

CryptoServer

Administration Manual



Imprint

Copyright 2024	Utimaco IS GmbH Germanusstr. 4 D-52080 Aachen Germany
Phone	AMERICAS +1-844-UTIMACO (+1 844-884-6226) EMEA +49 800-627-3081 APAC +81 800-919-1301
Internet e-mail	https://support.hsm.utimaco.com/ support@utimaco.com
Document Version	2.15.30
Product Version	6.0.0
Date	2024-11-25
Document No.	M010-0001-en
Status	PUBLISHED

All rights reserved	<p>No part of this documentation may be reproduced in any form (printing, photocopy or according to any other process) without the written approval of Utimaco IS GmbH or be processed, reproduced or distributed using electronic systems.</p> <p>Utimaco IS GmbH reserves the right to modify or amend the documentation at any time without prior notice. Utimaco IS GmbH assumes no liability for typographical errors and damages incurred due to them. Any mention of the company name Utimaco in this documents refers to the Utimaco IS GmbH.</p> <p>All trademarks and registered trademarks are the property of their respective owners.</p>
---------------------	--

Table of Contents

1	Introduction	9
1.1	Target Audience for this Manual	9
1.2	Document Conventions	9
1.3	Other Manuals	10
1.4	Abbreviations	12
2	Overview	14
2.1	Hardware	14
2.1.1	IRAM and Key-RAM	15
2.1.2	Random Number Generator (RNG)	16
2.1.3	Physical External Housing	16
2.1.4	Sensors.....	17
2.1.5	Tamper-protecting Foil.....	17
2.1.6	CryptoServer Batteries	17
2.2	Software	19
2.2.1	Firmware Modules	19
2.2.1.1	Firmware Module Management.....	21
2.2.1.2	Firmware Package.....	24
2.2.2	Signed License Files.....	26
2.2.3	CryptoServer Boot Process	26
2.2.3.1	Manually Starting SMOS.....	27
2.3	Administration.....	28
2.3.1	CryptoServer Administration Tool (CAT)	29
2.3.2	CryptoServer Command-line Administration Tool (csadm)	29
2.3.3	CryptoServer LAN Menu Options	30
2.3.4	Scenarios.....	30
2.3.4.1	Scenario 1: Local CryptoServer Tools/API, Local CryptoServer PCIe	30
2.3.4.2	Scenario 2: Local CryptoServer Tools/API, Remote CryptoServer LAN	31
2.3.4.3	Scenario 3: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN Without Blocking Firewall.....	32
2.3.4.4	Scenario 4: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN With Blocking Firewall	33
2.3.4.5	Scenario 5: Mixed Scenarios	34
3	Concepts.....	37
3.1	Security Mechanisms	37

3.1.1	Secure Messaging	37
3.1.2	Mutual Authentication	38
3.1.3	An Alarm and Its Consequences	38
3.1.4	Clear	42
3.1.5	Clear to Factory Settings	43
3.1.6	External Erase.....	44
3.2	Smartcards, PIN Pads and Keyfiles	47
3.2.1	Smartcards	47
3.2.2	PIN Pads.....	47
3.2.3	Keyfiles.....	49
3.3	User Management: Users, User Groups and Authentication.....	49
3.3.1	Users	49
3.3.2	User Groups.....	50
3.3.3	Permissions and Authentication Status	51
3.3.4	Permissions for New Users	53
3.3.5	Authentication Mechanisms.....	54
3.3.5.1	HMAC Password Authentication	54
3.3.5.2	Signature-based authentications	55
3.3.5.3	Naming of Authentication Mechanisms in CAT and csadm.....	58
3.3.6	Authentication Mode	60
3.3.7	ADMIN, the Default Administrator	60
3.3.8	Default Cryptographic User	62
3.3.9	PKCS#11 Users	62
3.3.10	Configurable Role-based Access Control (C-RBAC).....	65
3.3.11	Backup of Users and Keys	65
3.4	CryptoServer Modes and States	65
3.4.1	Operating Modes	65
3.4.2	Operating States.....	67
3.4.3	Error States	67
3.5	Implementation Environments	69
3.6	Clustering for Load Balancing and Failover	71
3.6.1	Load Balancing	71
3.6.2	Failover.....	72
3.6.3	Error Handling	74

3.6.4	Use of External and Internal Key Store	74
3.6.4.1	ODBC External Keystore	76
3.6.4.2	Copy External Key Store	87
3.6.4.3	Key Replication and Internal Keystore	89
3.7	System Keys	89
3.7.1	Master Key	89
3.7.2	Default Administrator Key	90
3.7.3	Module Signature Key	92
3.7.4	Alternative Module Signature Key	93
3.7.5	HSM Authentication Key	94
3.7.6	Firmware Encryption Key	95
3.7.7	Audit Log Signature Key	96
3.7.8	Master Backup Key (MBK)	98
3.7.9	Tenant Backup Keys (TBK)	100
3.8	Cryptographic Interfaces	108
3.8.1	Cryptographic eXtended Interface (CXI)	108
3.8.2	PKCS#11	109
3.8.3	Microsoft CryptoAPI and Cryptography API: Next Generation (CNG)	109
3.8.4	Java Cryptography Extension (JCE)	110
3.8.5	OpenSSL	110
3.8.6	Extensible Key Management (EKM)	111
3.9	Supported Algorithms	111
3.9.1	CXI API	111
3.9.2	CSP/CNG API	113
3.9.3	B.1 PKCS#11 API	114
3.9.3.1	PKCS#11 Defined Mechanisms	114
3.9.4	JCE API	119
3.10	Built-in Elliptic Curves	125
4	Setup	128
4.1	System Requirements	128
4.2	Preparations	129
4.3	Installation on Windows	130
4.3.1	Installing the Host Software and CryptoServer Simulator	131
4.3.2	Installing the Host Software and CryptoServer Simulator Without User Interaction	133
4.3.3	Setting up Java Cryptography Extension (JCE) for Windows	138
4.3.4	Installing the PIN Pad Driver	139

4.3.4.1	Prerequisites:	139
4.3.4.2	Installing on Windows 10 and later	140
4.4	Installation on Linux	146
4.4.1	Installing csadm	146
4.4.2	Installing CAT	147
4.4.3	Installing the CryptoServer Simulator	148
4.4.3.1	Prerequisites	148
4.4.3.2	Installation	148
4.4.4	Setting up Java Cryptography Extension (JCE) for Linux	150
4.4.5	Configuring the PIN Pad	151
4.5	Replacing the Default Administrator ADMIN	151
5	Configuration	154
5.1	Setting the CRYPTOSERVER Variable	154
5.2	Using a Local PIN Pad for a Remote CryptoServer	156
5.2.1	Requirements	156
5.2.2	Configuring the PIN Pad Daemon (PPD).....	158
5.2.3	Starting the PIN pad daemon on Windows	163
5.2.4	Starting the PIN Pad Daemon on Linux	167
5.2.5	Connecting a Local PIN Pad on Windows to Remote CryptoServer Tools/APIs on Windows	169
5.2.6	Connecting a Local PIN Pad on Windows to Remote CryptoServer Tools/APIs on Linux.....	172
5.2.7	Connecting a Local PIN Pad on Linux to Remote CryptoServer Tools/APIs on Windows	180
5.2.8	Connecting a Local PIN Pad on Linux to Remote CryptoServer Tools/APIs on Linux.....	183
5.2.9	Syntax for Authenticating a Command of a CryptoServer Tool/API	188
5.3	Load Balancing Mechanism.....	191
5.3.1	Mandatory Preconditions	191
5.3.2	Optional Preconditions	192
5.3.3	Configuration Settings for the Use with CXI	192
5.3.4	Configuration Settings for the Use with PKCS#11	192
5.4	Master Backup Key Rollover.....	192
5.4.1	MBK Rollover Preparations	194
5.4.2	MBK Rollover of an Internal Keystore.....	196

5.4.3	MBK Rollover of an External Keystore.....	201
5.5	Configurable Role-Based Access.....	209
5.6	Interface Hardening by Disabling Selected Functions.....	211
5.7	Configuring HMAC Password Requirements	212
6	CryptoServer Simulator	216
6.1	Physical Protection	216
6.2	Cryptographic Accelerator Chip.....	216
6.3	CryptoServer Series.....	216
6.4	Features Overview	217
6.5	Deliverables	217
6.5.1	The Master Key.....	217
6.5.2	ADMIN, the Default Administrator	218
6.5.3	Serial Numbers and Descriptors.....	218
6.6	Starting the Simulator with bl_sim5.exe.....	218
6.7	Multiple Connections.....	219
6.8	Windows: Using the CryptoServer Simulator	220
6.8.1	Starting Multiple CryptoServer Simulator Instances on Windows.....	221
6.9	Linux: Using the CryptoServer Simulator.....	223
6.9.1	Starting Multiple CryptoServer Simulator Instances on Linux.....	223
6.10	Sensor Detection	225
6.11	Device Address.....	226
6.12	Firmware Module Files	226
6.13	csadm Commands	227
7	Monitoring and Maintenance	228
7.1	Power Supply Monitoring	228
7.2	Temperature Monitoring.....	231
7.3	Logs.....	232
7.3.1	Boot Log	232
7.3.2	Audit Log	233
7.3.2.1	Format of the Audit Log Entries	233
7.3.2.2	Auditable Events.....	234
7.3.2.3	Signed Audit Logs	236
7.3.2.4	Audit Log File Names	236
7.3.2.5	Audit Log Entries	240
8	Troubleshooting	263
8.1	Gathering Diagnostic Information	263

8.1.1	Gathering Diagnostic Information with CAT	263
8.1.2	Gathering Diagnostic Information with csadm	265
8.1.3	Gathering Diagnostic Information with CryptoServer LAN	265
8.1.4	Gathering Diagnostic Information during the Boot Process	265
8.2	Problem Analysis	266
8.2.1	Alarms	266
8.2.2	CryptoServer LAN does not Boot	267
8.2.3	Firmware Package not loading correctly	268
8.2.4	Problem Analysis for CSP/CNG Provider 1.x and 2.x	271
8.2.4.1	Logfile for CSP/CNG Provider 1.x	271
8.2.4.2	Logfile for CSP/CNG Provider 2.x	272
8.2.5	Problem Analysis for EKM	272
8.2.6	Problem Analysis for Java-based GUI Applications	273
8.2.7	Problem Analysis for PKCS#11	274
8.2.8	Problems with the Network	276
8.2.9	The CryptoServer hangs-up during the Boot Phase	278
8.2.10	The CryptoServer is in Maintenance Mode	279
8.3	Providing Support Information for a CryptoServer PCIe	280
8.4	Providing Support Information for a CryptoServer LAN	281
9	Contact Address for Support Queries	283
10	References	284

1 Introduction

Thank you for purchasing our CryptoServer security system. We hope you are satisfied with our product. Please do not hesitate to contact us if you have any questions or comments (see chapter [Contact Address for Support Queries \(p. 283\)](#)).

This manual provides information about the CryptoServer functions and contains solution oriented information about how to set up, manage and maintain the CryptoServer/ CryptoServer LAN all series.

1.1 Target Audience for this Manual

This manual is primarily intended for administrators of the CryptoServer or CryptoServer LAN all series.

1.2 Document Conventions

We use the following document conventions:

<i>Convention</i>	<i>Use</i>	<i>Example</i>
Bold	Items of the Graphical User Interface (GUI), e.g., menu options	Press OK
<code>Monospaced</code>	Code that is given for explanation or as an example, file paths	<code>chsm-create</code>
<i>Italic</i>	References and important terms	See <i>Sample Chapter</i> in the <i>CryptoServer - Sample Manual</i>

Table 1: Document conventions

We use special icons to highlight the most important notes and information.



Here, you find important safety information that should be followed.



Here, you find additional notes or supplementary information.



This message marks the result expected after the successful execution of an instruction.

1.3 Other Manuals

The CryptoServer is supplied as a PCI-Express (PCIe) plug-in card in the following series:

- CryptoServer CSe-Series
- CryptoServer Se-Series Gen2

The CryptoServer LAN (appliance) is supplied in the following series:

- CryptoServer LAN CSe-Series
- CryptoServer LAN Se-Series Gen2

We provide the following manuals on the product CD for all series CryptoServer cards and for all series CryptoServer LAN (appliance):

Quick Start Guides

You will find these Manuals in the main directory of the SecurityServer product CD. They are available only in English, do not cover all possible scenarios, and are intended as a supplement to the product documentation provided on the SecurityServer product CD.

- *CryptoServer LAN V5 - Quick Start Guide*

This guide provides step-by-step instructions on how to bring the CryptoServer LAN into service, how to prepare a computer (Windows 7) for the CryptoServer administration and how to start administrating your CryptoServer with the Java-based GUI CryptoServer Administration Tool (CAT).

- *CryptoServer PCIe - Quick Start Guide for Linux*

This guide provides a step-by-step instruction on how to bring the CryptoServer PCIe card into service, how to install the CryptoServer driver on a computer with minimal RHEL 7.0 installation and how to start administrating your CryptoServer with the CryptoServer Command-line Administration Tool (csadm).

- *CryptoServer PCIe - Quick Start Guide for Windows*

This guide provides for step-by-step instructions on how to bring the CryptoServer PCIe card into service, how to install the CryptoServer driver on a Windows computer and how to start administrating your CryptoServer with the CryptoServer Command-line Administration Tool (csadm).

Manuals for System Administrators

You will find the administration manuals on the product CD in the following directory: ...

Documentation\Administration Guides\

- *CryptoServer – Administration Manual*

This manual provides provides a detailed description of the CryptoServer functionality, concepts and security mechanisms. It also describes CryptoServer setup and maintenance.

- *CryptoServer LAN V5 – Administration Manual*

This manual describes how to administer a CryptoServer LAN appliance by using the front panel of the appliance.

- *CryptoServer - Troubleshooting*

If problems occur during the use of the PCIe card or a LAN (appliance), read this manual.

- *CryptoServer - PKCS#11 P11CAT - Manual*

If you need to administer the PKCS#11 R3 interface with the PKCS#11 CryptoServer Administration Tool (P11CAT), read this manual.

