted with the session key (DES or AES) that was exchanged on establishment of the secure messaging session with the CryptoServer. A secure messaging session is therefore necessary for remote MBK management.

This chapter describes how Master Backup Keys (MBK) can be managed remotely.

In order to unify the MBK storage all Utimaco applications (e.g. PKCS#11, CXI) use the following convention:

Slot 0	DES key
Slot 1	Intermediate key (e.g. while key is copied)
Slot 2	not used
Slot 3	AES key (mandatory in FIPS mode)

4.9.1 MBKListKeys

This command lists all MBKs currently stored on a CryptoServer.

Syntax	csadm <*Authentication*> <Session> MBKListKeys
Authentication	In FIPS mode the command must be authenticated with level 2 in user group 6 (see chapters 3.2.2 and 4.10).

Output	`slot name len algo type k generation date key check value`
	`----------`
	`0 MyDES_01 16 DES SHARE 3 10/09/22 13:08:45 148EF5C582FD87C7`
	`1 MyAES_01 32 AES SHARE 3 10/09/22 13:08:41 491F19C95BBDDA02`
	`3 <NoName> 32 AES XOR 2 07/08/06 10:35:50 106B5E4E84031BDE`

Note	The CryptoServer is able to store up to four MBKs (in slots 0..3): • 'slot' gives the slot number • 'name' gives the key name (or <NoName> if no key name is given) • 'len' gives the MBK key size in bytes • 'algo' gives the key algorithm • 'type' indicates if the MBK was exported in two XOR halves ('XOR') or in more than two key shares ('SHARE') • 'k' gives the number of shares that is needed to recombine the MBK (with k=2 in case of export in two XOR halves) • 'generation date' is the date and time of generation of the MBK (up to seconds) • 'key check value' is a key check value (ISO-hash MDC2 over MBK) which can be used to identify the MBK If an older firmware package is used (firmware module MBK version < 2.2.1.0) the values of 'k', 'generation date' and 'key check value' are not returned.

4.9.2 MBKGenerateKey

This command generates an AES MBK on the CryptoServer, splits it into n key shares and transmits the encrypted key shares (encrypted with the session key of the current Secure Messaging session) to the host. The key shares will be decrypted on the host and stored either on n smartcards or in n key files.

Syntax	csadm <*Authentication*> <*Session*> Key=<keyspec> ... ... MBKGenerateKey=AES,<keylen>[,<n>,<k>,<keyname>]
Parameter	<keyspec> key specifier where the generated MBK shares should be stored (see 4.1.4): either smartcard specifier and record number (separated by comma), or list of filenames and passwords where required (e.g. file1#pwd1,file2#pwd2). <keylen> key size of MBK: 16, 24 or 32 bytes <n> number of key shares to be generated (default: 2) <k> number of key shares required to recombine key (default: 2) <keyname> Key name (up to 8 characters with ANSI / ISO-8859-1 encoding)
Examples	• csadm LogonSign=ADMIN,:cs2:cyb:COM3 Key=mbk1.key#swordfish,mbk2.key#sesame MBKGenerateKey=AES,32,2,2 • csadm LogonPass=Paul,ask Key=:cs2:cyb:COM3,15 MBKGenerateKey=AES,32,5,3
Authentication	The command must be authenticated with level 2 in user group 6 (see chapters 3.2.2 and 4.10). Additionally the command has to be executed within a secure messaging session.
Output	none on success or error message
Note	• If smartcards are used (recommended): A PIN pad including smartcard reader has to be connected to the computer where the CSADM tool is running. Watch the PIN pad's display for instructions on further command processing. • If key files are used: The usage of encrypted key files is strongly recommended! • The newly generated MBK is not stored on the CryptoServer. To store the MBK on the CryptoServer it has to be re-imported, see command *MBKImportKey*.

To protect the key parts during transmission from/to the CryptoServer, they are encrypted with the session key (DES or AES) that was exchanged on establishment of the secure messaging session with the CryptoServer.

A secure messaging session is therefore necessary for remote MBK management.

4.9.3 MBKImportKey

This command imports the key shares of a MBK either from two or more smartcards or from two or more key files.

Syntax	csadm *<Authentication> <Session>* ... ... Key=<keyspec> MBKImportKey=<key_no>
Parameter	<keyspec> key specifier where the new MBK should be loaded from (see 4.1.4 and above) <key_no> slot number on the CryptoServer where the key should be imported to (must be 3 in FIPS mode)
Examples	• csadm LogonSign=ADMIN,:cs2:cyb:COM3 Key=:cs2:cyb:COM3,1 MBKImportKey=3 • csadm LogonPass=Paul,ask Key=mbk1.key#swordfish,mbk2.key#sesame MBKImportKey=3
Authentication	The command must be authenticated with level 2 in user group 6 (see chapters 3.2.2 and 4.10). Additionally the command has to be executed within a secure messaging session.
Output	none on success or error message
Note	• If smartcards are used: A PIN pad including smartcard reader has to be connected to a the computer where the CSADM tool is running. Watch the PIN pad's display for instructions on further command processing. • If key files are used: The usage of encrypted key files, and of hidden password entry, is strongly recommended! • The old MBK has to be given if the desired key slot already contains an MBK. In this case the CryptoServer verifies if the existing MBK matches the old key before it overwrites it with the new key. If the key slot is still empty the new key is accepted without any verification.

To protect the key parts during transmission from/to the CryptoServer, they are encrypted with the session key (DES or AES) that was exchanged on establishment of the secure messaging session with the CryptoServer.

A secure messaging session is therefore necessary for remote MBK management.

4.9.4 MBKCopyKey

This command copies one key share of a MBK from one storage location to another.

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 Key=<keyspec_source> MBKCopyKey=<keyspec_target>	
Parameters	<keyspec_source>	key specifier where the key share should be loaded from (see 4.1.4): either smartcard specifier and record number (separated by comma), or list of filenames and passwords where required (e.g. file1#pwd1,file2#pwd2).
	<keyspec_target>	key specifier where key share should be stored (see 4.1.4 and above)
Examples	• csadm Key= mbk1.key#swordfish MBKCopyKey=:cs2:cyb:COM3,1 • csadm Key=:cs2:cyb:COM3,1 MBKCopyKey=mbk2.key#ask	
Output	none on success or error message	
Note	• With this command the storage location of a MBK can be migrated from smartcard to key file or vice versa. • If smartcards are used: A PIN pad including smartcard reader has to be connected to the computer where the CSADM tool is running. Watch the PIN pad's display for instructions on further command processing. • If key files are used: The usage of encrypted key files is strongly recommended! If encrypted key files are used, the usage of hidden password entry is strongly recommended!	

4.9.5 MBKCardInfo

This command lists the contents of a MBK smartcard. Here, every MBK key share is stored in a separate record.

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 MBKCardInfo=<keyspec>
Parameters	<keyspec> key specifier of the MBK smartcard (see 4.1.4)
Examples	csadm MBKCardInfo=:cs2:cyb:COM3
Output	REC LEN TYP ALG NAME TIMEGEN K ID HASH --- 01: 24 1 DES DESKEY 22.01.2007 13:52:01 00 00 C3C2A3F1C9E83A6D 02: 16 0 DES <NoName> 13.04.2007 13:41:58 00 00 6A7D994A662B4D22 03: 32 3 AES Test 13.04.2007 16:17:01 03 02 397620CEB7C61DBE
Note	• A PIN pad including smartcard reader has to be connected to the computer where the CSADM tool is running. • An MBK smartcard may contain up to 16 records / MBK parts. For each stored key share the following information is given: • REC: record number • LEN: key length (in bytes), • TYP: share type (only relevant if 0 or 1: first or second XOR-half; otherwise: key share), • ALG: MBK algorithm, • NAME: name of MBK (or <NoName> if no key name is given) • TIMEGEN: date and time of generation • K: number of shares that are necessary to recombine MBK (only relevant if TYP is not 0 or 1) • ID: ID of this share • HASH: check value over MBK (usually: first 8 bytes of ISO-hash MDC2 over MBK; if ID= 0 or 1: first respectively second 8 bytes of ISO-hash MDC2 over MBK)

4.9.6 MBKCardCopy

This command copies the whole content of a MBK smartcard to another.

Syntax	csadm MBKCardCopy=<keyspec>
Parameters	<keyspec> key specifier of the smartcard (see 4.1.4)
Examples	csadm MBKCardCopy=:cs2:cyb:COM1
Output	none on success or error message
Note	If records on the target smartcard already exist they will be overwritten without additional query.