Operating Manual

You will find these manuals on the product CD in the following directory: ...
Documentation\Operating Manuals\ . In these manuals, you find all the necessary information for using appropriately the CryptoServer PCIe card hardware and the CryptoServer LAN (appliance) hardware.

1.4 Abbreviations

Abbreviation	Meaning
AES	Advanced Encryption Standard
BSI	Bundesamt für Sicherheit in der Informationstechnik (Federal Office for Information Security)
CAT	CryptoServer Administration Tool
CNG	Cryptography API: Next Generation
CSP	Cryptographic Service Provider
CXI	Cryptographic eXtended Interface
csadm	CryptoServer command-line administration tool
DES	Data Encryption Standard
DRBG	Deterministic Random Bit Generator
DRNG	Deterministic Random Number Generator
DSA	Digital Signature Algorithm
DSN	Data Source Name
ECDSA	Elliptic Curve DSA
EKM	Extensible Key Management
FIPS	Federal Information Processing Standard
HSM	Hardware Security Module
JCE	Java Cryptography Extension
JRE	Java Runtime Environment
MAC	Message Authentication Code
MBK	Master Backup Key
NTP	Network Time Protocol
ODBC	Open Database Connectivity

<i>Abbreviation</i>	<i>Meaning</i>
P11CAT	PKCS#11 CryptoServer Administration Tool
PCIe	PCI Express Interface
PRNG	Pseudorandom Number Generator
RSA	Rivest, Shamir, Adleman (cryptosystem)
TRNG	True Random Number Generator
UCAPI	Unified Container API

2 Overview

The CryptoServer is a hardware security module (HSM) that was developed specifically to ensure the efficient and secure performance of the following cryptographic operations:

- Generate key
- Save the key securely
- Generate random numbers (hardware (true) and software (pseudo) random number generator)
- Generate and verify signatures
- Encrypt and decrypt data
- Calculate hash values

CryptoServer protects all cryptographic operations or keys you use against any form of attack. To do so, it uses technical software solutions, and physical measures to shield against attacks. The CryptoServer therefore guarantees the trustworthiness and integrity of data within your IT systems.

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.

	Mode I	Cryptographic Accelerator Chip	LAN Appliance	PCIe	
CryptoServer CSe-Series	CSe1 00	---	✓	✓	CryptoServer CSe-Series is designed for the highest requirements regarding physical security, as e.g. required in banking and governmental environments. Its extensive sensor mechanism, which includes the usage of a sensor-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 area "Physical Security", and is certified according to this security standard.
	CSe1 0	---	✓	✓	
CryptoServer Se-Series Gen2	Se15 00	Exar (for RSA and ECC operations)	✓	✓	CryptoServer Se-Series Gen2 has been designed to meet the requirements of FIPS 140-2 level 3 in the area "Physical Security". CryptoServer Se-Series Gen2 may be delivered with a cryptographic accelerator chip which provides highest performance for RSA and ECC operations.
	Se50 0	Exar (for RSA and ECC operations)	✓	✓	
	Se52	---	✓	✓	
	Se12	---	✓	✓	

- CryptoServer as a **plug-in card with a PCI Express bus**, is referred to as **CryptoServer**.
- CryptoServer **LAN as a network component (appliance)**, which can be easily integrated into a network, is referred to as **CryptoServer LAN**.

2.1.1 IRAM and Key-RAM

On the CryptoServer PCIe 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:

- **IRAM**

The CPU of the CryptoServer 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 sensor (each memory cell of the IRAM will be overwritten), see *Alarm*. If the CryptoServer goes down on power-off, the IRAM will be erased, too.

- **Key-RAM**

The Key-RAM is a non-volatile RAM, which is protected by the sensors. is used to store the Master Key because the sensor-protection holds on runtime and in retention that is, if power supply is switched off. This Master Key is only used on the CryptoServer to encrypt other keys and sensitive data that are stored on the CryptoServer, see *Master Key*. Independently from the mode of operation the mechanism, an internal power supply provides in any case enough power to detect an attack by the sensors, initiate an alarm and erase the Key-RAM including the Master Key actively within less than two milliseconds. Within this timeframe, the Key-RAM will be overwritten five times, alternately with 0x00 and 0xFF patterns. As a consequence, none of the other keys and sensitive data on the CryptoServer can be decrypted.

2.1.2 Random Number Generator (RNG)

The CryptoServer implements a hardware true random number generator (TRNG) and a software pseudorandom number generator (PRNG), which is also called deterministic random bit generator (DRBG).

The hardware random number generator is needed to produce real respectively true random numbers. It is implemented by using a physical noise generator, the output of which is mathematically post-processed and online tested to guarantee the quality of the generated random numbers.

CryptoServer's TRNG complies with class PTG.2 of a TRNG as specified in the BSI specification AIS 20/AIS 31. The generated random numbers can be used for the generation of signature key pairs, random padding bits, etc.

To achieve high-performant random number generation, the TRNG is enhanced by a software random generator (DRBG) which complies with class DRG.4 of DRNG as specified in the BSI specification AIS 20/AIS 31. The algorithm used by this software is specified in NIST 800-90A "Recommendation for Random Number Generation Using Deterministic Random Bit Generators". The generated pseudorandom numbers can be used for the generation of keys, padding bits, etc.

2.1.3 Physical External Housing

All hardware components are physically protected and securely sealed in place. The CryptoServer and the keys and data it contains are therefore fully protected against any attack. Any attempt to break into this physical external enclosure, or to degrade any of its surfaces, will leave unmistakable traces of the attack that cannot be disguised. Any attempt

to remove the physical external enclosure will destroy the CryptoServer's internal components.

2.1.4 Sensors

In addition to its physical external enclosure, the CryptoServer is equipped with a range of sensors that constantly monitor its critical operating parameters. In particular, these sensors check whether the device is running within the upper and lower threshold values for its operating voltage and operating temperature. If temperature or voltage falls below or rises above these threshold values, the sensors react as if the device is being attacked and the CryptoServer's internal key memory is deleted. This prevents any keys and sensitive data from being read-out as a result of unauthorized operating parameters.

An internal power supply (battery) ensures that, even when the CryptoServer is switched off, it has sufficient power to operate the sensors and to delete sensitive data or memory if necessary.

2.1.5 Tamper-protecting Foil

In the CryptoServer CSe-Series device, a tamper-protecting foil is connected to the sensors. If the physical external enclosure was under a mechanical or chemical attack, which damages the tamper-protecting foil, the sensors trigger an alarm which immediately deletes all keys and sensitive data on the CryptoServer.

When an alarm is triggered, an entry is recorded in the CryptoServer's audit log file.



An alarm triggered because of a damaged or destroyed tamper-protecting foil, is a permanent alarm. The CryptoServer cannot be brought back into operation until the tamper-protecting foil has been repaired or replaced by Utimaco IS GmbH.

After a restart, the CryptoServer goes into Maintenance Mode. This allows you to check the status information and log files to analyze the problem more closely before sending the CryptoServer device to Utimaco IS GmbH for repair.

2.1.6 CryptoServer Batteries

CryptoServer CSe/Se-Series Gen2

The CryptoServer PCIe card is equipped with one battery - Carrier Battery - placed on its carrier that ensures that the sensors of the hardware security module are always able to function correctly, and that no security-critical information is lost or deleted, even if the computer

where the CryptoServer PCIe card is mounted in, is switched off. If the device is not used over a prolonged period, i.e., during storage or if the computer, where the device is mounted in, 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.

For a step-by-step instructions how to exchange the carrier battery depending on the CryptoServer series, see *Replacing the Battery* of the respective Operating Manual.

CryptoServer LAN CSe/Se-Series Gen2

The CryptoServer LAN is equipped with two batteries to ensure that no security-critical information is lost or deleted on the hardware security module when the device is switched off, or if operation is interrupted due to a power failure.

- The Carrier Battery is placed on the carrier of the mounted CryptoServer PCIe card. If the CryptoServer PCIe card is not used over a prolonged period, i.e., during storage or if the CryptoServer LAN, where it is mounted in, is turned off, this 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.
For a step-by-step instructions how to exchange the carrier battery depending on the CryptoServer series, see *Replacing the Battery* of the respective Operating Manual.
- The External Battery is located in the battery compartment of the CryptoServer LAN. 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 1.5 years at a minimum.
For a step-by-step instructions how to exchange the external battery of the CryptoServer LAN, see *Replacing the External Battery* of the respective Operating Manual.



These batteries are not rechargeable. The state of the batteries must be checked regularly. In case both batteries are exhausted, an alarm is triggered and all sensitive data stored on the CryptoServer is deleted.

2.2 Software

Different software components work together on the CryptoServer at different times. This section gives a brief introduction to these software components and describes their functionality.

2.2.1 Firmware Modules

This section describes the firmware that is normally running on a CryptoServer; SMOS and the firmware modules.



A software module or a firmware module in the context of this documentation denotes an encapsulated software part running inside the CryptoServer. A module can have an external interface that can be used by an application from outside the CryptoServer device, and an internal C interface that can be called by other firmware modules.

In addition to the actual SMOS (Security Module Operating System) operating system there are a number of other firmware modules each performing specific functions, for example:

- Generating random numbers
- Signature generation
- Data encryption (RSA, elliptic curves cryptography, DES, AES, etc.)
- Performing hash functions
- Access to the internal real-time clock
- Storing keys and other data
- Communicate with the PIN pad and smartcard, etc.

Most of these firmware modules are loaded, updated, and deleted independently of each other. When you upgrade a firmware module it is ensured that any keys and data already present, are transferred automatically and therefore do not need to be imported from scratch.

The CryptoServer is delivered with firmware modules loaded (firmware package of the SecurityServer) and stored as `*.msc` files. Additionally, during production at the manufacturer's site, a subset of these firmware modules is loaded into the CryptoServer as so-called system firmware modules. (For this a specialized `LoadFile` Bootloader command will be used which stores the firmware modules as `*.sys` files (system firmware modules).

The Bootloader command `LoadFile` is not available for any CryptoServer after delivery, i.e., it is not available at the customer's site.) The purpose of these system firmware modules is to provide a backup copy of every system-relevant firmware module.



Even if the operating system or other firmware modules is later on deleted or replaced by newer versions, or if the CryptoServer is cleared by the customer, all system firmware modules (`*.sys`) will be preserved.

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.

Additionally, to these system firmware modules, which are always loaded inside the CryptoServer and can be used as a backup system of firmware modules for emergency cases, a complete set of all wanted firmware modules should be loaded. This may include Utimaco's standard firmware modules including specialized application-specific firmware modules (e.g., CXI) or further customer-specific firmware modules, which may provide the actual CryptoServer functionality through an appropriate external interface, for example, for key management, cryptographic algorithms or time stamping.



In general, the CryptoServer stores a firmware module in form of an MSC (Module Storage Container, `*.msc`). The MSC format is for CryptoServer-internal use only. It stores the raw binary firmware module together with certain module information and the check value for the raw binary firmware module's integrity (SHA-512 hash value over the raw binary firmware module, which is verified at every startup). Firmware modules that are loaded 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 an MSC, just with another extension `*.sys` . These system firmware modules cannot be deleted or updated and thereby offer anytime a fallback possibility for firmware.



In addition to the standard firmware modules, which are provided by Utimaco, it is possible to load further firmware modules into the CryptoServer



The functional design of each standard firmware module and its interface is specified in more detail in the respective module-specific documentation. The specifications are only available for owners of Utimaco's Software Development Kit (SDK) and require the signing of a non-disclosure agreement.

2.2.1.1 Firmware Module Management

This chapter describes the management of the firmware modules of the CryptoServer from a security point of view.

With the aim to link the firmware with administrative information (for example, 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 load 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 loading. Please see chapter [Package Files](#) (p. 24), of this manual for the concept and usage of these firmware packages.

See also section *Commands for Firmware Management* in *CryptoServer - csadm Manual* and *Installing/Updating the Firmware - CryptoServer - CAT Manual*.

2.2.1.1.1 Firmware Containers: MTC/MMC, MSC, SYS

A raw binary firmware module is the executable module compiled for the CryptoServer platform, i.e., ready for interpretation of the CryptoServer operating system SMOS. The raw binary firmware module can be in COFF format (`*.out`), as used by the CryptoServer hardware, in DLL format (`*.dll`), as used by the CryptoServer Simulator/SDK (Software Development Kit) for Windows or in ELF format (`*.so`) as used by the CryptoServer Simulator/SDK for Linux. The raw binary firmware module is also called raw firmware module (RFM), binary or module code.

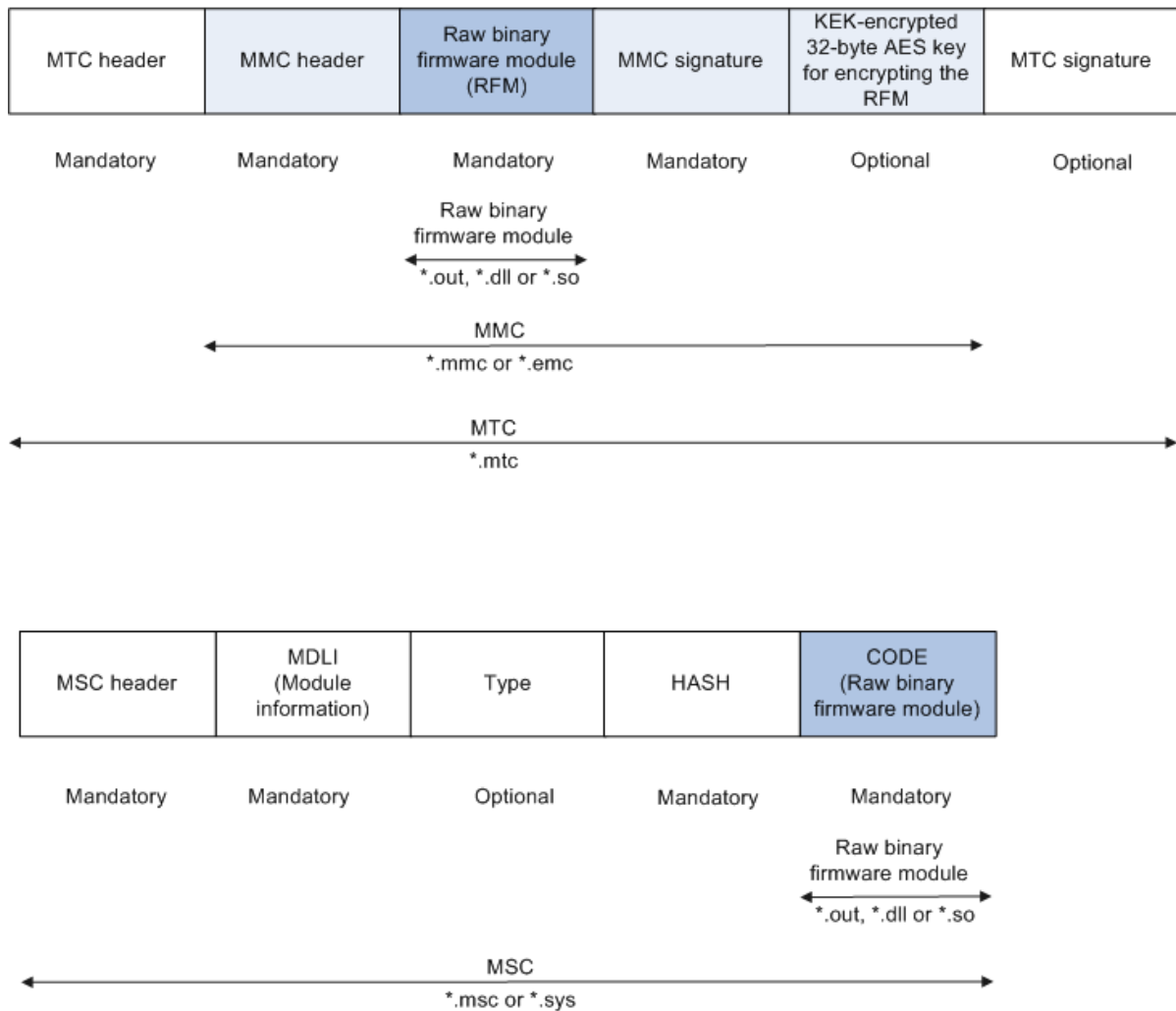


Figure 1 : RFM, MMC, EMC, MTC, MSC and SYS

For various purposes (firmware transport, loading, and storage inside the CryptoServer) this RFM is enveloped in so-called firmware module containers.

- MMC

For the RFM to be loaded into the CryptoServer, the RFM is enveloped into an MMC (Module Manufacturer Container).

The MMC adds a header with module information to the RFM and an MMC signature, which guarantees for the integrity and authenticity of the RFM. The MMC signature has to be calculated with the private part of either Utimaco's Module Signature Key (see section [Module Signature Key \(p. 92\)](#)) if the RFM has been written by Utimaco, or with the private part of the customer's Alternative Module Signature Key (see section [Alternative Module Signature Key \(p. 93\)](#)) if the RFM has been written by the customer. This

signature is calculated over the MMC header and the RFM. This signature is verified during firmware loading. An MMC is usually stored as an `*.mmc` file.

The MMC header, the RFM and the MMC signature are mandatory. The 32-byte AES key is optional. It is only needed if the RFM is encrypted (see section [Encrypted Firmware Modules](#) (p. 23), for details). In this case and under certain circumstances, the MMC is not stored as usual as an `*.mmc` file but as an `*.emc` file.

- MTC

The MMC is enveloped into an MTC (Module Transport Container), which is used to secure the transport of the RFM from the developer to the customer and in this way guarantee the authenticity of the RFM during delivery. The MTC adds an MTC header, which contains additional module transport information, to the MMC and an MTC signature for secure and authentic transport. The MTC signature is calculated over the complete MMC. The MTC header is mandatory but the MTC signature is optional. An MTC is stored as an `*.mtc` file.

- MSC

After the MTC has been transported to and loaded into the CryptoServer, the CryptoServer unwraps the RFM from the MMC/MTC and stores it as an MSC (Module Storage Container).

The MSC format is for CryptoServer-internal use only. It stores the RFM together with certain module information and the check value for the RFM's integrity (SHA-512 hash value calculated over the RFM). All sections of an MSC except the type section are mandatory.

Perform the `csadm ListFiles` command to show all available `*.msc` files on a CryptoServer.

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 as MSC files but they are stored with file extension `*.sys`. `*.sys` files cannot be deleted and are never updated, even if the CryptoServer is updated. Perform the `csadm ListFiles=SYS` command to show all available `*.sys` files on a CryptoServer.

To show the contents of a firmware module (an `*.out` file, a `*.dll` file, a `*.so` file, an `*.mmc` file or an `*.mtc` file), perform the `csadm ModuleInfo` command. See section *ModuleInfo* in *CryptoServer - csadm Manual*, for details.

2.2.1.1.2 Encrypted Firmware Modules

Optionally, firmware modules may be loaded and stored inside the CryptoServer in an encrypted way.

This procedure requires that the private part of an appropriate RSA key, the Firmware Encryption Key, is imported into the CryptoServer. See section "[Firmware Encryption Key \(p. 95\)](#)", for details. For importing the private part of the Firmware Encryption Key, the `csadm LoadFWDecKey` command is used, which needs to be authenticated with administrative user rights (permission 2 in the user group 6, see section `LoadFWDecKey` in the *CryptoServer - csadm Manual*). The private part of the Firmware Encryption Key is stored in an appropriate key database, `fw_dec_key.db`, inside the CryptoServer and is encrypted with the Master Key.

Outside the CryptoServer, the public part of the Firmware Encryption Key is now used as a Key Encryption Key (KEK) for a 32-byte AES key. The raw binary firmware module itself is encrypted with this AES key. The AES-encrypted firmware module, the KEK-encrypted AES key (encryption according to PKCS#1 block 02) and some key type information are embedded in the MMC structure.

During the load procedure of a firmware module, the CryptoServer searches the MMC structure for the field with the encrypted AES key. If the field is found and the Firmware Encryption Key is already stored, CryptoServer uses its private part to decrypt the AES key. After that the CryptoServer decrypts the firmware module with the AES key and checks its integrity. Before the firmware module is stored (now as an MSC) inside the CryptoServer it is encrypted again, this time with the Master Key. A flag in the MMC header indicates that the contained firmware module is encrypted.

All other internal steps of the firmware load (for example, checking module integrity with SHA-512 hash) are the same as described above. If an encrypted firmware module `*.mtc` is loaded into the CryptoServer but the Firmware Encryption Key is not yet stored inside the CryptoServer, the firmware module is stored in the FLASH file system with the file extension `*.emc` for later decryption, as part of the `LoadFWDecKey` command.

2.2.1.2 Firmware Package

The SecurityServer product CD supplied along with every CryptoServer device contains all firmware modules, provided as firmware packages. You can easily identify these packages by their `.mpkg` file extension in the `\Firmware\SecurityServer-<CryptoServer-Series>` directory.

For the different series of CryptoServer, Se-Series Gen2 and CSe-Series, as well as for the CryptoServer Simulator and the FIPS 140-2 Level 3-certified CryptoServer Se-Series Gen2, the corresponding SecurityServer firmware packages are provided.

To simplify the firmware management, Utimaco offers firmware also in the so-called package files. A `*.mpkg` package file can contain several firmware modules (for example, MTCs) as well as other files in a packed form.



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

For this purpose, the CryptoServer Administration Tool csadm offers the `csadm LoadPkg` command, which can replace a series of succeeding csadm commands. The `csadm LoadPkg` command loads the contents of the given `*.mpkg` package file into the CryptoServer, adding to or replacing existing firmware.

Furthermore, the csadm tool offers the following features:

- Create a package file from a collection of MTCs:
`csadm Pack` command
- List the contents of a package:
`csadm ListPkg` command
- Retrieve the contained firmware modules from a `*.mpkg` file:
`csadm Unpack` command
- Compare the contents of a package file with the firmware that is already stored on the CryptoServer:
`csadm CheckPkg` command
- Check the MMC signature of each firmware module contained in a package:
`csadm VerifyPkg` command

For details see sections *Commands for Firmware Management* and *Commands for Firmware Packaging* in *CryptoServer - csadm Manual*.

2.2.2 Signed License Files



The Signed License File concept is used to regulate the availability of certain features and performances (for instance the speed of certain cryptographic algorithms) in a customer-specific way.

These regulations do not cause any security-relevant changes to the software.

A Signed License File (SLF) is a customer-specific license file `*.slf` which is generated by Utimaco and signed by Utimaco's Module Signature Key (RSA signature).

The SLF can be loaded with the `LoadFile` command. During loading the signature is verified, and a check value for the SLF's integrity (SHA-512 hash value over the SLF) is calculated. The SLF is stored together with this check value which replaces the signature. With every startup the operating system SMOS searches for the SLF and verifies its hash value during the-boot process.

If a SLF is missing or its integrity cannot be verified, the firmware modules on the device 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 device will for instance perform certain algorithms only with lowest speed.

2.2.3 CryptoServer Boot Process

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

- First boot phase, which is controlled by the bootloader
- Second boot phase, which is controlled by the operating system SMOS

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. (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.) This is the bootloader firmware code. The bootloader does the first necessary start procedures, including self-tests, and offers some basic commands like status queries.

At the end of the bootloader-controlled boot phase there is an 3 seconds time window for entering commands.

If the bootloader does not receive any command, and if the operating system module SMOS is loaded and its integrity can be verified (SHA-512 hash value), then the bootloader starts SMOS automatically. If this was successful, the CryptoServer is considered to be in Operational Mode (or Maintenance Mode) and the bootloader terminates itself (see also [CryptoServer Modes and States \(p. 65\)](#)).

After the bootloader has passed the control to the operating system, SMOS roughly performs the following steps:

1. Initialize the memory.
2. Initialize the flash file system.
3. Initialize all hardware peripherals (including serial, PCIe and USB ports).
4. Search 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.
5. Verify the integrity of all firmware modules.
6. Start all firmware modules.

2.2.3.1 Manually Starting SMOS

The operating system of the CryptoServer (firmware module SMOS) can be started in two ways – either in the regular way or by starting the failsafe backup copy of SMOS:

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

Both starting methods can also be explicitly chosen by the user if the respective external CryptoServer command is performed:

- The command `StartOS` (see also section *Commands for Administration* in *CryptoServer - csadm Manual*) corresponds to the usual way of starting SMOS (i.e., starting `smos.msc`) whereas
- The command `RecoverOS` corresponds to the start of the backup copy of SMOS (i.e., start of the system module `smos.sys`).



If starting SMOS is successful, the CryptoServer enters Operational or Maintenance Mode (see below), and the bootloader terminates.



If the OS start fails, then the CryptoServer remains in Bootloader Mode. The bootloader is active and further bootloader commands (e.g. status queries) can be performed.

2.3 Administration

The CryptoServer is administered with the GUI CryptoServer Administration Tool (CAT) and with the Command Line Administration Tool (csadm). The CryptoServer LAN can additionally be administered with the front-panel menu.

While most functionalities can be performed both with CAT and csadm, there are some exceptions. The following table provides an overview of functionalities that are either only available with one tool or differ across tools.

Functionality	CAT	csadm	Comment
Setting maximum authentication failures	✗	✓	Only possible with the <code>csadm SetMaxAuthFail s</code> command
Enabling and disabling the <i>Administration Only mode</i>	✗	✓	Only possible with the <code>csadm SetAdminMode</code> command
Enabling and disabling the <i>Administration Only mode</i> upon startup	✗	✓	Only possible with the <code>csadm SetStartupMode</code> command
Getting the startup mode	✗	✓	Only possible with the <code>csadm GetStartupMode</code> command
NTP management	✓	✗	Only possible with CAT at Manage > NTP Settings
Generating an RSA Key Backup Share or an Elliptic-Curve Cryptography Key Backup Share on Smartcards	✓	✗	
Copying of a user authentication key backup shares from one smartcard to another smartcard	✓	✓	For CAT, the RSA key backup share and the ECDSA key backup share are copied at the same time. For the <code>csadm CopyBackupCard</code> command, only the share specified by the <code><keytype></code> parameter is copied.

Functionality	CAT	csadm	Comment
Display of audit log entries	✓	✓	<p>The <code>csadm GetAuditConfig</code> command returns the activated audit log classes as a bit interface. The CAT display at Manage > Audit Log Settings displays the full audit log event names. The audit log entries displayed in CAT can be incorrect when one of the following values has been set with the <code>csadm SetAuditConfig</code> command:</p> <ul style="list-style-type: none"> 0x7f000000: Message class number 25-30 (Customer-defined audit message classes) 0x80000000: Message class number 13-24 (Audit message classes reserved for future use) 0xffffffff: Equal to OS_AUDIT_CLASS_ALWAYS

Table 2: Functionalities of CAT and csadm

2.3.1 CryptoServer Administration Tool (CAT)

The CryptoServer administration tool is a Java application used to administer CryptoServer PCIe cards as well as CryptoServer LANs. Use the CAT to perform all tasks to bring the CryptoServer into operation, to monitor its status and manage users, firmware and keys.

In addition, the CAT offers functions that can be implemented for key files and smart cards without the involvement of the CryptoServer.

All host operating systems supported for the use of CAT are listed in the [CS_PD_SecurityServer_SupportedPlatforms.pdf](#) on the SecurityServer/CryptoServer SDK product CD in the `...\Documentation\Product Details` directory.

The CAT runs in the Java runtime environments (JRE) 11 and 14/15.

2.3.2 CryptoServer Command-line Administration Tool (csadm)

The CryptoServer command-line tool `csadm` can be called from either a command line or a batch file. You can use `csadm` to administer CryptoServer PCIe cards and CryptoServer LANs. With `csadm` you can perform the same administrative tasks as with CAT. You can perform additional, advanced administration functions that are mainly of interest to customers who want to expand the standard functionality of the CryptoServer with self-developed firmware modules providing specific cryptographic functions and commands. For Details, see the *CryptoServer – csadm Manual*.

All host operating systems supported for the use of `csadm` are listed in the [CS_PD_SecurityServer_SupportedPlatforms.pdf](#) on the SecurityServer/CryptoServer SDK product CD in the `...\Documentation\Product Details` directory.