4.9.7 MBKPINChange

This command changes the PIN on a MBK smartcard.

Syntax	csadm MBKPINChange=<keyspec>
Parameters	<keyspec> key specifier of the smartcard (see 4.1.4)
Examples	csadm MBKPINChange=:cs2:cyb:USB0
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.

4.10 Command Authentication

CryptoServer provides a variety of command authentication mechanisms. See chapter 3.2.2 for a detailed explanation of the authentication concept.

Most pre-defined security-relevant commands require the authentication level 2 in a specific user group. Since each user usually has only permission level 1 in one or more user groups (apart from the pre-defined user ADMIN who has permission level 2 in user group 7 and 6, i. e. for the user management and for the administrative commands), this means that two users of the specific user group are necessary to authenticate the command. For this reason it can (and will) be said that the specific command requires authentication according to the **2-persons-rule**, i. e. authentication by two independent users is necessary if the command shall be executed.

If a command requires an authentication according to the 2-persons-rule, both users have to apply their authentication prior to command execution:
csadm <Authentication1> <Authentication2> <command>
The two users may even use different authentication mechanisms.
Example:
csadm LogonSign=roberta,:cs2:cyb:COM1 LogonPass=paul,swordfish DeleteFile=xxx.mtc

It is recommended to use exclusively the authentication commands LogonSign or LogonPass, because they do not only perform command authentication but provide also a Secure Messaging session for the protection of the confidentiality of the command data in one step. See 4.10.1 and 4.10.2.

In the following chapters the syntax of all authentication mechanisms is explained. These ‘authentication commands’ (leading to the necessary authentication state) can be inserted for the placeholder *<Authentication>* which are given in the syntax of all security relevant commands described in this chapter 4.

4.10.1 LogonSign

With this command a user with RSA Signature authentication mechanism opens an authenticated Secure Messaging session for the given command. The session key is established using the Diffie-Hellman key agreement (according to [PKCS#3]).

The LogonSign command provides a combination of command authentication and opening of a Secure Messaging session in one step, and it simplifies the command syntax, i. e. it replaces and simplifies commands of the form

csadm [Dev=<device>] AuthRSASign=<user>,<keyspec> SessionDH=1024 <command>

The authenticated Secure Messaging session will automatically close when the command execution finishes and CSADM ends.
For any further CSADM command which needs authentication and/or confidentiality a new authenticated Secure Messaging session has to be opened.

Syntax	csadm [Dev=<device>] [Password=<password>] … … LogonSign=<user>,<keyspec> [<*further Authentication*>] <command>
Parameters	<device> device specifier (see 4.1.3) <password> if the private part of the user's authentication key is given in an encrypted key file: password of encrypted key file: • for hidden password entry: string 'ask' (see 4.1.5; hidden password entry is strongly recommended!); • otherwise: password of key file *(see 4.1.4 for encrypted key files)* <user> user name <keyspec> private part of the user's authentication key (key specifier of smartcard or key file, see 4.1.4) <command> command which has to be authenticated
Examples	csadm LogonSign=paul,:cs2:cyb:COM1 DeleteFile=xxx.mtc csadm LogonSign=paul,:cs2:cyb:COM1 … … LogonPass=pauline,1234hello DeleteFile=xxx.mtc
Output	none on success, or error message
Note	If the private part of the user's RSA authentication key is stored on a smartcard, the user will be prompted at the PIN pad to insert his smartcard and enter the PIN. The PIN pad has to be connected to the computer where the CSADM tool is running.

4.10.2 LogonPass

With this command a user with HMAC Password authentication mechanism opens an authenticated Secure Messaging session for the given command. The session key is established using the Diffie-Hellman key agreement (according to [PKCS#3]).

The LogonPass command provides a combination of command authentication and opening of a Secure Messaging session in one step, and it simplifies the command syntax, i. e. it replaces and simplifies commands of the form

csadm [Dev=<device>] AuthHMACPwd=<user>,<password> SessionDH=1024 <command>

The authenticated Secure Messaging session will automatically close when the command execution finishes and CSADM ends.
For any further CSADM command which needs authentication and/or confidentiality a new authenticated Secure Messaging session has to be opened.

Syntax	csadm [Dev=<device>] LogonPass=<user>,<password> ... ... [<*further Authentication*>] <command>
Parameters	<device> device specifier (see 4.1.3) <user> user name <password> user password or string ‘ask’ if hidden password entry should be used (see 4.1.5 and below; hidden password entry is strongly recommended!). <command> command which has to be authenticated
Examples	• csadm LogonPass=paul,swordfish DeleteFile=xxx.mtc • csadm LogonPass=paul,ask LogonPass=paula,ask DeleteFile=xxx.mtc
Output	none on success, or error message

The usage of hidden password entry is strongly recommended!
In this case CSADM requests password input (‘Enter Passphrase:’) and does not echo the input.

4.10.3 AuthRSASign

With this command a user with the authentication mechanism 'RSA Signature' authenticates a single command.

The following steps are performed during command execution:

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 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 RSA key from its user database.
5. If the verification has been successful (and if the necessary authentication state has been achieved), the CryptoServer will execute the command

Syntax	csadm [Dev=<device>] [Password=<password>] … … AuthRSASign=<user>,<keyspec> [<*further Authentication*>] <command>
Parameters	<device> device specifier (see 4.1.3) <password> if user's RSA key is given in an encrypted key file: password of encrypted key file: • for hidden password entry: string 'ask' (see 4.1.5; hidden password entry is strongly recommended!); • otherwise: password of key file *(see 4.1.4 for encrypted key files)* <user> user name <keyspec> private part of the user's RSA key (smartcard or key file, see 4.1.4) <command> command which has to be authenticated
Examples	csadm AuthRSASign=paul,:cs2:cyb:COM1 DeleteFile=xxx.mtc csadm AuthRSASign=paul,:cs2:cyb:COM1 LogonPass=pauline,1234hello DeleteFile=xxx.mtc
Output	none on success, or error message
Note	If the user's private RSA key is stored on a smartcard, the user will be prompted at the PIN pad to insert his smartcard and enter the PIN. The PIN pad has to be connected to a serial line of the computer where the CSADM tool is running.

4.10.4 AuthHMACPwd

With this command a user with the authentication mechanism 'HMAC Password' authenticates a single command.

The following steps are performed during command execution:

1. A challenge value is requested from the CryptoServer.
2. A HMAC value is calculated over the command data and the challenge value using the user's password as the HMAC key.
3. Command data and hash (HMAC) value are sent to the CryptoServer.
4. The CryptoServer re-calculates the HMAC value via the user's 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 (and if the necessary authentication state is achieved), the CryptoServer will execute the command

Syntax	csadm	[Dev=<device>] AuthHMACPwd=<user>,<password> ... [<*further Authentication*>] <command>
Parameters	<device>	device specifier (see 4.1.3)
	<user>	user name
	<password>	user password or 'ask' if hidden password entry should be used.
	<command>	command which has to be authenticated
Examples	• csadm AuthHMACPwd=paul,swordfish DeleteFile=xxx.mtc • csadm AuthHMACPwd=paul,ask DeleteFile=xxx.mtc	
Output	none on success, or error message	

The usage of hidden password entry is strongly recommended!
In this case CSADM requests password input ('Enter Passphrase') and does not echo the input.

4.10.5 ShowAuthState

CryptoServer's current authentication state is displayed.
For the meaning of the authentication state, see 3.2.2.

Syntax	csadm [Dev=<device>] ShowAuthState
Parameter	<device> device specifier (see 4.1.3)
Authentication	none
Output	**current AUTH state: 03000012**

4.11 Secure Messaging

With Secure Messaging, commands to and from the CryptoServer will be encrypted and integrity protected using a 32 bytes AES session key which was exchanged when opening the session.

See also 3.3 for a description of CryptoServer's Secure Messaging concept.

A session will automatically be closed when the command execution finishes and CSADM ends.

If a command is authenticated with LogonSign or LogonPass (i.e. one of the authentication commands which provide authentication and Secure Messaging in one step, see 4.10.1 and 4.10.2) it is not necessary to additionally use one of the Secure Messaging commands described in the following subsections.

In FIPS mode only Secure Messaging using the Diffie-Hellman key agreement is available. It is recommended always to use one of the LogonSign or LogonPass authentication commands which automatically open a Secure Messaging session.