2.3.3 CryptoServer LAN Menu Options

You can use the display and the control keys on the front of the CryptoServer LAN to access the menu options for the most important administration functions. They include commissioning and status monitoring, as well as a limited range of functions for managing firmware and keys. However, you cannot use these menu options to manage user data.

With these menu options you can administer both the CryptoServer LAN and the CryptoServer PCI card built into the CryptoServer LAN.

2.3.4 Scenarios

Some interactions with the CryptoServer include the use of a private or public RSA or ECDSA key to sign a command or a file, or to verify a signature.

One option to do so is to use RSA/ECDSA keys that are stored on a smartcard. The smartcard must be inserted into a PIN pad, and this PIN pad must be directly connected to the computer the CryptoServer administration tools are running on. The CryptoServer offers as well the possibility to connect the PIN pad to a local computer while the CryptoServer and its tools are running on a remote computer.

The following subchapters give an overview of the possible setup scenarios. The setup of a PIN pad for a remote CryptoServer is described in detail in [Using a Local PIN Pad for a Remote CryptoServer \(p. 156\)](#).

The PIN pad daemon is a TCP server that provides access to a PIN pad over the network. Basically it serves remote procedure calls from a remote PIN pad API and dispatches them to a local PIN pad API. It can use secure messaging to encrypt messages to the commands.

The default port number has been set to 6070 but it can be changed in the configuration file.

2.3.4.1 Scenario 1: Local CryptoServer Tools/API, Local CryptoServer PCIe

The simplest scenario is a CryptoServer PCIe mounted in an administration computer. The CryptoServer tools and APIs, for example, csadm, are running on this computer, and the PIN pad is directly connected to a USB port of this computer or to a USB port of the CryptoServer PCIe card.

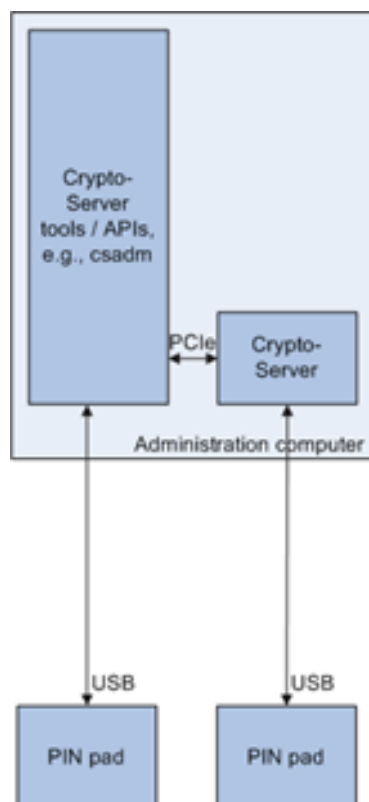


Figure 2 : PIN pad access to a local CryptoServer PCIe

2.3.4.2 Scenario 2: Local CryptoServer Tools/API, Remote CryptoServer LAN

In this scenario, the CryptoServer tools and APIs are installed as before on a local administration computer but the CryptoServer is mounted in a remote CryptoServer LAN. They are connected via a local area network. Commands that are triggered by the CryptoServer tools/APIs can be authenticated by using a PIN pad that is directly connected to the administration computer via a USB port. However, if actions are performed by using the front panel of the CryptoServer LAN, the PIN pad must be directly connected to a USB port of the CryptoServer LAN, see [CryptoServer LAN V5 - Operating Manual. \(p. 284\)](#)

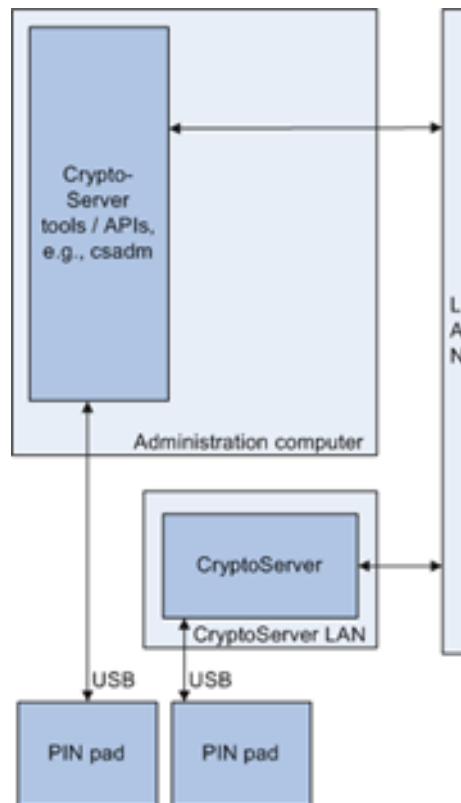


Figure 3 : PIN pad access to a CryptoServer LAN

2.3.4.3 Scenario 3: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN Without Blocking Firewall

In this scenario, the PIN pad is directly connected to a local computer. The CryptoServer tools/APIs, for example, csadm, are installed on a remote computer. Both computers are connected to each other by a local area network. There is no firewall between these computers. You have physical access only to the local computer, that means for example that any input via a keyboard must be done here. The local computer must provide a method to forward the keyboard input to the remote computer, for example, by using an SSH connection or a remote desktop connection. For example, an SSH client on the local computer opens an arbitrary port on this computer and connects it via the LAN to the remote computer where an SSH daemon listens to port 22 and forwards the information to csadm. That is, you open a command-line on the local computer, and in this command-line, you perform a csadm command on the remote computer.

This csadm command communicates either with a CryptoServer on a PCIe card in the same computer or via the local area network with a CryptoServer LAN.

If this csadm command has to be authenticated by a key that is available on a smartcard in a PIN pad that has been connected to the local computer, csadm opens an arbitrary port on the remote computer and connects it via the local area network to port 6070 (default value)

on the local computer where a PIN pad daemon listens to this port 6070. This daemon communicates with the PIN pad via a USB port.

The operation system on the local computer may differ from the one on the remote computer, for example, Windows on the local computer and Linux on the remote computer.

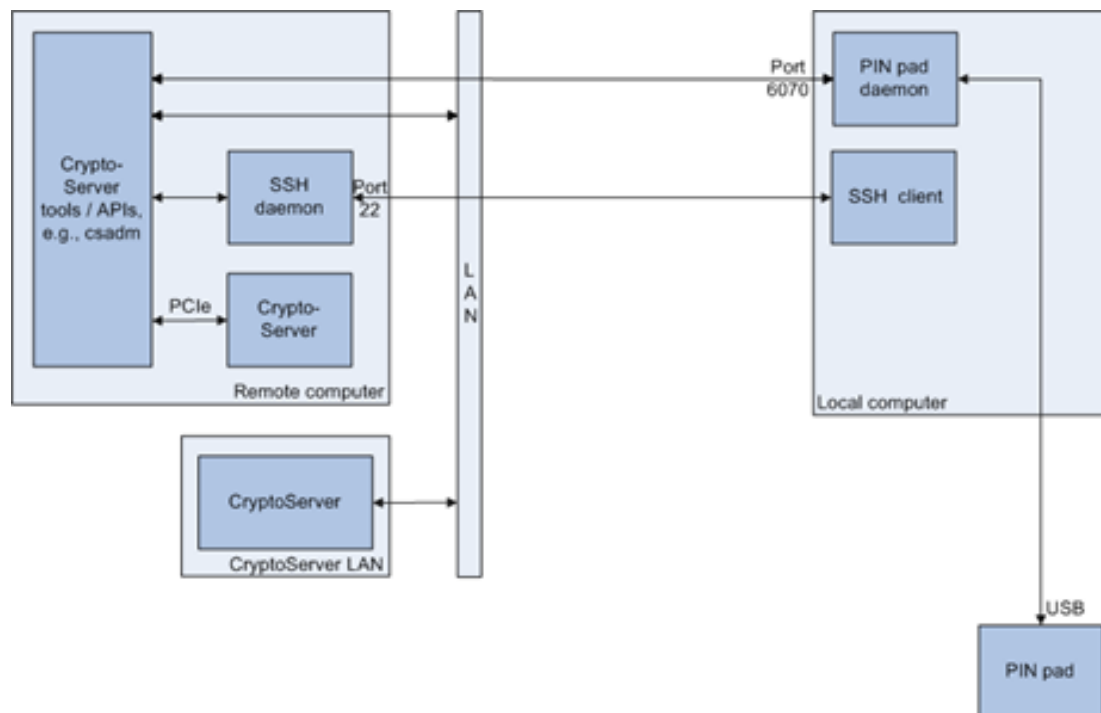


Figure 4 : PIN pad access to a remote CryptoServer without a blocking firewall

2.3.4.4 Scenario 4: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN With Blocking Firewall

If there are one or more firewalls between the local computer and the remote computer that block port 6070, a setup as described in this scenario must be used.

In this scenario, the PIN pad is directly connected to a local computer. The CryptoServer tools and APIs, for example, csadm, are installed on a remote computer. Both computers are connected to each other by a local area network.

You have physical access only to the local computer, that means for example that any input via a keyboard must be done here. The local computer must provide an SSH client on the local computer. No remote desktop connection or anything similar can be used. The SSH client opens an arbitrary port on this computer and connects it via the LAN to the remote computer where an SSH daemon listens to port 22 and forwards the information to csadm. That is, you open a command-line on the local computer, and in this command-line, you perform a csadm command on the remote computer.

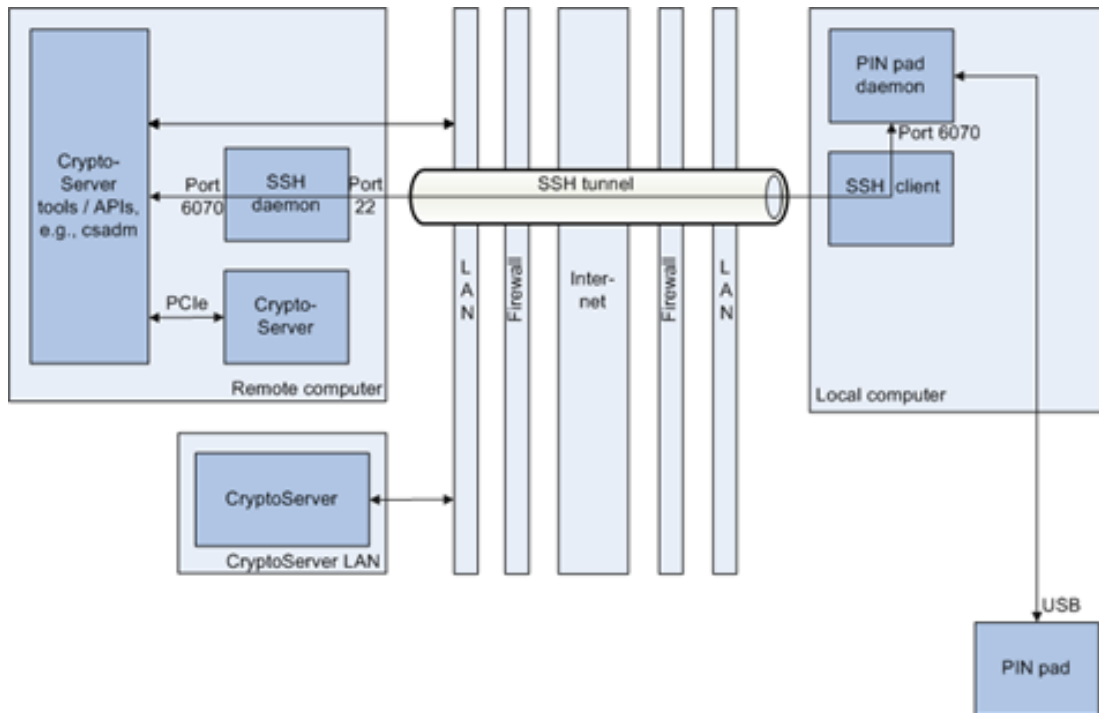


Figure 5 : PIN pad access to a remote CryptoServer with a blocking firewall

2.3.4.5 Scenario 5: Mixed Scenarios

The scenarios described above might be combined with each other, for example, one PIN pad is connected to the local computer and another one is for example connected to the remote computer. This might be suitable if a command must be authenticated by more than one user (two-person rule). The following figures show two examples.