4.11.1 SessionDH

This command opens a session for Secure Messaging using the Diffie-Hellman key agreement (according to [PKCS#3]).

Syntax	csadm [Dev=<device>] [<*Authentication*>] SessionDH=<size> <commands>	
Parameters	<device>	device specifier (see 4.1.3)
	<size>	size of the Diffie-Hellman parameters (prime) in bits, valid values are 512, 1024 and 2048 (in FIPS mode only 1024 and 2048)
	<commands>	commands which should be executed within the session
Authentication	authentication optional (mandatory in FIPS mode)	
Output	none on success or error message	
Note	If a user has authenticated the session, his permissions are granted to the whole session (to all commands following the *SessionDH* command).	

4.12 Administration of the CryptoServer LAN

This command group explicitly addresses the CryptoServer LAN (communication unit); commands will not be forwarded to a CryptoServer PCIe (PCIe card with the 'real' security module). Instead they will be processed by the control module of the TCP-Server ('csxlan'-daemon) and responded in the same way a firmware module would respond to a command.

These commands are used to administrate a CryptoServer LAN remotely (e. g. to get/set its configuration).

Since the commands of this group are not directed to any CryptoServer (PCIe card) but only to the CryptoServer LAN, the presence, mode or state of eventually underlying CryptoServers are not relevant for their performance. For the same reason, none of these commands have to be authenticated using the CryptoServer's authentication mechanism.
Nevertheless some commands of this group are protected against unauthorized use with an own authentication mechanism: these commands have to be authenticated with the root password of the CryptoServer LAN. See also [CSLANAdminManual].

4.12.1 CSLGetConnections

All current connections to the CryptoServer LAN are listed.

Syntax	csadm [Dev=<device>] CSLGetConnections
Parameter	<device> device specifier (see 4.1.3); "PCI:" addressing is not allowed
Authentication	none
Output	**current connections to csxlan** **# ip address port protocol** **1: 192.168.4.122 1131 TCP** **2: 192.168.4.098 2563 TCP** **3: 192.168.4.122 1137 TCP**
Note	• The following information is shown: ip-address: IP-address of the client PC port: port on the client PC used to open the connection protocol: used protocol (either UDP or TCP) • Up to 10.000 connections can be established to one CryptoServer LAN at the same time.

4.12.2 CSLScanDevices

The *CSLScanDevices* command returns the internal configuration of one or more CryptoServer LAN as set up in the configuration file '/etc/csxlan.conf' on each CryptoServer LAN (see [CSLANAdminManual]). If the command is called using the dedicated IP-address (as 'Dev='-parameter) of a CryptoServer LAN, only the configuration of this single CryptoServer LAN will be shown.

However, *CSLScanDevices* can also be used to scan the whole network for all CryptoServer LAN currently available. Therefore this command has to be sent as 'Multicast Message' by using the special IP address 'UDP:224.1.0.1' as device parameter (Dev=). It will be responded by every CryptoServer LAN which has been set up to respond to Multicast Messages (see [CSLANAdminManual]).

CSLScanDevices does only find such CryptoServer LAN which have been configured to respond Multicast Messages.

Syntax	csadm [Dev=<device>] CSLScanDevices
Parameter	<device> device specifier (see 4.1.3). "PCI:" addressing is not allowed. <device> can be for example 'UDP:224.1.0.1' (Multicast) if the whole network shall be scanned.
Authentication	none
Output	**CSLAN ip address port prot device** **----------** **1 192.168.4.201 288 TCP PCI:/dev/cs2a** **1 192.168.4.201 289 TCP TCP:192.168.10.50** **1 127.0.0.1 288 UDP PCI:/dev/cs2a** **1 224.1.0.1 288 UDP PCI:/dev/cs2a** **2 192.168.10.50 288 TCP PCI:/dev/cs2a** **2 192.168.10.50 57 TCP PCI:/dev/cs2a** **2 127.0.0.1 288 UDP PCI:/dev/cs2a** **2 224.1.0.1 288 UDP PCI:/dev/cs2a** **3 192.168.4.127 288 TCP PCI:/dev/cs2a** **3 192.168.4.127 289 TCP PCI:/dev/cs2b** **3 224.1.0.1 288 UDP PCI:/dev/cs2a**

Note	On a CryptoServer LAN the included CryptoServer PCI/PCIe card is accessible by setting up several parameters: • ip address: interface address (either the TCP/IP address of the CryptoServer LAN, the localhost address '127.0.0.1' or the Multicast address '224.1.0.1') • port: port the TCP-Server (daemon) listens to (default: 288, each device can be set up with multiple ports) • protocol: either TCP or UDP (UDP has a packet size limitation) • device: either a local CryptoServer (PCI/PCIe) or a second CryptoServer LAN (TCP)

4.12.3 CSLGetVersion

The version number of the TCP-Server ('csxlan'-daemon) is shown.

Syntax	csadm [Dev=<device>] CSLGetVersion
Parameter	<device> device specifier (see 4.1.3). "PCI:" addressing is not allowed.
Authentication	none
Output	**csxlan 3.0.5**

4.12.4 CSLGetLogFile

With this command the log file from the CryptoServer LAN can be retrieved.

Syntax	csadm [Dev=<device>] CSLGetLogFile[=<fileno>]
Parameter	<device> device specifier (see 4.1.3). "PCI:" addressing is not allowed. <fileno> number of the log file on the CryptoServer LAN. <fileno> must be between 1 and 9. If omitted the first log file (number 0) is retrieved.
Authentication	none
Output	content of log file on success, or error message
Note	• Use '>' to dump the output into a file on the local computer. **Example:** csadm CSLGetLogFile > c:\temp\csxlan.log • The log file is formatted according to the UNIX style. Use an appropriate editor (e. g. WordPad) on a Windows system. • The contents of the log file depend on the trace level setting of the CryptoServer LAN. See [CSLANAdminManual] for a description how to set up logging on the CryptoServer LAN.

4.12.5 CSLGetConfigFile

With this command the configuration file (‘/etc/csxlan.conf’) of the CryptoServer LAN can be retrieved.

Syntax	csadm [Dev=<device>] CSLGetConfigFile
Parameter	<device> device specifier (see 4.1.3). “PCI:” addressing is not allowed
Authentication	none
Output	contents of configuration file on success, or error message
Note	• Use ‘>’ to dump the output into a file. **Example:** csadm GetConfigFile > c:\temp\csxlan.conf • The configuration file is formatted according to the UNIX style. Use an appropriate editor (e. g. WordPad) on a Windows system. • See [CSLANAdminManual] for a description of how to configure CryptoServer LAN.

4.12.6 CSLPutConfigFile

This command imports a new configuration file into a CryptoServer LAN.

Syntax	csadm [Dev=<device>] CSLPutConfigFile=<password>,<file>
Parameters	<device> device specifier (see 4.1.3). "PCI:" addressing is not allowed. <password> root password of the CryptoServer LAN. If the password is given as 'ask' hidden password entry is possible. <file> configuration file to be imported
Authentication	special password authentication using the root password of the CryptoServer LAN
Output	none on success, or error message
Note	• Independently from the given file name the configuration file will be imported as '/etc/csxlan.conf'. • The imported file may also be formatted according to the Windows style. On the CryptoServer LAN the file will be reformatted to the UNIX format style. • See [CSLANAdminManual] for a description of how to configure the CryptoServer LAN.

4.12.7 CSLSetTracelevel

This command sets the trace level (== log level) of a CryptoServer LAN. Depending on the trace level more or less information will be written into the log file (‘/var/log/csxlan.log’). Each level is independent from the others (the *Verbose* level does not include the *Info* level).

Any ‘added’ combination of the following values can be used:

level	***value***	***description***
Info	0x80	informational messages like connection establishment, termination and the execution of functions of the control module are written to the log file
Verbose	0x40	details about the state machine are logged
Packet Data	0x20	the content of request and reply packets is logged (max. 256 bytes per packet)

Syntax	csadm [Dev=<device>] CSLSetTraceLevel=<password>,<level>
Parameters	<device> device specifier (see 4.1.3). “PCI:” addressing is not allowed. <password> root password of the CryptoServer LAN: • for hidden password entry: string ‘ask’ (see 4.1.5; hidden password entry is strongly recommended!); • otherwise: root password <level> desired trace level, may be any (added) combination of: Info: 0x80 Verbose: 0x40 Packet_Data: 0x20
Authentication	special password authentication using the root password of the CryptoServer LAN
Output	none on success or error message
Note	The new trace level is only a temporary setting. The next time CryptoServer LAN (re)starts, the level which is put down in the configuration file (‘/etc/csxlan.conf’) will be used again.