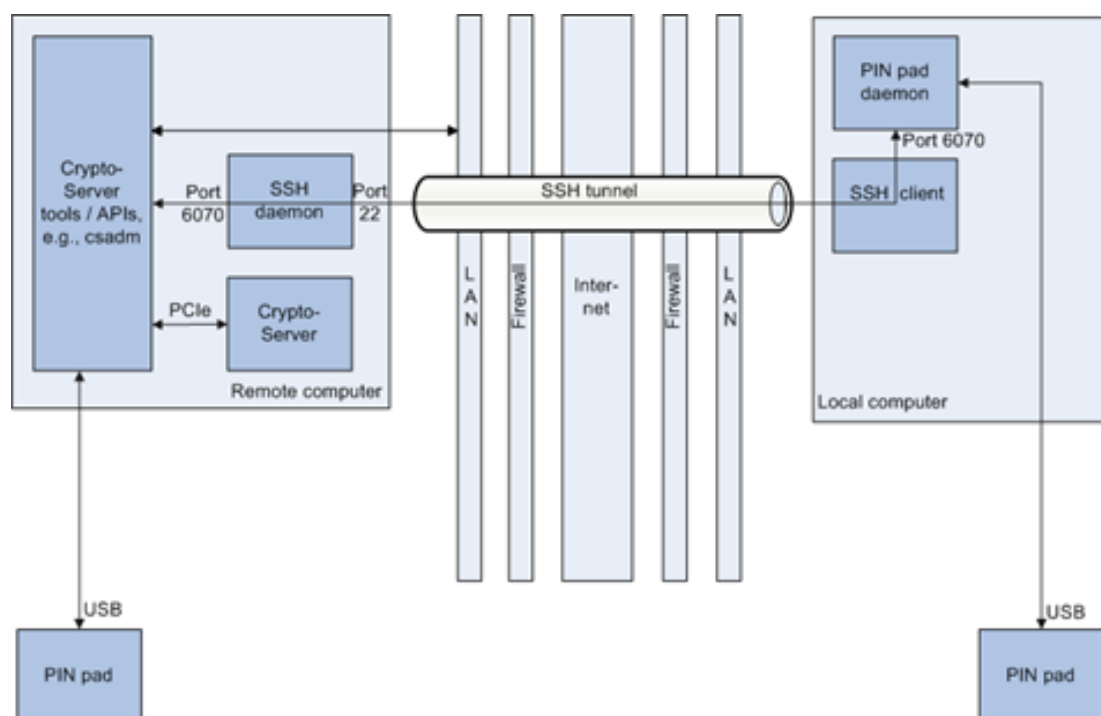


Figure 6 : An example for a mixed scenario

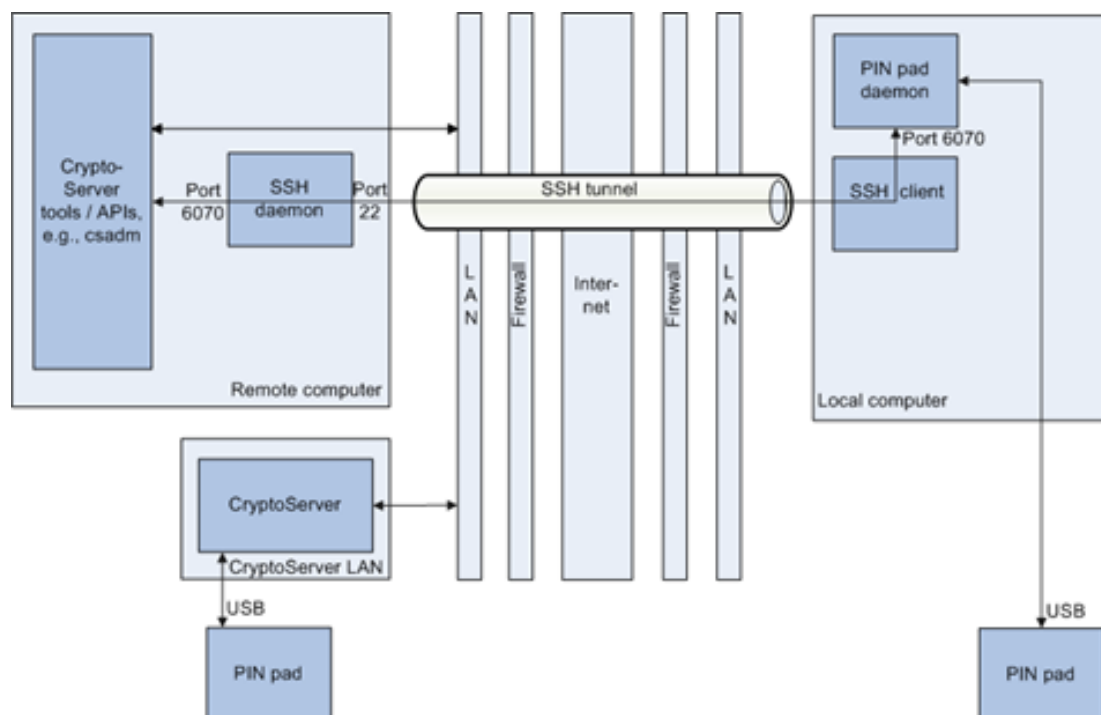


Figure 7 : An example for a mixed scenario

The mixed scenarios are not described in more detail.

3 Concepts

3.1 Security Mechanisms

3.1.1 Secure Messaging

The device supports Secure Messaging for communications between the host and itself. Commands from the host to the device and the replies to the host can be AES encrypted and the integrity of the data can be protected by a AES MAC (Message Authentication Code). For this purpose, a secure messaging header data block is added to the command and the answer data block. The detailed structure of the header data blocks is described in *CryptoServer - Firmware Module CMDS - Interface Specification* provided within the product bundle.

In Secure Messaging between the device and the host, a new random session key is generated for each connection. This prevents recorded data packets belonging to an earlier connection from being imported into a new connection. No valid data packets can be present once the imported data packet has been decrypted with the current session key.

In addition, the device generates a start value for a sequence counter and a session ID for every new session key. The sequence counter increases every time a command is sent and is also included in the MAC calculation or integrity check. This ensures that no commands are recorded within the same connection and sent to the device again. The unique session ID guarantees that every session key is uniquely identifiable. All commands that use the same session key and session ID are part of the same session (connection).



The SecurityServer/CryptoServer SDK 4.10 and later support a maximum of 4096 session keys (connections) active at the same time. If another session key is requested for a connection, exceeding the maximum of 4096, the oldest connection is closed and the session key, associated with it, becomes invalid.

The SecurityServer/CryptoServer SDK 4.01 and earlier support a maximum of 256 session keys which are active at the same time and used by different host applications (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 the session key, associated with it, becomes invalid.

For a detailed description of the Secure Messaging process, see *Secure Messaging* in the [CryptoServer - csadm Manual \(p. 284\)](#).

3.1.2 Mutual Authentication

Mutual authentication is a security feature introduced with the SecurityServer SDK 4.10. It requires that both communicating parties, i.e., the host application and the CryptoServer device, prove their respective identities to each other before performing any application functions. Security-relevant commands are sent to the CryptoServer by host applications, within a Secure Messaging session, that must be always authenticated by one or more users with appropriate permissions. The CryptoServer does not authenticate itself to the host applications by default. It can be configured to do so as described in *Enabling Mutual Authentication* in the [CryptoServer - csadm Manual \(p. 284\)](#).

For the CryptoServer to be able to authenticate itself to request host applications, the concept of the HSM Authentication Key has been introduced, see [HSM Authentication Key \(p. 94\)](#). This is a device-individual 3072-bit RSA key pair, automatically generated on request and owned by the CryptoServer. Once created, the key cannot be modified or deleted by any CryptoServer user. It is used by the CryptoServer to authenticate itself in the initialization phase of a Secure Messaging session if requested by the respective host application. Both the private and the public part of the HSMAuthKey are securely stored inside the CryptoServer's FLASH memory in the `authkey.db` database. The private part of the HSMAuthKey is encrypted with the Master Key and cannot be exported outside the HSM. The public part of the key can be retrieved with the dedicated `csadm` command `GetHSMAuthKey`, and exported into a file, for example, `HSMauth.key`, maintained by the CryptoServer administrator. Thus, it can be used to verify the authenticity of a Secure Messaging session originating from the HSM. The HSMAuthKey resp. the `authkey.db` is deleted when an alarm has occurred or a Clear command has been executed. A new HSM Authentication Key is generated the next time the `csadm GetHSMAuthKey` command is executed.

3.1.3 An Alarm and Its Consequences

An alarm state is caused by certain extraordinary physical circumstances.

Anytime a physical alarm occurs on the CryptoServer, it is detected immediately by the sensor, which triggers the defined alarm mechanism. Part of this mechanism is a restart of the CryptoServer. An alarm is given if the operating system SMOS detects an alarm condition in the alarm status register during the boot process. The CryptoServer responds with the current alarm condition the alarm must be reset in the alarm status register, see *AlarmTreatment* in the [CryptoServer - csadm Manual \(p. 284\)](#) and *Resetting an Alarm* in the [CryptoServer - CAT Manual \(p. 284\)](#).

Two different kinds of alarms are possible:

- Temporary alarms. These alarms can be reset.

- Permanent alarms. These alarms cannot be reset.

Permanent alarms occur in case of a damage of the inner or outer tamper-protecting foil, whereas usually all other possible alarms are temporary.

Abbreviation	Description
Temp_low	Temperature too low (see Temperature Monitoring (p. 231))
Temp_high	Temperature too high (see Temperature Monitoring (p. 231))
	Voltage too high (CryptoServer-internal battery)
Pow_low	Voltage too low (CryptoServer-internal battery)
In_foil	Inner tamper-protecting foil damaged. This alarm reason is only relevant for CryptoServer CSe-Series.
Out_foil	Outer tamper-protecting foil damaged. This alarm reason is only relevant for CryptoServer CSe-Series.
ext_Erase	External deleting / clearing done
inval_MK	Invalid (corrupted) Master Key (reason for this is usually an empty battery, see below)
Power failed	Sensor controller without power
Sensory Controller failed	No reaction from sensor controller

Table 3: Possible alarm reasons

Whenever a physical alarm occurs (noticed by the sensor), a dedicated alarm bit in the alarm status sensor register will be set immediately, together with bits which indicate the alarm reason.

The Master Key will be deleted immediately (i.e., the Key-RAM will be actively erased by the sensor 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. A 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), all security-relevant data inside the CryptoServer is deleted.

When an alarm is triggered, the following data is automatically deleted on the CryptoServer:

- Master Key
Deleting the Master Key automatically makes all other keys (including the MBK) and

sensitive data stored on the CryptoServer unusable, because without the Master Key they can no longer be decrypted.

- HSM Authentication Key (`authkey.db`)
- Firmware Encryption Key
- The audit log signature key (`auditkey.db`)
- Master Backup Key (MBK)
- All users in the user database (`user.db`) using HMAC password authentication
- Cryptographic key database (`CXIKEY.db`)

When an alarm is triggered, the following items are not deleted:

- Public parts of the:
 - Production Key
 - Default Administrator Key
 - Module Signature Key
 - Alternative Module Signature Key
- All firmware modules (`*.msc` files) except for the FIPS140 firmware module.



To meet the requirements of the FIPS 140-2 standard for cryptographic modules, a special FIPS mode has been introduced to the CryptoServer. In FIPS mode CryptoServer's behaviour differs from normal behaviour in some points (see CryptoServer's specialized FIPS 140 documentation). FIPS mode only becomes active if firmware module FIPS140 is loaded into the CryptoServer.

- Bootloader code and system firmware modules (`*.sys`)
- Alarm state file (`alarm.sens`)
- Audit log file(s) (`audit*.log`)
- Audit log configuration file (`audit.cfg`)
- Any Signed License File (`*.slf`), if present
- Signed configuration file `cmds.scf` , if available

- Bootloader configuration file (`bl.cfg`)
- Configuration file for the maximum number of consecutive failed authentication attempts `auth_fails.max`
- All users in the user database (`user.db`) with a public key authentication mechanism, e.g., RSA signature authentication, ECDSA signature authentication or RSA smartcard authentication
- A second copy of the public part of the Default Administrator Key (`init.key file`). This copy cannot be changed nor deleted. It is used to restore the default administrator user ADMIN with his Default Administrator Key as authentication token using the `csadm Clear = DEFAULT` (ClearFactoryDefaults) command, see "*The Clear Functionality*" in the *CryptoServer – csadm Manual*, or **CAT > Manage > Clear to Factory Settings**.

Consider that the list differs from the items that are deleted if you use the **Clear** command, see Clear.

After an alarm, the CryptoServer is no longer in Operational Mode and will go into Maintenance Mode after a restart.

If the CryptoServer is stored for a long period of time without voltage supply and the battery gets empty, the 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 bootloader additionally checks at each boot process the integrity of the loaded Master Key. If the verification fails, the bootloader will then activate a 'virtual' alarm. This alarm is displayed as `Inval_MK`, see above.

The alarm must be reset by an administrator who has the appropriate authentication status before the CryptoServer returns to Operational Mode. Before you reset an alarm, you should find out why it was triggered. When doing this, note the following:

- If it was a temporary alarm, and the event that triggered it is no longer present, e.g. the power supply and the internal temperature of the CryptoServer are again within the permitted threshold values, the CryptoServer returns to Operational Mode after a reset of the alarm and a restart.
- If it was is a temporary alarm and the event that triggered it is still present, e.g. the power supply or the CryptoServer's internal temperature are still not within the permitted threshold values, the CryptoServer goes into Maintenance Mode after a restart. The CryptoServer will not return to Operational Mode after a restart until you resolved the event that triggered the alarm and reset the alarm.

- If it is a permanent alarm, e.g. the tamper-protecting foil in a CryptoServer CSe-Series has been damaged or destroyed, the CryptoServer returns to Maintenance Mode after a restart. Any attempt to reset the alarm will fail. You cannot bring the CryptoServer into operation until it has been repaired by Utimaco.

3.1.4 Clear

Use the `clear` command to manually delete the following sensitive data and firmware modules from the CryptoServer:

- Master Key
The deletion of the Master Key automatically makes all other keys and sensitive data stored on the CryptoServer unusable because they can no longer be decrypted without the Master Key. A new Master Key is generated for the CryptoServer.
- HSM Authentication Key (`authkey.db`)
- Firmware Encryption Key
- The audit log signature key (`auditkey.db`)
- Master Backup Key (MBK)
- All firmware modules (`*.msc` files) except for the bootloader code and the system firmware modules (`*.sys` files)
- Signed configuration file `cmds.scf`
- All users in the user database using a HMAC password for authentication

The following items are not deleted:

- Public parts of the Production Key, Default Administrator Key, Module Signature Key and Alternative Module Signature Key
- Bootloader code and system firmware modules (`*.sys`)
- Alarm state file (`alarm.sens`)
- Audit log file(s) (`audit*.log`)
- Audit log configuration file (`audit.cfg`)
- Any Signed License File (`*.slf`), if present

- Bootloader configuration file (`bl.cfg`)
- All users in the `user.db` user database using a public key authentication mechanism, e.g., RSA signature authentication, ECDSA signature authentication or RSA smartcard authentication
- A second copy of the public part of the Default Administrator Key (`init.key` file). This copy can neither be changed nor deleted, but it is used to restore the default administrator user ADMIN with his Default Administrator Key as authentication token (using the `csadm Clear = DEFAULT` (ClearFactoryDefaults) command, see *The Clear Functionality* in the *CryptoServer – csadm Manual*, or **CAT > Manage > Clear to Factory Settings**).



Note that the above list differs from the items that are deleted due to an alarm, see [An Alarm and Its Consequences](#) (p. 38).

After the `Clear` the CryptoServer is no longer in Operational Mode and returns to Maintenance Mode after a restart.

The CryptoServer does not return to Operational Mode until the firmware modules of the corresponding SecurityServer firmware package are reloaded, for details about how to reload the firmware package see sections *Commands for Firmware Management* and *Commands for Firmware Packaging* in *CryptoServer - csadm Manual*.



You should only use the `Clear` command if you want to set up or reinstall the CryptoServer again.

3.1.5 Clear to Factory Settings

If you cannot find the user authentication key for the default administrator, ADMIN, and can therefore no longer administer the CryptoServer, you can also perform an External Erase directly on the CryptoServer and on the CryptoServer LAN.

However, if you do perform this External Erase directly on the CryptoServer or the CryptoServer LAN, this will trigger an alarm which enables you to perform a Clear to Factory Settings.

The Clear to Factory Settings command can only be implemented if the alarm triggered by the *External Erase* is still present.

When you run the `Clear to Factory Settings` command, it resets the CryptoServer to the state it was in when it was first supplied. The `Clear to Factory Settings` command triggers the following actions:

- The firmware modules are deleted.
Only the system firmware modules required for base administration remain on the CryptoServer.
- All users in the CryptoServer user database are deleted.
- The default administrator ADMIN is set up again and can use the original user authentication key ADMIN.key to log in to the CryptoServer.
- A new Master Key is generated for the CryptoServer. This automatically makes all other keys (including the MBK) and sensitive data stored on the CryptoServer unusable, because they can no longer be decrypted without the "old" Master Key.

After `Clear to Factory Settings` the CryptoServer is no longer in Operational Mode and returns to Maintenance Mode after a restart.

The CryptoServer does not return to Operational Mode until the firmware modules of the corresponding SecurityServer package are reloaded.

3.1.6 External Erase

An **External Erase** is accompanied by an alarm and actively deletes the following sensitive data from the CryptoServer:

- The Master Key
This automatically makes all other keys (including the MBK) and sensitive data stored on the CryptoServer unusable, because they can no longer be decrypted without the Master Key.
- All users from the user database who use a password-based mechanism to authenticate themselves

However, all users who log in to the CryptoServer with RSA Signature, RSA Smartcard or ECDSA are not deleted.

After the External Erase the CryptoServer is no longer in Operational Mode and returns to Maintenance Mode after a restart.

The alarm triggered by an **External Erase** must be reset by an administrator who has the appropriate authentication status before the CryptoServer returns to Operational Mode.

The alarm triggered by an **External Erase** is the prerequisite for performing a **Clear to Factory Settings** on the CryptoServer.

However, you must ensure that the alarm triggered by the **External Erase** is not reset so that the **Clear to Factory Settings** function can also be performed.

An External Erase can only be performed on the CryptoServer PCIe card during normal operation when the host computer is running. This is the only way to ensure the CryptoServer is supplied with enough power to perform the External Erase. Note that the different CryptoServer series – CryptoServer CSe and Se Gen2 - have different designs.

This section describes how to perform an External Erase on a CryptoServer PCIe Card. Detailed information on how to perform an External Erase on the CryptoServer LAN is in the *CryptoServer LAN – Administration Manual*.

CryptoServer CSe-Series

1. Make sure that the CryptoServer PCIe card has been plugged in a computer and this computer has been switched on. The next step is only effective if these conditions are fulfilled.
2. Push the push-button under the aperture labeled **A** by using an appropriate screwdriver. You find this aperture on the slot plate of a CryptoServer CSe-Series, see next figure.



Figure 8 : Slot plate of a CryptoServer CSe-Series PCIe card

3. Check if the CryptoServer is now in Maintenance Mode by clicking **Show Status** in CAT or by performing a `csadm GetState` command.



Regardless of whether you have performed an External Erase or not, the following applies:

If you remove the CryptoServer PCIe card from the computer and remove any battery from this PCIe card, the sensitive data on this PCIe card is deleted automatically in any case after a maximum of 30 minutes.

CryptoServer Se-Series Gen2

1. Push the push-button under the aperture labeled **B** by using an appropriate screwdriver. You find this aperture on the slot plate of a CryptoServer Se-Series Gen2, see next figure.



Figure 9 : Slot plate of a CryptoServer Se-Series Gen2 PCIe card

The LED flash light **C** flashes up red to confirm the activation of the Erase push-button **B**.



This action can be performed as well if the Cryptoserver PCIe card is not plugged in a computer or the computer is switched off. However, then the next step is not possible.

2. Check if the CryptoServer is now in Maintenance Mode by clicking **Show Status** in CAT or by performing a `csadm GetState` command.



Regardless of whether you have performed an External Erase or not, the following applies:

If you remove the CryptoServer PCIe card from the computer and remove any battery from this PCIe card, the sensitive data on this PCIe card is deleted automatically in any case after a maximum of 30 minutes.



The External Erase has been performed successfully and the device is in Maintenance Mode.

3.2 Smartcards, PIN Pads and Keyfiles

Smartcards, PIN pads and keyfiles are used to administer the CryptoServer.

3.2.1 Smartcards

Only smartcards that are provided by Utimaco IS GmbH can be used for the administration of the CryptoServer. The smartcards are preconfigured by Utimaco before they are shipped.



The CryptoServer LAN is supplied with ten smartcards.
If you purchase a CryptoServer PCIe, smartcards are not included in the deliverables.

Each one of these smartcards is already preloaded with a user authentication key for the default administrator ADMIN.

The smartcards are PIN protected. If you enter the incorrect PIN three consecutive times, the smartcard is blocked and can no longer be used. If this happens you must order a new smartcard from Utimaco.

3.2.2 PIN Pads

The PIN pads are smartcard readers to be used with the CryptoServer. They have an integrated display and a keypad, and are supplied exclusively by Utimaco IS GmbH. You cannot use any other PIN pads for the CryptoServer.



The CryptoServer LAN is supplied with a PIN pad.
If you purchase a CryptoServer PCIe, the PIN pad is not included in the deliverables.

Prior to SecurityServer 4.10 the REINER SCT cyberJack PIN pad has been supplied.



Figure 10 : REINER SCT cyberJack PIN pad

As from SecurityServer 4.10 a new PIN pad of type Utimaco cyberJack one has been introduced.

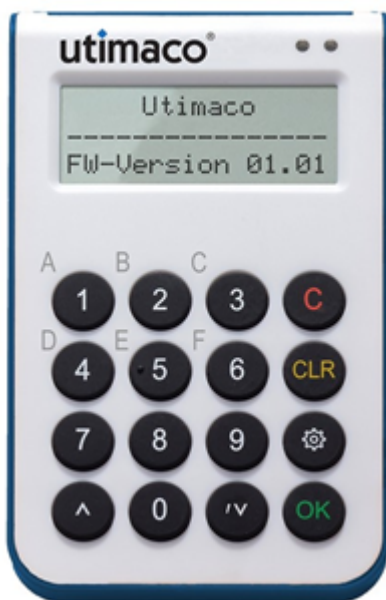


Figure 11 : Utimaco cyberJack one

Both PIN pads provide the same functionality and can be used with the smartcards delivered by Utimaco IS GmbH.



As of SecurityServer 4.10, the CryptoServer LAN is supplied either with the REINER SCT cyberJack or with the Utimaco cyberJack one. To use these PIN pads on a Windows host computer, the appropriate PIN pad driver must be installed as described in [Installing the PIN Pad Driver \(p. 139\)](#). For using the PIN pads on a Linux host computer, the udev rule provided in [Configuring the PIN Pad \(p. 151\)](#) must be defined.

3.2.3 Keyfiles

After the [installation \(p. 128\)](#) of the CryptoServer host software from the product CD, the authentication keyfile ADMIN.key for the default administrator ADMIN is stored on the host computer you are going to use for the CryptoServer administration.



Keyfiles can be used either with or without password protection. However, we strongly recommend that you always use a password to protect your keyfiles.

3.3 User Management: Users, User Groups and Authentication

To control and restrict the access to security-relevant commands, the CryptoServer implements the concept of users. Security-relevant external commands sent to the CryptoServer may only be performed if the sender has authenticated the command. Only users, who are registered at the CryptoServer, and are equipped with the relevant permissions are permitted to authenticate and execute security-relevant commands.

Additionally, the SecurityServer/CryptoServer SDK 4.10 introduces the CryptoServer's ability to authenticate itself towards host applications at the beginning of a secure messaging session, if requested by the corresponding application. CryptoServer's concept of HSM authentication implements a new HSM Authentication Key described in chapter [HSM Authentication Key \(p. 94\)](#) and used for mutual authentication.

3.3.1 Users

Every user is identified by a unique name. In addition, every user is assigned a number of other properties. All user data is stored in a user database on the CryptoServer.

Property	Meaning
Name	Used as a unique identification for a user
Permission	User's permissions for the different user groups

Property	Meaning
Authentication mechanism	Defines how the user is to authenticate themselves
Authentication token	Keyfile or password depending on the authentication mechanism
Attributes	Additional data used to specify particular properties or restrictions, such as that a user may only access specific keys

Table 4: CryptoServer user properties

3.3.2 User Groups

The user management functionality has eight user groups, numbered from group 0 to group 7. Some user groups are reserved by Utimaco for applications and their corresponding role-based user profiles.

Group	Application	Role-based user profile and permissions
0	All cryptographic interfaces	Cryptographic User and PKCS#11 User; 00000002
1	PKCS#11	PKCS#11 Key Manager; 00000020
2	PKCS#11	PKCS#11 Security Officer (SO); 00000200
3		
4		
5	NTP Administration	NTP Manager; 00200000
6	System Administration	System Manager; 02000000
7	User Management	User Manager; 20000000

Table 5: User groups and role-based user profiles reserved for specific applications

It is possible to assign new permissions to a predefined user group. As a result, all users in a predefined user group are automatically given the new permission. Therefore, a new permission in group 5 (NTP Administration) can also administrate NTP. As the predefined user groups 6 and 7 have already been assigned wide-ranging administration permissions, we recommend not assigning other permissions to these groups.



The permissions within applications have already been defined at the programming stage. Once an application has been loaded into the device, the assignment to a user group or the permission level cannot be changed. When you assign new permissions within the individual user groups, you must also take into account the fact that several user groups have already been predefined by Utimaco.

3.3.3 Permissions and Authentication Status

The device accepts certain external commands only after one (or more) appropriate user(s) have been successfully authenticated and a certain authentication state has been reached. The firmware module CMDS is responsible for managing the authentication of commands.

The device requires all security-relevant commands to be always authenticated separately. This means that the authentication holds only for a single command and must be performed together with the corresponding command. After the execution of the command the authentication status is automatically set back to the previous value.

The authentication concept of the device contains eight different user groups (0 to 7) that each belong to a list of available commands. Each command has a permission level. Every user is assigned a permission level as well, which determines if they are allowed to execute commands on their own or need other users for authentication. Each authentication level can vary from 0 (no permission/authentication) to 15 (highest level of permission/authentication).

Example

Value of authentication state	2	0	0	1	0	3	0	1
Authentication Level for User Group	7	6	5	4	3	2	1	0

In this example, the device 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 device, the authentication state will be augmented by the permissions of the user:

Example

Initial authentication status (no user logged in)	00000000
Permissions of the Admin1	21000000
Permissions of the Admin2	33000000
Permissions of the Admin3	51000000
Permissions of the Admin4	51000000
Permissions of the User	00000002
Authentication status: Admin1, Admin2, Admin3, Admin4 and User logged in	F6000002

Consider that the authentication statuses can be added without any further restriction as described above only if the involved users are logged in to perform a command

- of a firmware module that is not the CXI firmware module or
- of the CXI firmware module and the cryptographic key (CXI key) that is used to perform this command does not belong to any key group.

If however, the used cryptographic key in the CXI firmware module is assigned to a key group, only the authentication statuses of those logged in users can be summed up that belong to the same key group.

Example:

userA has authentication status 0x00000001 and is assigned to the key group A.

userB has authentication status 0x00000001 and is assigned to the key group B.

Three cryptographic keys are created. keyA is assigned to key group A, keyB to key group B and keyAB to key group AB.

If userA and userB are simultaneously logged in, only those commands of the CXI firmware module can be performed that only need

- an authentication status 0x00000001 for using keyA or
- an authentication status 0x00000001 for using keyB.

If a command needs

- an authentication status 0x00000001 or higher for using keyAB or
- an authentication status 0x00000002 or higher for using keyA or

- an authentication status 0x00000002 or higher for using keyB,

it cannot be performed.

3.3.4 Permissions for New Users

The permissions within applications have already been defined at the programming stage. Once an application has been loaded into CryptoServer you can no longer change its assignment to a user group or the permission. When you assign new permissions within the individual user groups, you must also take into account the fact that several user groups have already been predefined by Utimaco. The following table shows which user groups have been predefined by Utimaco for applications and the corresponding role-based user profiles and which are unused:

<i>User Group</i>	<i>Application</i>	<i>Role-based user profile and permissions</i>
0	All cryptographic interfaces	Cryptographic User and PKCS#11 User; 00000002
1	PKCS#11	PKCS#11 Key Manager; 00000020
2	PKCS#11	PKCS#11 Security Officer (SO); 00000200
3		
4		
5	NTP Administration	NTP Manager; 00200000
6	System Administration	System Manager; 02000000
7	User Management	User Manager; 20000000

Table 6: CryptoServer user groups and their permissions

In principle, it is possible to assign new permissions to a user group that has already been predefined. As a result, the new permissions in a predefined user group are automatically given the predefined permission. Therefore, a new permission in group 5 (NTP Administration) can also administrate NTP.



As the predefined user groups 6 and 7 have already been assigned wide-ranging administration permissions for the CryptoServer, we recommend you do not assign any other permissions to these groups.

3.3.5 Authentication Mechanisms



The authentication mechanisms described here must not be confused with the `AuthMech` configuration parameter in the `ppd.cfg` configuration file.



A maximum for failed authentication attempts can be set via `csadm`, see *Maximum Authentication Attempts* in the *CryptpServer - csadm Manual*.

The following authentication mechanisms are implemented:

3.3.5.1 HMAC Password Authentication

The device supports HMAC password mechanism for user authentication. The HMAC password has an arbitrary length (minimum 8 byte). It is transferred to the device in a protected manner: A random number is added to each authentication. This prevents the authentication from being intercepted and performed again at a later point in time. For details about configuring HMAC password requirements, see [Configuring HMAC Password Requirements \(p. 212\)](#).