4.12.8 CSLShutdown

This command shuts down the CryptoServer LAN as the Linux command ‘shutdown -h’ would do at the local console. All programs and services will be stopped, all devices will be unmounted and the system will be put into ‘RunLevel 0’. The CryptoServer LAN can then be powered off.

Syntax	csadm [Dev=<device>] CSLShutdown=<password>
Parameter	<device> device specifier (see 4.1.3). “PCI:” addressing is not allowed. <password> root password of the CryptoServer LAN: • for hidden password entry: string ‘ask’ (see 4.1.5; hidden password entry is strongly recommended!); • otherwise: root password
Authentication	special password authentication using the root password of the CryptoServer LAN
Output	none on success or error message

4.12.9 CSLReboot

This command reboots the CryptoServer LAN the same way the Linux command ‘shutdown -r’ would do at the local console. All programs and services will be stopped, all devices will be unmounted and the system will rebooted again.

Syntax	csadm [Dev=<device>] CSLReboot=<password>
Parameter	<device> device specifier (see 4.1.3). “PCI:” addressing is not allowed. <password> root password of the CryptoServer LAN: • for hidden password entry: string ‘ask’ (see 4.1.5; hidden password entry is strongly recommended!); • otherwise: root password
Authentication	special password authentication using the root password of the CryptoServer LAN
Output	none on success or error message

4.12.10 CSLGetTime

This command reads the local system time of the CryptoServer LAN box (not the clock of the CryptoServer PCI/PCIe card) and outputs the date and time.

Syntax	csadm [Dev=<device>] CSLGetTime
Parameter	<device> device specifier (see 4.1.3). “PCI:” addressing is not allowed.
Authentication	none
Output	date and time of the CryptoServer LAN

4.12.11 CSLSetTime

This command sets the local system time of the CryptoServer LAN box (not the clock of the CryptoServer PCI/PCIe card)

Syntax	csadm [Dev=<device>] CSLSetTime=<password>,<time>
Parameter	<device> device specifier (see 4.1.3). “PCI:” addressing is not allowed. <password> root password of the CryptoServer LAN: • for hidden password entry: string ‘ask’ (see 4.1.5; hidden password entry is strongly recommended!); • otherwise: root password <time> ‘YYYYMMDDHHMMSS’ or ‘SYSTEM’
Authentication	special password authentication using the root password of the CryptoServer LAN
Output	none on success or error message
Note	• The time has to be given as digit string: YYYYMMDDHHMMSS where YYYY=year, MM=month, DD=day, HH=hour, MM=minute, SS=second • If SYSTEM is given as argument the system time of the host PC is used.

4.12.12 CSLGetSerial

This command reads and outputs the serial number of the CryptoServer LAN box (not the serial number of the CryptoServer PCI/PCIe card).

Syntax	csadm [Dev=<device>] CSLGetSerial
Parameter	<device> device specifier (see 4.1.3). “PCI:” addressing is not allowed.
Authentication	none
Output	serial number of the CryptoServer LAN (if available)

4.12.13 CSLGetLoad

This command reads and outputs the work load of the CryptoServer PCI/PCIe card in percent.

Syntax	csadm [Dev=<device>] CSLGetLoad
Parameter	<device> device specifier (see 4.1.3). “PCI:” addressing is not allowed.
Authentication	none
Output	Work load of the CryptoServer (ratio of the time that requests/commands spend in the CryptoServer PCI/PCIe card to the total time).

The information of the work load of the CryptoServer is read from the display module of the CryptoServer LAN. If the display module is busy (i.e. an operator is working at the display) this information is not available and the command does not output any value.

The value returned by the command is not the work load at that time but is an average value of the last 60 seconds. It is recalculated and updated every 5 seconds.

4.12.14 CSLListPPApps (ListPINPadApps)

Some applications (firmware modules) require a PIN pad (smartcard reader with display and keyboard) directly connected to CryptoServer Se‘s serial port or to the CryptoServer CSe’s USB port, e. g. for a secure input of a Master Backup Key. To easily execute such a command on a CryptoServer LAN, the corresponding function of the firmware module can be registered as a ‘PIN pad application’. Now the command is listed in the ‘PIN Pad Applications’ menu on CryptoServer LAN’s display and can be selected from there (see below).

ListPinPadApps shows all commands that have been registered in this way.

PIN pad applications are intended to be used on a CryptoServer LAN without the need of connecting a monitor and keyboard to it. The specific PIN pad application can be selected via the CryptoServer LAN display (as sub menu point under the `PIN Pad Applications` menu point, see below), all further user guidance will then be done over the PIN pad display.
Alternatively PIN pad applications can be executed like any other command using the function code (module ID) and subfunction code (command ID) returned by ListPinPadApps. Same as above, watch the PIN pad display for the further user guidance.

The CSLListPPApps command is not available in FIPS mode!

Syntax	csadm [Dev=<device>] ListPINPadApps
Parameter	<device> device specifier (see 4.1.3)
Output	**1. Create + store MBK on SC** **FC=0xc4 SFC=17 Data=** **2. Import MBK from SC** **FC=0xc4 SFC=18 Data=** **3. PIN change for MBK smartcard** **FC=0xc4 SFC=22 Data=** **4. Copy MBK smartcard** **FC=0xc4 SFC=23 Data=**
Note	• The registration of a PIN pad application has to be implemented during development of the firmware module! • A PIN pad application can be started with the CSADM tool by entering the command csadm Cmd=FC,SFC,Data where FC, SFC and Data are the parameters from the list of PIN pad applications (see also 4.13.1).

4.13 Miscellaneous Commands

In this chapter various helpful CSADM commands are described.

4.13.1 Cmd

The *Cmd* command provides a generic command interface. This makes it possible to send a command as a byte stream to any firmware module without having (developed) a special tool on the host PC.

Syntax	csadm [Dev=<device>] [<*Authentication*>] … … Cmd=<fc>,<sfc>,<$byte_1$>,<$byte_2$>,<$byte_3$>,...
Parameter	<device> device specifier (see 4.1.3) <fc> function code (module ID of the firmware module) (0x00, ..., 0x3FF or 0, ..., 1023) <sfc> sub function code (number of the called function) (0x00, ..., 0xFF, or 0, ..., 255) <$byte_n$> n^{th} data byte (dependent on the command) (0x00, ..., 0xFF or 0, ..., 255)
Example	csadm LogonPass=paul,swordfish Cmd=0x123,0x5,1,2,3,4,5,6,7,8
Authentication	depending on the command
Output	depending on the command
Note	The parameters fc, sfc and $byte_n$ can be coded either as decimal or as hexadecimal value (prefixed with "0x").

Although in theory it is possible to send commands of up to 256 KBytes to the CryptoServer using 'Cmd=', there is a restriction in praxis by the maximal command line length that can be entered if running the CSADM tool on a Windows system. If a longer command byte stream shall be executed, use the CmdFile command, see 4.13.2.

4.13.2 CmdFile

The *CmdFile* command provides a generic command interface. Unlike the *Cmd* command it reads the input data from a file. The length of the command contained in the file may be up to 256 kBytes.

This file may contain the following characters and strings:

Character	***Name***	***Description***
'**#**'	hash sign	rest of line is comment
','	comma	separator
' '	space	separator
TAB	tabulator	separator
CRLF	new line	separator
'hex'	tag	interpret all values as hexadecimal from now on, even if '0x' is omitted
'dec'	tag	return to normal mode (i. e. hexadecimal interpretation requires a leading '0x')
"..."	string	string which can reach the end of the line

Example:	
#	
# demo command file	
#	
0x123	# function code / module ID
0x05	# sub function code
0x00,0,0x00,11	# hexadecimal and decimal notation may be mixed
"Hello World"	# strings are framed with quotation marks
hex 0F,1E,2D,3C,4B,5A,60,59	# leading hex-tag allows omitting of '0x'!
68,77,86,95,A4,B3,C2,D1	# still understood as hexadecimal values
dec	# return to normal notation
11,22,33	# decimal values

Syntax	csadm [Dev=<device>] [<*Authentication*>] CmdFile=<file>
Parameters	<device> device specifier (see 4.1.3) <file> file (name and extension of the file do not matter)
Example	csadm LogonPass=paul,swordfish CmdFile=demo.cmd
Authentication	depending on the command
Output	depending on the command

4.13.3 CSTerm

This command retrieves messages from a serial port and displays them on the screen. This command is executed locally without sending commands to a CryptoServer. It just listens to a serial port.