Procedure

First, the host demands an 8-byte random value for each authentication from the device. Then the host calculates the HMAC value over this random value and the command data block using the password as the input for the HMAC key-derived function (HMAC-PBKDF). This HMAC hash value is transmitted to the device together with the command data. The device recalculates and checks the hash with by using the password stored in the user database `user.db`. The default hash algorithm for the HMAC calculation is SHA-256. Other hash algorithms can be used on demand (not in FIPS mode).

The HMAC-PBKDF (HMAC password-based key-derived function) according to NIST SP 800-132 does not use the HMAC password itself for authentication but a function derived from the HMAC password. For HMAC-PBKDF, 1000 iterations are used here. This number of iterations, as used for the key derivation from a given password, is fix and not configurable. HMAC-PBKDF is only applied if on both sides, the host side and the firmware side, HMAC-PBKDF is applied. If it is not available on one of these sides, the legacy version of the HMAC password-based mechanism is applied, whereby the user's password is used directly as an HMAC key. In FIPS mode, using HMAC-PBKDF is mandatory.

Deployment

HMAC Password Authentication has the following advantages:

- The password is not submitted in clear text and can therefore not be scanned.
- Because of the random value the authentication data block cannot be scanned and replayed at a later time. A 'Playback'-attack of the command becomes impossible.
- The command data is protected against unnoticed manipulation.

This authentication mechanism is also qualified for the remote authentication to a LAN device via Ethernet.

The administration tools support authentication with user name and HMAC password. This authentication mechanism cannot be used if you are administering a LAN device locally by using its front panel controls.

This mechanism is not as secure as the authentication with an RSA smartcard.

3.3.5.2 Signature-based authentications

If users with signature-based authentication mechanisms (RSA signature authentication, ECDSA signature authentication or RSA smartcard authentication) are created, their authentication token (public RSA or ECDSA key) is stored in the user database. Therefore, they need an ECDSA or RSA key pair either on a smartcard or in a keyfile (keyfile optionally encrypted).

All three authentication mechanisms offer the possibility to use the private part of the encryption key on a smartcard, which the users must keep in a safe place. Using the private key in a keyfile (*.key) on a computer is only possible for the RSA signature authentication and the ECDSA signature authentication. In case of the RSA smartcard authentication, only a smartcard can be used.



Except in test environments, we strongly recommend that you always save private keys on smartcards. Only in this case the private key never leaves the secure token and is not used for calculations on the host.

The corresponding public part of the encryption key is stored as authentication data in the `user.db` database.

- **RSA and ECDSA signature authentication**

This authentication mechanism is divided into the RSA signature authentication and the

ECDSA signature authentication depending on the used algorithm.

If the RSA or ECDSA authentication with smartcard is used, the smartcard must be inserted into a PIN pad that is connected to a USB port of the computer where the CAT or csadm tool is running or to another computer, see *Using a Local PIN Pad for a Remote CryptoServer* in the *CryptoServer – csadm Manual*.

Procedure

Every command that is sent to the device is signed by the user's private key. A random number provided by the device is included in the signature calculation process. This prevents a signed command from being intercepted and entered again later. The device then uses the corresponding public key to check this signature. The signature authenticates the command and checks whether the command is genuine, i.e., no transfer errors have occurred and there is no sign of data manipulation.

Deployment

As RSA and ECDSA signature authentication do not transfer any confidential authentication data, it is particularly suitable for remote authentication to the LAN device when the network connection is not completely under your control. RSA and ECDSA signature authentication can be used with all administration tools and for local administration tasks on the LAN device, as long as the user's private key is stored on a smartcard.

If the private part of the key is stored in a keyfile, RSA and ECDSA signature authentication cannot be used for local administration tasks on the LAN device.

- **RSA signature authentication**

Procedure

For this mechanism first the host demands an 8-byte random value from the device. Then the host (or RSA smartcard) calculates an RSA signature over this random value and the command data block with the user's private RSA key (PKCS#1 signature format). This command signature will then be transmitted to the device together with the command data. The device will verify the signature with the help of the RSA key's public part, which is stored in the user `user.db` database. The default hash algorithm for the signature calculation is SHA-256. Other hash algorithms can be used on demand (not in FIPS mode).

- **ECDSA signature authentication**

Procedure

For this mechanism the host first demands an 8-byte random value from the device. Then the host (or ECDSA smartcard) calculates an ECDSA signature over this random value and the command data block with the user's private ECDSA key.

This command signature will then be transmitted to the CryptoServer which will verify it with the help of the public part of the ECDSA key which is stored in the `user.db` database.

The default hash algorithm for the signature calculation is SHA-256. Other hash algorithms can be used on demand (not in FIPS mode).

▪ RSA smartcard authentication

In this authentication mechanism, the user's private RSA key is used only on a smartcard and not in a keyfile (`*.key`) on a computer. The corresponding public part is stored as authentication data in the `user.db` database.

The smartcard must be inserted in a PIN pad that is directly connected to the CryptoServer PCIe card. A connection of the PIN pad to the computer where the CAT or csadm tool is running on is not sufficient. The USB port of the PCIe card is used. If you use a LAN device, the PIN pad may also be connected to a port on the front panel that is directly connected to the CryptoServer PCIe card. Depending on the version of the LAN device, this is a USB port labeled **CS USB**, **USB CS**, **USB CS2** or **HSM**.

However, if you use the CryptoServer Simulator, the smartcard must be inserted into a PIN pad that is connected to the computer where the CAT or csadm tool is running (USB port) or to another computer, see *Using a Local PIN Pad for a Remote CryptoServer* in the *CryptoServer – csadm Manual*.

If users who use RSA smart card authentication are to be created, the PIN pad for authenticating this command (not for making the new user's private RSA key available) does not necessarily need to be connected directly to the PCIe card. It depends on the authentication mechanism the user administrator uses who authenticates the user creation process, see *Creating a User* in the *CryptoServer - CAT Manual*.

RSA smartcard authentication does not support:

- ECDSA keys
- The user's private RSA key stored on a smartcard that is connected locally to the computer where CAT or the csadm tool is running on (exception: CryptoServer Simulator). However, when you create a user who shall authenticate commands using the RSA smartcard authentication mechanism, the smartcard with the user's private key needs to be available during the user creation process. This smartcard must be inserted in a PIN pad that is connected to the computer where CAT or the csadm tool is running on.
- The user's private RSA key stored on a smartcard that is connected remotely to the CryptoServer, that means that the instructions in "Using a Local PIN Pad for a Remote CryptoServer", do not apply (exception: CryptoServer Simulator).

Although this authentication mechanism does not have "signature" in its name, it is a signature-based authentication mechanism. Although it is the only authentication mechanism having "smartcard" in its name, using a smartcard with the private key is not only supported by this mechanism but also by RSA signature authentication and ECDSA signature authentication.

Procedure

In contrast to the RSA and ECDSA signature authentication, the signature for the RSA smartcard authentication is not calculated before the command is sent to the device. The command is first sent to the device and then the device requests a signature via the RSA smartcard.

For this mechanism, the entire authentication is performed within the device, by using an RSA signature smartcard.

The user's smartcard (containing the user's RSA authentication key) has to be inserted into the PIN pad which is directly connected to the device. The device can now check the authentication of commands by requesting over this PIN pad a signature created with the private part of the user's RSA key. The device verifies the signature using the public part of the user's RSA key, which is stored in the user `user.db` database.

The default hash algorithm for the signature calculation is SHA-1. Other hash algorithms can be used on demand (not in FIPS mode).

Deployment

You should use the RSA smartcard authentication if you want to force an authorized user to authenticate a command by using a smartcard in a PIN pad that is directly connected to the USB port of the PCIe card.

3.3.5.3 Naming of Authentication Mechanisms in CAT and csadm

The table shows how the authentication mechanisms are named in **CAT >Manage > User... > column Mechanism** and in **Manage > User... > Add User > Authentication Mechanism**.

<i>Authentication mechanism</i>	<i>Authentication mechanism naming in the user management</i>	<i>Authentication mechanism naming in the "Add User" dialog</i>
HMAC password authentication	HMAC Password	Password (HMAC)
RSA signature authentication with a keyfile	RSA Sign	Smartcard (RSA Signature)
RSA signature authentication with a smartcard		Keyfile (RSA Signature)

Authentication mechanism	Authentication mechanism naming in the user management	Authentication mechanism naming in the "Add User" dialog
ECDSA signature authentication with a keyfile	ECDSA Sign	Smartcard (ECDSA Signature)
ECDSA signature authentication with a smartcard		Keyfile (ECDSA Signature)
RSA smartcard authentication	RSA Smartcard	Smartcard (PIN Pad at CryptoServer)

Table 7: Naming of authentication mechanisms in CAT

The following table shows which expressions are used for the authentication methods in csadm commands.

<i>Authentication mechanism</i>	<i>csadm ListUser output value</i>	<i>csadm authentication commands</i>	<i>csadm user creation parameter value</i>
HMAC password authentication	HMAC Password	csadm ... LogonPass=...	hmacpwd
RSA signature authentication with a keyfile	RSA Sign	csadm ... LogonSign=...	rsasign
RSA signature authentication with a smartcard			
ECDSA signature authentication with a keyfile	ECDSA Sign		ecdsa
ECDSA signature authentication with a smartcard			
RSA smartcard authentication	RSA Smartcard		rsasc

Table 8: Naming of authentication mechanisms in csadm

The authentication mechanism that is applied to a user can be retrieved from the **Mechanism** column in the output of the `csadm ... ListUser` command.

Example			
Name	Permission	Mechanism	Attributes
ADMIN	22000000	RSA sign	Z[0]I[0]
KeyUsrEcdsaSignKeyFile	00000002	ECDSA sign	
KeyUsrEcdsaSignSmartcard	00000002	ECDSA sign	
KeyUsrHmacPwd	00000002	HMAC passwd	I[0]
KeyUsrRsaSignKeyfile	00000002	RSA sign	

KeyUsrRsaSignSmartcard	00000002	RSA sign
KeyUsrRsaSmartcardPinPad	00000002	RSA smartcard

The following table shows the identifiers of the authentication mechanisms. These identifiers are shown in some audit log entries, for example, when a user is added or an authentication attempt has succeeded or failed.

<i>Authentication mechanism</i>	<i>Identifier</i>
HMAC password authentication	4
RSA signature authentication	0
ECDSA signature authentication	5
RSA smartcard authentication	3

Table 9: Authentication mechanism identifiers

3.3.6 Authentication Mode

Every security-relevant command must be authenticated by an authorized user before it is sent to the CryptoServer.

If you administrate the CryptoServer with the command-line tool csadm, you shall authenticate every command individually, with the csadm LogonSign or LogonPass command, depending on the user's authentication mechanism.

If you administrate the CryptoServer with the Java-based administration tool CAT, a Secure Messaging connection to the CryptoServer is established every time a user logs in to the CryptoServer. This connection is set to a maximum duration of 14 minutes and 59 seconds and resets to 14 minutes and 59 seconds every time the user sends a command to the CryptoServer, or when another user logs in to the CryptoServer. No matter how many administrators log in simultaneously to the CryptoServer, only one Secure Messaging connection is ever set up to the CryptoServer. As a consequence, all users currently logged on to the CryptoServer are logged off simultaneously. This is because the Secure Messaging connection, which is being used by these multiple users, is terminated.

3.3.7 ADMIN, the Default Administrator

On every CryptoServer, the default user ADMIN is created by default which enables the customer to do initial administration tasks. The ADMIN is allowed to perform all standard administration and user management commands.

The authentication mechanism of the default user ADMIN is RSA Signature and they are granted permission 2 for the user groups 6 and 7. The permissions of the user ADMIN cannot be changed. The Default Administrator Key `ADMIN.key` for the user ADMIN is

delivered on the SecurityServer product CD
(... \Administration\key) and on each of the delivered smartcards, enabling the user ADMIN to authenticate themselves towards the device. Since the ADMIN.key is not customer-individual, it should either be changed by the customer, or the ADMIN user should be replaced by other users with administrator rights:



Anyone who has the user authentication key on the smartcard or as a keyfile can log in to the CryptoServer as the default administrator ADMIN and manage the device on their own.

The user ADMIN should as soon as possible replace their Default Administrator Key ADMIN.key by an individual authentication token, i.e., by an individual RSA key.

In case the two-person rule is required for the administration tasks, the user ADMIN must be replaced by two (or more) new users with the necessary user permissions.

Permissions

The default administrator with authentication status 22000000 is permitted to perform these functions on the CryptoServer:

- Create a user
- Delete users
- Create a backup of user data
- Restore user data
- Load, replace and delete firmware modules
- Set time
- Reset alarm
- Clear CryptoServer
- Import Master Backup Key
- Delete audit l



In an emergency case, when the administration key is lost (e.g. smartcard(s) with private part of customer-individual administrator key is lost), the device can be reset by executing the `csadm Clear=DEFAULT` command. In this case, the device deletes the user database and creates a new one that contains the default ADMIN user described above.

For changing the authentication token of ADMIN, see [Replacing the Default Administrator ADMIN](#) (p. 151).

3.3.8 Default Cryptographic User

A Cryptographic User with permission 2 in the user group 0 (authentication status 00000002) has been set up in CAT by default. The Cryptographic User has all permissions in user group 0 and can authenticate all commands on its own. The Cryptographic User has been created to enable you to administer all cryptographic interfaces.

- CXI (Cryptographic eXtended Interface)
- JCE (Java Cryptography Extension)
- CSP/CNG (Microsoft CryptoAPI and Cryptography Next Generation)
- Open SSL (Cryptographic Token Interface), only if the CryptoServer is connected directly via an ENGINE interface
- EKM (Extensible Key Management)



The permissions assigned to the Cryptographic User in user group 0 have been defined in the CXI firmware module and cannot be changed.

3.3.9 PKCS#11 Users

If you want to apply PKCS#11 (for details, see "[PKCS#11](#) (p. 109)"), special restrictions regarding the users have to be considered.

- **CryptoServer Administrator**

The CryptoServer Administrator is responsible for administering the CryptoServer. Only the default ADMIN user or an administrator with permission 2 in the user group 7 (20000000) is permitted to initialize the tokens (the slots) by initializing the Security Officer's (SO's) authentication (PIN or password).

There are no restrictions regarding the administrator's authentication mechanism.