Syntax	csadm CSTerm=<port>
Parameter	<port> serial port of the computer (e. g. COM1 or /dev/ttyS0)
Example	csadm CSTerm=COM1
Output	messages read from the serial port
Note	A serial port of that computer where the CSADM tool is running has to be connected to the serial port which shall be observed.

4.13.4 Sleep

The *Sleep* command delays the further command processing by the time given.

Syntax	csadm Sleep=<time>
Parameter	<time> time delay in seconds.
Example	csadm StartOS Sleep=3 GetState
Output	none
Note	In the example above a delay of 3 seconds is inserted between the execution of the *StartOS* and the *GetState* commands.

5 Configuration of CryptoServer via 'p11tool2'

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

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

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

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

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

The definitions and access rules as found in chapter 3.2.2.5 are helpful to understand the security concept of the CryptoServer in order to choose the correct configuration for the CryptoServer and should be read carefully before changing the configuration.

Remarks:

- A 's*lot'* according to PKCS#11 corresponds to Key Group 'SLOT_<nnnn>'.
- The security officer SO for slot <nnnn> according to PKCS#11 corresponds to user 'SO_<nnnn>' with permission mask '00000200' (e.g. Security Officer 'SO_0001' for slot 1 rsp. Key Group 'SLOT_0001').
- The slot user according to PKCS#11 for slot <nnnn> corresponds to user 'USER_<nnnn>' with permission mask '00000002' (e.g. user 'USER_0001' for slot 1 rsp. Key Group 'SLOT_0001').
- Users with different user names, Key Groups or permission masks (e.g. Key Managers with permission mask '00000020') have to be configured by an Administrator using the appropriate CSADM commands, see section 4.7.

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

Basic Commands (no authenticaton required)

Command	*Description*	*Authenticated by*[12]
Help	If called without any parameter, this command shows a list of all available *p11toolv2* commands. If the command name is given as a parameter, specific help will be provided.	none
PrintError	This command displays the corresponding error message text to an error code.	none
Version	This command shows the version of the *p11toolv2*.	none
ListSlots	This command displays a list of all slots in the system.	none
GetInfo	This command displays general information about the CryptoServer PKCS#11 library.	none
GetSlotInfo	This command displays information about a specific slot.	none
GetTokenInfo	This command displays information about a specific CryptoServer (in failover mode: about the currently active CryptoServer).	none
ListConfig	This command displays a list of all configuration attributes.	none
GetLocalConfig	This command displays the value of the local (host) configuration attribute.	none
GetBackupInfo	This command displays information about a given backup file.	none

General Commands (valid authentication required but several user roles possible)

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

[12] None: no authentication required

Any: valid authentication for any user role required

SO: Security Officer

CU/U: Cryptographic User or User

KM: Key Manager

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

Administrator commands (valid Administrator authentication required)

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

SO commands (valid Security Officer authentication required)

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

Key Management commands (valid Key Manager authentication required)

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

BackupInternalKeys	This command creates a backup of all available internal[13] keys within a slot.	CU/KM
BackupExternalKeys	This command creates a backup of all available external[13] keys within the slot.	CU/KM
RestoreInternalKeys	This command restores all keys from the given key backup file to the internal[13] key store.	CU/KM
RestoreExternalKeys	This command restores all keys from the given key backup file to the external[13] key store.	CU/KM

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

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

6 Batteries of the CryptoServer

The CryptoServer contains a battery to ensure that security relevant data are not lost even when the CryptoServer is turned off. If the device is not used over a prolonged period, i. e. during storage or if the computer is turned off, the battery will have a durability of half a year at a minimum. If the CryptoServer is permanently powered on, the battery is not discharged and the lifetime of the battery increases. This battery is called “carrier battery” because it is mounted on the PCIe carrier card/PCIe card.

See [CSPCIe-InstallManual] for how to exchange the carrier battery.

An additional external battery may be connected to the CryptoServer to increase battery lifetime. In this case the carrier battery is used only after the external battery is exhausted. The CryptoServer LAN is always equipped with an external battery with a durability of 2 years at a minimum.

See [CSLAN-InstallManual] for how to exchange the external battery.

The state of the batteries has to be checked regularly.

In case that carrier and external battery are both exhausted, all sensitive data inside the CryptoServer will be deleted!

6.1 Check State of the Batteries

To check the state of the batteries the command “GetBattState” of the CSADM tool should be used (see 4.5.4). The output of this command shows the state of the batteries:

Output	Meaning
`Carrier Battery:  ok  (3.070 V)` `External Battery: ok  (3.553 V)`	All connected batteries contain sufficient charge.
`Carrier Battery:  low  (2.650 V)` `External Battery: ok  (3.553 V)`	The carrier battery of the CryptoServer has low power and must be exchanged.
`Carrier Battery:  ok  (3.070 V)` `External Battery: low  (3.100 V)`	The external battery has low power and must be exchanged.
`Carrier Battery:  ok  (3.068 V)` `External Battery: absence`	No external battery connected to the CryptoServer (e.g. PCIe card).

Only the CryptoServer LAN is equipped with an external battery.
The CryptoServer PCIe card is delivered without external battery.

7 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 (permission level 2 in user groups 6 and 7).
All user(s) who want to perform the administrative tasks should have their authentication token at hand.
For the first set-up of the CryptoServer in particular the Default Administrator Key of default administrator user ADMIN is needed.

7.1 How to Install the CryptoServer

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

.

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 either of these installation manuals:

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

Follow the instructions in section 4.1.2 in order to install the command line tool *csadm* which is needed for the next steps during set-up.

7.2 Generate User's Authentication Token

For every user who shall authenticate himself towards the CryptoServer with an RSA signature, an authentication token has to be generated. Basically the private part of the user key can either be stored in a key file, which optionally is password-encrypted, or on a smartcard. As the private part of the key can't be read out from the smartcard, the usage of smartcards is more secure and therefore recommended.

Before delivery Utimaco creates the initial user 'ADMIN' on every CryptoServer as the default administrator. The authentication token of ADMIN is shipped by Utimaco as clear text key file ('init_dev_prv.key') and is initially stored on every smartcard that is shipped by Utimaco (with default PIN '123456').

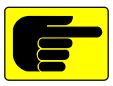
The customer should either replace the authentication token of ADMIN with a self-generated RSA key (see command ChangeUserRSASign) or create other users with sufficient permissions on the CryptoServer and then delete the user ADMIN.

Preconditions:

If smartcards shall be used, at least one smartcard per user has to be at hand. If the smartcard already contains a key, this key will be overwritten. Furthermore a smartcard reader (PIN pad) has to be connected to the serial line or USB port of the computer.

What to do:

1. Create a new RSA key pair as an encrypted key file (command *GenKey*, see 4.8.1).



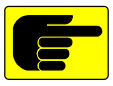
```
csadm NewPassword=ask KeyType=RSA …
… GenKey=MyRsa.key,2048,MyName

generating RSA key: MyRsa.key, 2048 bits, owner: MyName
Enter New Passphrase:
Repate New Passphrase:
> Follow the instructions on the PIN pad reader.
```

The generated key file can already be used to authenticate to the CryptoServer.

If the key shall not be copied on a smartcard the following steps can be skipped.

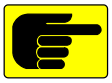
2. Store the key on a smartcard (command *SaveKey*, see 4.8.2).



```
csadm Password=ask KeyType=RSA PrvKey=MyRsa.key …
… Savekey=:cs2:cyb:COM3

Enter Passphrase:
> Follow the instructions on the PIN pad reader.
```

3. Change the PIN of the smartcard (command *ChangePIN*, see 4.8.7)

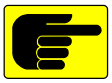


```
csadm ChangePIN=:cs2:cyb:COM3
> Follow the instructions on the PIN pad reader.
```

4. *Depending on your security policy, the following step 4 can be omitted if steps 2 and 3 will be performed several times, and if the this way created additional smartcards will be used as backup for the generated key.*

   Create a backup of the private key on two smartcards (command *BackupKey*, see 4.8.3).

   A *back-up smartcard* is used for key storage only. It cannot be used directly for authentication towards the CryptoServer: a back-up smartcard contains only one XOR-half of a key, so two back-up cards are necessary to regain the complete key. The key halves can be read out of the card to generate new administration smartcards containing the stored key at a later time (command *SaveKey*, see 4.8.2).



```
csadm KeyType=RSA PrvKey=MyRSA.key Password=ask …
… BackupKey=:cs2:cyb:COM3

Enter Passphrase:
> Follow the instructions on the PIN pad reader.
```

5. Store the key file at a protected place (e.g. on a USB-stick in a safe) or delete it (in case you made a backup copy on two smartcards).
6. Create User on CryptoServer with the *AddUserRSASign* command, see 4.7.2.

7.3 How to Enter FIPS-Mode: Set-Up and First Personalization of a New CryptoServer

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

The personalization process comprises the following initial steps:

- Change the authentication token of the default administrator 'ADMIN'
- Load FIPS firmware package and enter FIPS mode
- Create other users

This personalization process can furthermore be necessary

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

In this chapter it will be explained step by step how a CryptoServer can be personalized in order to enter FIPS mode.

The process of personalizing a CryptoServer can only be performed by the CryptoServer's System Administrator*, i. e. by a user who is allowed to assume the Administrator role.*

Precondition:

- CryptoServer and CSADM tool are installed and running.
- The authentication token(s) of a user (or a group of users) with (added) user permission '22000000' (or higher) is available. If the CryptoServer is set up for the first time,[14] this means that the authentication key of the default administrator ADMIN is at hand (either as key file 'init_dev_prv.key', given e.g. on CryptoServer's product CD, or on a smartcard as delivered by Utimaco (with default PIN '123456')).
- Furthermore the appropriate FIPS firmware package
  - Se series: 'SecurityServer-Se-Series-FIPS-3.0.2.0.mpkg'
  - CSe series: 'SecurityServer-CSe-Series-FIPS-4.0.3.0.mpkg'

  is needed containing all firmware modules that are mandatory in FIPS mode; see chapter 10 for a complete list.
- For every new user that shall be created on the CryptoServer, his authentication token has to be at hand. See 7.2 for the creation of customer-individual administration keys.

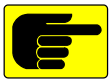
What to do:

1. Perform a *GetState* command (see 4.6.1).

[14] or after a *ClearToFactoryDefaults* had been performed (see 4.6.18 and 3.6, this command is only available in personalization mode and after alarm)

- The CryptoServer should be in *initialized* state and in *operational mode* or in *maintenance mode,* and the indicator for the FIPS Approved mode (FIPS mode = ON) is not shown (see 7.5). If this is not the case, see sections 7.7 and 8.1 for help.
- Alarm state should be “off”. If this is not the case, see 8.2 for help.

2. Replace the key of the default administrator ‘ADMIN’ (command ChangeUserRSASign, see 4.7.3, which has to be authenticated by ADMIN himself).



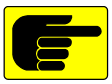
```
csadm LogonSign=ADMIN,:cs2:cyb:COM3 …
… ChangeUserRSASign=:cs2:cyb:COM3

> Follow the instructions on the PIN pad reader (Insert new
  smartcard first and old smartcard on second prompt)
```

For the implementation of e.g. the 2-Persons-Rule, instead of replacing the key of the default administrator alternatively the user ADMIN can be deleted after one or more user with sufficient permissions have been created.

In order to implement the 2-Persons-Rule: Perform step 3 instead of step 2. Doing this, please pay attention to the restrictions as explained in chapter 3.2.4.1.

3. Alternatively (or additionally) to step 2: Create other users on the CryptoServer (commands AddUserXXX, see 4.7), possibly replacing the default administrator.



```
csadm LogonSign=ADMIN,:cs2:cyb:COM3 …
…
AddUserHMACPwd=paul,00000002{CXI_GROUP=*},no_login+sma,ask

Enter New Passphrase:
Repeat New Passphrase:
> Follow the instructions on the PIN pad reader
```

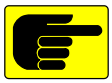
4. Load FIPS firmware module package by using command LoadPKG and setting flag ‘ForceClear’ (see 4.6.16):

Se series:

```
csadm LogonSign=ADMIN,:cs2:cyb:COM3 …
… LoadPkg=SecurityServer-Se-Series-FIPS-
3.0.2.0.mpkg,ForceClear

> Follow the instructions on the PIN pad reader.
```

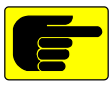
CSe series:



```
csadm LogonSign=ADMIN,:cs2:cyb:COM3 …
… LoadPkg=SecurityServer-CSe-Series-FIPS-
4.0.3.0.mpkg,ForceClear

> Follow the instructions on the PIN pad reader.
```

Verify whether the CryptoServer is now in FIPS mode and is not in any error state (using the *GetState* command, see 4.6.10 and 7.5):

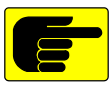


output of the *GetState* command should be

```
mode       = Operational Mode
state      = INITIALIZED (0x00100004)
FIPS mode  = ON
temp       = (…)
(…)
```

If this is not the case: you can check if all firmware modules have been started properly using the *ListModulesActive* command (see 4.6.10). If some firmware modules have not been initialized yet, use the *GetBootLog* command (see 4.6.11) to analyze the problem.

5. Optionally: Set the CryptoServer's clock with the SetTime command (see 4.6.9).



```
csadm LogonSign=ADMIN,:cs2:cyb:COM3 SetTime=GMT

> Follow the instructions on the PIN pad reader.
```

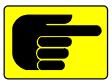
7.4 How to Generate a Master Backup Key

Preconditions:

If the Master Backup Key shall be stored on smartcards, at least two smartcard have to be at hand. If one of the smartcards already contains a key, this key will be overwritten. Additionally a smartcard reader (PIN pad) has to be connected to the serial line or USB port of the computer where the CSADM tool is running. Watch the PIN pad's display for instructions on further command processing.

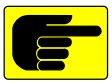
What to do:

1. Create a new AES MBK key and store it in shares on two smartcards (command *MBKGenerateKey*, see 4.9.2):



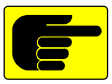
```
csadm LogonSign=ADMIN,:cs2:cyb:COM3 Key=:cs2:cyb:COM3,15
   MBKGenerateKey=AES,32,2,2,myMBK
```

2. Change the PIN of the smartcards (command *ChangePIN*, see 4.8.7)



```
csadm ChangePIN=:cs2:cyb:COM3
```

3. Restore the MBK key in slot 3 on the CryptoServer (command *MBKImportKey*, see 4.9.3):



```
csadm LogonSign=ADMIN,:cs2:cyb:COM3 Key=:cs2:cyb:COM3,15
   MBKImportKey=3
```

   If the key slot was still empty the new key is accepted without any further verification. If key slot 3 already contains an MBK the old MBK has to be given. In this case the CryptoServer verifies if the existing MBK matches the old key before it overwrites it with the new key.

4. Store the MBK shares at a protected place (e.g. in a safe).

7.5 How to Get State Indicators

For all administrator's tasks it is vital to be informed about the CryptoServer's current state and mode.

Precondition:

None.

What to do:

Perform a *GetState* command (see 4.6.1).

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

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

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

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

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

2. GetState command returns `state = DEFECT:`

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

   ➔CryptoServer is in **defect state**.

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

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

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

3. GetState command returns `state = INITIALIZED:`

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

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

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

   ➔ CryptoServer is in **Personalization Mode.** This means that the CryptoServer is (not in power-down mode and) NOT in FIPS mode (see chapter 2.2.5). The following sub-states are possible:

   3.1.1 GetState command returns `mode = Bootloader Mode`

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

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

The CryptoServer is ready for the personalization process, see 7.3. This process can only be performed by an *Administrator*.

3.1.2 GetState command returns `mode = Maintenance Mode`

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

3.1.2.1 GetState command returns `alarm = ON`:

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

bl_ver     = (…)
(…)
```

➔ CryptoServer is in **alarm state**:

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

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

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

For more information about possible alarms, see section 3.5.

No command can be performed before the alarm is set back. See section 8.2 for alarm treatment.

3.1.2.2 GetState command returns `alarm = OFF`:

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

➔ No alarm is given. This may happen if a *RecoverOS* command has been performed. The CryptoServer is ready for the personalization process, see 7.3. This process can only be performed by an *Administrator*.

3.1.3 GetState command returns `mode = Operational Mode`

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

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

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

3.2 GetState command returns `FIPS mode = ON`:

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

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

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

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

The number behind the '`FIPS error state`' indicator serves for error analysis and serves here just as an example. Only basic status request services are available. For leaving the FIPS error state, see 7.6.

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

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

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

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

The number behind the '`FIPS error state`' indicator serves for error analysis and serves here just as an example. Only some non-sensitive services (like status request services) are available. For leaving the FIPS error state, see 7.6

(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.):

3.2.3 GetState command returns no `FIPS error state.`

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

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

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

The state indicators can occur in various (but not all) combinations (see also 2.2.5 and 4.6.1).

7.6 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.

Precondition:

The CryptoServer is in any FIPS error state, i. e. the *GetState* command (see chapter 4.6.1) answers with '**FIPS error state = ON**'.[15]

What to do:

1. Perform the *Restart* command or power-cycle the CryptoServer.
2. Check if the module is still in FIPS error state by performing the *GetState* command. If not, 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 not, you are ready. If yes, go to step 5.
5. The CryptoServer has to be erased completely. To do this, trigger artificially an alarm by performing an *External Erase*: This has to be done manually by a short-circuit of the 'External Erase' pins on the PCIe-card, or by pressing the sealed delete switch inside the battery compartment of the CryptoServer LAN device (see 3.5.1).
6. Execute *GetState* again to check the success. The alarm must be indicated, but the alarm reason is no longer present, i. e. '**alarm = ON**' and '**alarm has occured**' should be returned.[16]
7. If this is not the case: please contact the manufacturer/Utimaco.
8. Else: The CryptoServer is in personalization mode now. Perform the *ResetAlarm* command to reset the alarm (see chapter 8.2).
9. Execute *GetState* again. The alarm state should now be 'OFF' ('**alarm = OFF**'). If this is not the case, please contact the manufacturer/Utimaco.
10. Else: Since the alarm has erased most data including the FIPS140 firmware module, the CryptoServer is in personalization mode and a new personalization of the CryptoServer must be performed now (see chapter 7.3). If this was successful, the CryptoServer has entered FIPS mode again.
11. Execute *GetState* to check if the CryptoServer is still in FIPS error state. If this is not the case, you are ready. If the module is still in any FIPS error state, please contact the manufacturer/Utimaco.

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

[16] In the following line the alarm reason **'external Erase is executed'** will be displayed.

7.7 How to Clear the CryptoServer

You want to clear all data inside of a CryptoServer and reload the complete firmware. This may be useful (for example) in the following situations:

Case 1: You want to securely clear all secret data inside a CryptoServer.

Case 2: Emergency case only! You have lost your customer-individual administrator keys and want to regain an administrable CryptoServer system.

In both cases after clearing the CryptoServer will no longer be in FIPS mode.

Please be aware that in any case you will lose all your customer-specific data that are stored inside the CryptoServer, in particular all stored keys.

*In emergency **case 2** you will have to set the CryptoServer back to the factory default setting in which it usually is delivered.*
Follow the instructions for case 2 only if this emergency case is given, i.e. if you have lost your customer-individual administrator keys!

*If you follow the instructions for **case 1**, you will get a similar result than with the instructions for case 2, but with the exception that all users in the user database that use an RSA key as authentication token will remain (i.e. only users with HMAC password mechanisms will be erased).*

See also section 3.6 for an explanation of the different clear options.

Precondition:

Case1:

A user with sufficient administrative rights (permission level 2 in user group 6) is needed. His authentication token has to be at hand.

Case 2:

Direct physical access to the CryptoServer hardware (PCIe card) is needed as the external erase circuit has to be short-cut.

What to do:

Case 1:

1. Quit FIPS mode by deleting the firmware module ‘fips140.msc’ and restarting the CryptoServer (see 4.6.7 and 4.5.3).
2. Perform the normal ***Clear*** command (see 4.6.18), option ‘csadm Clear=INIT’. This command has to be authenticated by a user with administrative rights.
3. Restart the CryptoServer (command *Restart*, see 4.5.3).
4. Now you may set-up the CryptoServer again.

All secret data will be deleted on the CryptoServer (e.g. key databases).

On a CSe all data are deleted including all users in the user database! The default administrator 'ADMIN' (see 3.2.4.1) will be restored and the CryptoServer must be initialized from the beginning as described in section 7.3.

On an Se only users with a password authentication mechanism will be deleted. Users with signature authentication (e.g. the default administrator ADMIN) will not be deleted, because only the public key part of their authentication key is stored on the CryptoServer.

Case 2:

1. Execute an *External Erase* on the CryptoServer (see 3.5.1). Afterwards the CryptoServer is no longer in FIPS mode.
2. Perform the ***ClearToFactoryDefaults*** command (syntax 'csadm Clear=DEFAULT', see 4.6.18). This command does not have to be authenticated.
3. Restart the CryptoServer (command *Restart*, see 4.5.3).
4. Reset CryptoServer alarm (command *ResetAlarm*, see 4.6.19).

The CryptoServer will be cleared into delivery state and the default administrator ADMIN will be restored with his initial authentication key.

In both cases most data including the FIPS control module are deleted. The CryptoServer is in personalization mode and a new FIPS personalization of the CryptoServer must be performed now (see chapter 7.3). If this was successful, the CryptoServer has entered FIPS mode again.

8 Troubleshooting

8.1 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.

Precondition:

- If the CryptoServer is installed on your local computer, make sure that the PCIe driver is running and that the administration tool CSADM is installed.
- If the CryptoServer is a part of a CryptoServer LAN, make sure that CryptoServer LAN and client-PC are properly connected to the network (try to 'ping' the LAN box from the client-PC) and that the administration tool CSADM is installed on the client-PC.

What to do?

(1) Perform GetState command (see 4.6.1). Compare answers with state indicators as listed in chapter 7.5

State indicator	Result	Explanation/Reason/Adjustment
1.	No answer	Dead state or power down mode Reset or power-cycle the CryptoServer.
2.	state not INITIALIZED	CryptoServer is not correctly initialized or even defect. ⇒ Please get in contact with manufacturer/Utimaco.
3.1.1	state = INITIALIZED, ALARM = off, mode = BL Mode, but CryptoServer is not in FIPS mode	You may analyze the problem using the *ListModulesActive* (see (2)) and *GetBootLog* commands: Check if all necessary firmware modules are present and successfully initialized (see chapter 10 for a complete list). After that the CryptoServer should be set-up and personalized again, see 7.3.
3.1.2.1	state = INITIALIZED mode = Maintenance mode alarm = ON	An alarm has occurred (and is possibly physically still present) ⇒ See chapter 8.2 for alarm treatment (and 3.5 for more information about CryptoServer alarms).
3.1.2.2	mode : Maintenance mode state : INITIALIZED alarm : OFF CryptoServer is not in FIPS	Only the backup set of system firmware modules (*.sys) has been started (see 2.2.5). No cryptographic services are available. CryptoServer is in personalization mode or

State indicator	Result	Explanation/Reason/Adjustment
	mode	not correctly initialized in FIPS mode. ⇒ Re-initialize the CryptoServer, see 7.3
3.1.3	mode : Operational state : INITIALIZED alarm : OFF but CryptoServer is not in FIPS mode	CryptoServer is in personalization mode or not correctly initialized in FIPS mode. ⇒ Re-initialize the CryptoServer, see 7.3.
3.2.1, 3.2.2	state INITIALIZED, FIPS mode = ON, CryptoServer is in FIPS error state, ALARM = off	Quit error state, see 7.6.
3.2.3	state = INITIALIZED, FIPS mode = ON, CryptoServer is not in FIPS error state	Everything is ok. End.
	Error B9011xxx, B9015xxx, B9016xxx, B9017xxx or B9021xxx until B9024xxx	CryptoServer's PCIe carrier card does not react. ⇒ Try a restart (3).
	other errors: B901xxxx or B902xxxx	No connection to CryptoServer / CryptoServer LAN, communication problem, wrong host or device name, problem with network. ⇒ Check parameters. ⇒ Perform 'ping' at CryptoServer LAN. ⇒ Check state/configuration of the TCP daemon on the CryptoServer LAN.

(2) Perform *ListModulesActive* command (see 4.6.10).

Result	Explanation/Reason/Adjustment
All necessary modules are listed and initialized (INIT_OK).	OK. CryptoServer is in *operational mode* and ready to work. End.
Some necessary modules are missing in the given list.	Modules are not loaded onto CryptoServer. ⇒ Check presence of modules with *ListFiles* command. Load missing modules with *LoadFile* and restart CryptoServer. If then modules cannot be started: ⇒ Perform *GetBootLog* and search boot log for errors.
At least one module is not initialized (i. e. not INIT_OK or INACTIVE).	Firmware module(s) cannot be started (e. g. module dependencies cannot be resolved). ⇒ Perform *GetBootLog* and search boot log for errors

(3) Perform *Restart* command (see 4.5.3).

Result	Explanation/Reason/Adjustment
no error	OK ⇒ back to (1)
other error	⇒ Switch CryptoServer's power off and on again, then back to (1) If this does not help: may be hardware problem ⇒ Please contact manufacturer/Utimaco

8.2 Alarm Treatment

An alarm can be triggered on the CryptoServer for various physical reasons, like e. g. the temperature being too high or too low, empty battery, the tamper detection foil has been damaged or after an External Erase had been executed. See chapter 3.5 for a list of all possible alarm reasons and a detailed description of the CryptoServer's alarm mechanism.

Most alarm reasons can be removed (e. g. exchange low battery or cool down high temperature). The *GetState* command shows the reason of an alarm and if the alarm is still present, see 4.6.1. If the reason for an alarm cannot be removed then please get in contact with the manufacturer/Utimaco, else you can reset the pending alarm state (see below).

Precondition:

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.

What to do?

1. If the alarm is physically still present: Remove the alarm cause if possible.
2. Then restart the CryptoServer (command *Restart*, see 4.5.3).
3. Execute *GetState* again (see chapter 4.6.1) to check the success. Even if the reason for the alarm has been removed, the alarm state will still be 'ON', but the alarm should no longer be shown as 'present' (only as 'has occurred').
4. If the alarm is still shown as 'present': please contact the manufacturer/Utimaco.
5. Else: Perform the *ResetAlarm* command (see chapter 4.6.19), then restart the CryptoServer (command *Restart*, see 4.5.3).
6. Execute *GetState* again. The alarm state should now be 'OFF'.
7. Please be aware that all users who have used a password authentication mechanism are lost after an alarm. These users have to be replaced, if needed.
8. As the alarm has deleted the FIPS mode validation module (firmware module FIPS140) you have to perform the personalization process again as described in 7.3 to reach FIPS mode again.

9 Built-in Elliptic Curves

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

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

Name(s)	Size	Defined in:
NIST-P192 / secp192r1	192	[FIPS186-2], [ANSI-X9.62], [SEC2]
NIST-P224 / secp224r1	224	[FIPS186-2], [ANSI-X9.62], [SEC2]
NIST-P256 / secp256r1	256	[FIPS186-2], [ANSI-X9.62], [SEC2]
NIST-P384 / secp384r1	384	[FIPS186-2], [ANSI-X9.62], [SEC2]
NIST-P521 / secp521r1	521	[FIPS186-2], [ANSI-X9.62], [SEC2]
NIST-K163 / sect163k1	163	[FIPS186-2], [ANSI-X9.62], [SEC2]
NIST-B163 / sect163r2	163	[FIPS186-2], [ANSI-X9.62], [SEC2]
NIST-K233 / sect233k1	233	[FIPS186-2], [ANSI-X9.62], [SEC2]
NIST-B233 / sect223r1	233	[FIPS186-2], [ANSI-X9.62], [SEC2]
NIST-K283 / sect283k1	283	[FIPS186-2], [ANSI-X9.62], [SEC2]
NIST-B283 / sect283r1	283	[FIPS186-2], [ANSI-X9.62], [SEC2]
NIST-K409 / sect409k1	409	[FIPS186-2], [ANSI-X9.62], [SEC2]
NIST-B409 / sect409r1	409	[FIPS186-2], [ANSI-X9.62], [SEC2]
NIST-K571 / sect571k1	571	[FIPS186-2], [ANSI-X9.62], [SEC2]
NIST-B571 / sect571r1	571	[FIPS186-2], [ANSI-X9.62], [SEC2]

10 Appendix: FIPS Validated Firmware Package

The following firmware modules are mandatory in FIPS mode (FIPS approved mode of operation according to [FIPS140-2]) and have to be loaded (in MTC format) by the CryptoServer's *Administrator* during the setup and personalization process (see 7.7):

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

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

If the FIPS Mode Control Module FIPS140 is loaded but the other loaded firmware modules are not identical with the above listed firmware modules

- ***the CryptoServer Se is NOT in validated FIPS mode, but most of the restrictions implemented for FIPS mode are nevertheless applied. In particular all restrictions for the usage of non-approved cryptographic algorithms and key lengths are applied. This mode is indicated as "FIPS restrictions applied" (see getstate, section 4.6.1).***

- ***the CryptoServer CSe is in FIPS error state. This state can only be left by executing an External Erase which deletes all sensitive data!***

The set of FIPS approved firmware modules is provided as firmware package 'SecurityServer-Se-Series-FIPS-3.0.2.0.mpkg' or 'SecurityServer-CSe-Series-FIPS-4.0.3.0.mpkg' *on the product CD of the SecurityServer.*

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

- version 3.0.3.0 on a CryptoServer Se and
- version 4.0.5.0 on a CryptoServer CSe.

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

11 References

No.	Title/Company	Doc.-No.
[AIS20]	AIS 20, Version 1: " Functionality classes and evaluation methodology for deterministic random number generators, Version 2.0 of 2.12.1999"; BSI (Bundesamt für Sicherheit in der Informationstechnik - Federal Office for Information Security; Germany) http://www.bsi.bund.de/zertifiz/zert/interpr/ais_cc.htm	
[AIS31]	AIS 31, Version 1: "Functionality classes and evaluation methodology for true (physical) random number generators, Version 3.1 of 25.9.2001"; BSI (Bundesamt für Sicherheit in der Informationstechnik - Federal Office for Information Security; Germany) http://www.bsi.bund.de/zertifiz/zert/interpr/ais_cc.htm	
[ANSI-X9.62]	ANS X9.62-2005: Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA) / ANSI (American National Standards Institute)	
[ANSI-X9.63]	ANSI X9.63-2001: Public Key Cryptography for the Financial Services Industry, Key Agreement and Key Transport Using Elliptic Curve Cryptography / ANSI (American National Standards Institute)	
[BP]	ECC Brainpool Standard Curves and Curve Generation, v1.0, 19.10.2005, www.ecc-brainpool.org	
[CSXAPI]	CryptoServer – Extended Application Interface (CSXAPI) / Utimaco IS GmbH	2007-0003
[CSCMDS]	CryptoServer – Firmware Module CMDS – Interface Specification – CMDS Version ≥ 3.0.0.0 / Utimaco IS GmbH	2009-0002
[CSSMOS]	CryptoServer – Operating System SMOS – SMOS Version ≥ 2.5.0.0 – Interface Specification / Utimaco IS GmbH	2008-0001
[CSCXI]	CryptoServer Cryptographic Service Interface – Firmware Module CXI – Interface Specification / Utimaco IS GmbH	2008-0009
[CSPCIe-InstallManual]	CryptoServer PCIe – Operating & Installation Manual – Se-Series / Utimaco IS GmbH	M010-0004-en
	CryptoServer PCIe – Operating & Installation Manual – CSe-Series / Utimaco IS GmbH	M013-0002-en
[CSLAN-InstallManual]	CryptoServer LAN – Operating Manual – Se-Series / Utimaco IS GmbH	M010-0006-en
	CryptoServer LAN – Operating Manual – CSe-Series / Utimaco IS GmbH	M013-0003-en

No.	Title/Company	Doc.-No.
[CSLAN-AdminManual]	CryptoServer LAN – Manual for System Administrators / Utimaco IS GmbH	M010-0002-en
[CSFIPS-UserGuide]	CryptoServer – User's Guide for CryptoServer Se/CSe in FIPS Mode / Utimaco IS GmbH	2011-0003
[CSP11-ToolGuide]	CryptoServer Manual for CryptoServer PKCS#11 Administration Tool Release 2	2012-0004
[FIPS140-2]	FIPS PUB 140-2, Security Requirements for Cryptographic Modules / National Institute of Standards and Technology (NIST), May 2001	
[FIPS186-2]	FIPS PUB 186-2, Digital Signature Standard / 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	
[SEC2]	SEC2: Recommended Elliptic Curve Domain Parameters – Certicom Research – September 20, 2000, Version 1.0	