- **Security Officer (SO)**

The Security Officer (minimum permission 2 in user group 2; 00000200) is responsible for configuring the Cryptographic Token or Slots. The Security Officer (SO) can change the PIN or password they use for authentication, which was initialized by the ADMIN user (or a CryptoServer administrator with minimum permission 2 in the user group 7). In addition, only the Security Officer has the right to initialize the PIN or password used to authenticate the User.

The name of a Security Officer must be SO_xxxx with xxxx being a 2-byte decimal representation of the key group/PKCS#11 slot ID. The name may range from SO_0000 to SO_9999.

For a Security Officer, only HMAC password authentication is supported.

- **User**

A cryptographic user is called "User" in the PKCS#11 context. The User (minimum permission 2 in user group 0; 00000002; by default permission 2 in user group 0 and 1, 00000022) can generate, load, delete (00000020) and use (00000002) certificates or keys for the cryptographic token. This means that, by default, the User is a combination of a key manager and a key user. The User can change the PIN or password they use for authentication, which was initialized by the Security Officer (SO). This User can also be an application that uses the cryptographic token.

The name of the User must be USR_xxxx with xxxx being a 2-byte decimal representation of the key group/PKCS#11 slot ID. The name may range from USR_0000 to USR_9999.

For the User, only HMAC password authentication is supported.

For a list of functions that the User (key user, KU) can execute, see "Key Management Functions in PKCS#11" in the PKCS#11 R3 Developer Guide.

- **Configurable Key Manager**

The role of the User can be split into a key manager role, with permission (00000020) to manage cryptographic keys, and a key user role, with the permission (00000002) to use cryptographic keys. The CryptoServer PKCS#11 library should be configured accordingly, if you want to use these two new user roles.

It is advisable but not mandatory to use KM_xxxx as the name of the key manager with xxxx being a 2-byte decimal representation of the key group/PKCS#11 slot ID. The name

may range from KM_0000 to KM_9999.

There are no restrictions regarding the key manager's authentication mechanism.

For a list of functions that a key manager (KM) can execute, see "Key Management Functions in PKCS#11" in the PKCS#11 R3 Developer Guide.

For further details about available configuration objects, see *Setting Up the Configuration Objects* and *Separated Key Manager and Key User Role* in the *CryptoServer – PKCS#11 P11CAT Manual* or *Separated Key Manager and Key User Role* and the beginning of *Configuration Commands* in the *CryptoServer –PKCS#11 p11tool2 Reference Manual*.

PKCS#11 users can be created using the following tools:

- **CAT**

CAT can be used to create the PKCS#11 users listed above. The described restrictions must be considered, see *Creating a User* in the *CryptoServer - CAT Manual*.

- **csadm**

CAT can be used to create the PKCS#11 users listed above. The described restrictions must be considered. For details, see *AddUser* in the *CryptoServer – csadm Manual*.

- **P11CAT**

P11CAT can be used to create a Security Officer or the User. The requirements regarding the name and the authentication mechanism are fulfilled automatically.

For details, see Section "Setting up a Security Officer (SO)", Section "Setting up a User", Section "Setting Up the Configuration Objects" and Section "Separated Key Manager and Key User Role" in the *CryptoServer – PKCS#11 P11CAT Manual*.

- **p11tool2**

p11tool2 can be used to create a Security Officer (the `p11tool2 InitToken` command) or the User (the `p11tool2 InitPIN` command). The requirements regarding the name and the authentication mechanism are fulfilled automatically. For details, see *InitToken*, *InitPIN*, *Separated Key Manager and Key User Role* and the beginning of *Configuration Commands* in the *CryptoServer –PKCS#11 p11tool2 Reference Manual*.

For an overview of functions that can be performed by a PKCS#11 Key Manager or the PKCS#11 User, see "Key Management Functions in PKCS#11" in the PKCS#11 R3 Developer Guide.

3.3.10 Configurable Role-based Access Control (C-RBAC)

With release 4.0 of the SecurityServer product CD the CryptoServer provides the possibility to increase the required permissions for the authentication of specific CryptoServer functions.

The increased permission requirements can be individually configured by the customer for all external functions specified in the signed configuration file `cmds.scf`. During command execution, the currently reached permission for the function is compared to the required permission in the configuration file. If they are equal or no permission restrictions list exists, the command is executed normally performing the default (unchanged) permissions check for the command.

For detailed information about signed configuration files and their use, see *Signed Configuration Files* in the *CryptoServer – csadm Manual*. There you will also find a list of all firmware functions for which you might define enhanced permission requirements.

3.3.11 Backup of Users and Keys

In many applications, for security reasons or to set up a redundant device 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, the device provides the possibility to use Master Backup Keys.

A Master Backup Key is used to encrypt secret data which is exported from the device (for example, create a key backup), or to decrypt data, which is imported into the device (for example, restore a key backup). This is done, for example, by Utimaco's application firmware module CXI. It can also be used to backup and restore a user and his user data or database records. The device is able to store up to 256 Master Backup Keys (MBK slot 0...255; SecurityServer/CryptoServer SDK 4.20 or earlier: 4 Master Backup Keys corresponding to MBK slot 0...3) to be used by various applications. Each Master Backup Key (MBK) can either be a DES key (DES keys are supported for csadm versions < 2.5.3 only; 16-byte or 24-byte length) or an AES key (32 bytes). The MBK slot must not be confused with the PKCS#11 slot.

The handling of such Master Backup Keys is implemented over the firmware module MBK.

3.4 CryptoServer Modes and States

3.4.1 Operating Modes

The CryptoServer can be in one of four different modes, depending on which firmware is currently active.

Mode	Description
Bootloader Mode	<p>The bootloader is active, i.e., the bootloader is powered up and running but the operating system SMOS of the CryptoServer has not yet been started.</p> <p>With the <code>csadm ResetToBL</code> command, the CryptoServer can be set to Bootloader Mode, e.g. for diagnostic purposes. By performing a <code>csadm Restart</code> command in BootloaderMode, the CryptoServer can be set to Operational Mode.</p>
Operational Mode (optionally Administration Only)	<p>The operating system SMOS and further firmware modules (<code>*.msc</code> modules) have been started successfully and are active. The CryptoServer can be configured to boot into restricted Operational Mode (Operational Mode – Administration-Only). In this mode, all cryptographic functions are blocked, and only administration functions can be executed, see <i>SetAdminMode</i> in the <i>CryptoServer - csadm Manual</i>. If the CryptoServer is not set to boot into Administration Only Mode via the <code>c sadm SetStartupMode</code> command, restarting the CryptoServer will enable full Operational Mode again, see</p>
Maintenance Mode	<p>the CryptoServer is no longer ready for normal operation, and only the backup-set of firmware modules (<code>.sys</code>) is running. This mode provides an interface for performing administration tasks, but no cryptographic services are available.</p> <p>The Maintenance Mode occurs in the following situations.</p> <ul style="list-style-type: none"> ▪ An alarm occurs on the CryptoServer, either by sensor detection or an External Erase, see External Erase (p. 44) and An Alarm and Its Consequences (p. 38). ▪ The CryptoServer's operating system, SMOS, cannot be started or loaded without an error occurring. ▪ A Clear was performed to delete the entire contents of the CryptoServer, see Clear (p. 42). ▪ A <code>csadm ResetToBL</code> command was executed, followed by a <code>csadm RecoverOS</code> command, see <i>CryptoServer - csadm Manual</i>. <p>The administrator can perform all functions required to make the CryptoServer operational again. Applications like PKCS#11, JCE or CSP/CNG cannot access the CryptoServer when it is in Maintenance Mode. To reach full operativeness again, the CryptoServer has to be set to Operational Mode, see <i>Checking the Operativeness and the State of the CryptoServer</i> in the <i>CryptoServer – csadm Manual</i>.</p>
Power Down Mode	<p>No firmware is active, the CryptoServer is shutdown. In power down mode the CryptoServer is not able to receive any commands. A hardware reset has to be performed to get the CryptoServer active again.</p>

Table 10: CryptoServer Operating Modes

3.4.2 Operating States

The CryptoServer has two operating states.

Operating State	Description
INITIALIZED	When the CryptoServer is started correctly, it goes into the INITIALIZED operating state. As a minimum, the bootloader 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.
DEFECT	The bootloader performs a self-test every time the CryptoServer starts to check whether particular hardware and software components are functioning correctly. If this self-test detects an error, booting is interrupted and the CryptoServer switches to the DEFECT operating state. If it is still possible to call the CryptoServer's status whilst it is in the DEFECT operating state, you will see its operating state displayed as DEFECT. In this operating state the CryptoServer accepts either only very few or no administration commands.

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




If a significant hardware defect is present, it may not be possible to start the bootloader itself. If the bootloader cannot be started, the operating state DEFECT will not be displayed. In this situation, the CryptoServer's register displays an undefined value.

3.4.3 Error States


In every operating mode, Operational Mode, Maintenance Mode or Bootloader Mode, the CryptoServer can be in some error state.

Every time the CryptoServer is started after a reset, various initialization tasks and power-on self-tests are performed to check whether particular hardware and software components are functioning correctly. If an error occurs during this initialization (one of the power-on self-tests fails), or if a critical error is detected at a later time (one of the periodic firmware integrity tests failed), the security module enters a secure error state. Depending on the phase in which the error occurred, the error state is indicated as one of the following items:

- DEFECT means that an error occurred in a very early boot phase of the Bootloader module


 If the security module is in error state, denoted as `state = DEFECT` in the `csadm GetState` output, this implies that an irreparable hardware defect has occurred. Any security module being in DEFECT state can no longer be used for any cryptographic operations or administrative functions and has to be sent back to the manufacturer Utimaco IS GmbH (see chapter [Contact Address for Support Queries \(p. 283\)](#)).

- The error state in Bootloader Mode means that an error occurred in a later boot phase of the Bootloader.

 The CryptoServer is in DEFECT state, if the answer data of the `csadm GetState` command contains the following information:

```
mode = Bootloader Mode
state = INITIALIZED
error state (<hexadecimal error code>)
```

- The error state in Operational or Maintenance Mode means that an error occurred in the boot phase of the operating system of the security module, SMOS, or one of the periodic firmware integrity tests failed.

 The CryptoServer is in Maintenance Mode and error state, if the answer data of the `csadm GetState` command contains the following information:

```
mode = Maintenance Mode
state = INITIALIZED
error state (<hexadecimal error code>)
```

If the CryptoServer is in error state, no cryptographic services are available.

3.5 Implementation Environments

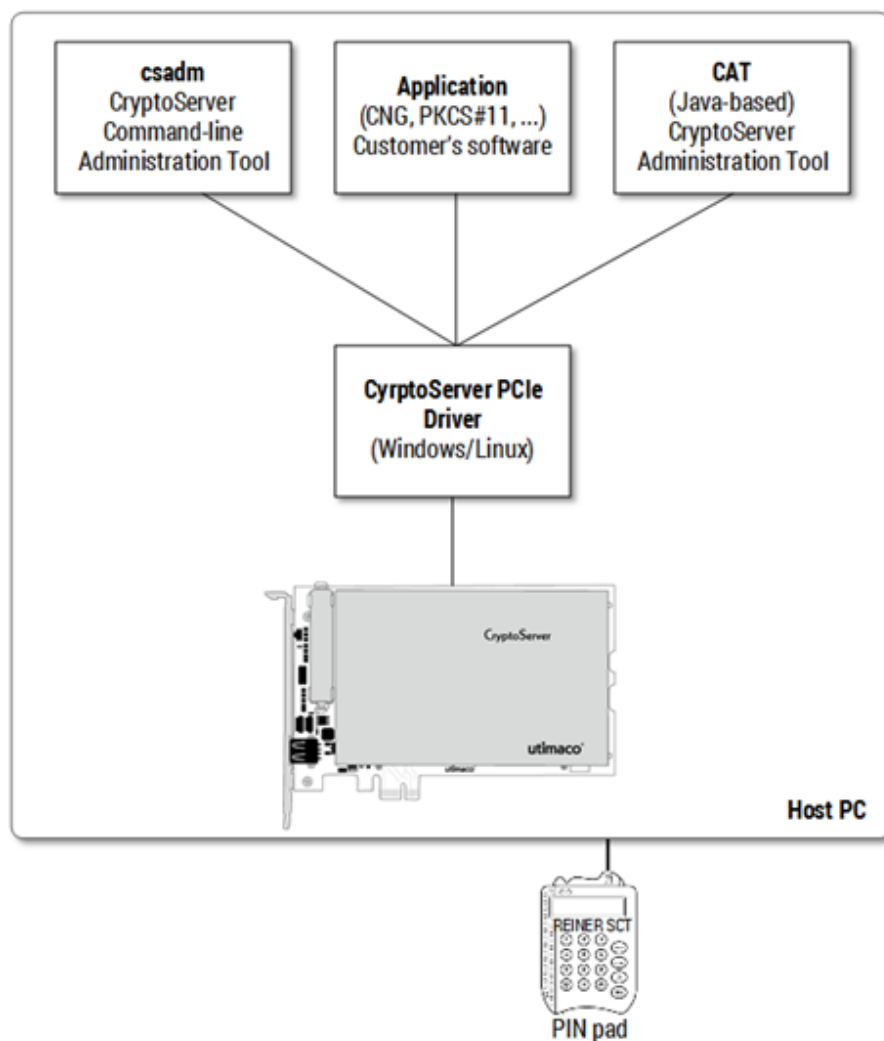


Figure 12 : Example for a system with integrated CryptoServer CSe PCIe card

The CryptoServer PCIe is mounted in a host computer that is running either a Windows or Linux operating system. To enable the operating system of the host computer to communicate with the CryptoServer, the CryptoServer PCIe driver for the corresponding operating system, has been installed on the host computer. You can run more than one CryptoServer PCIe on the same host computer. The CryptoServer itself is administered with the administration tools installed on the host computer.

You find the list of all currently supported operating systems for the host computer in the section [System Requirements](#) (p. 128).

Most of the commands used to administer the CryptoServer must be authenticated by users with appropriate permissions. Users can use a keyfile, a passwords or a smartcard as an authentication token. If you want to use smartcards, the PIN pad is usually connected directly

to the host computer via a serial or an USB port. For some security-relevant operations you can also connect the PIN pad directly to the CryptoServer PCIe.

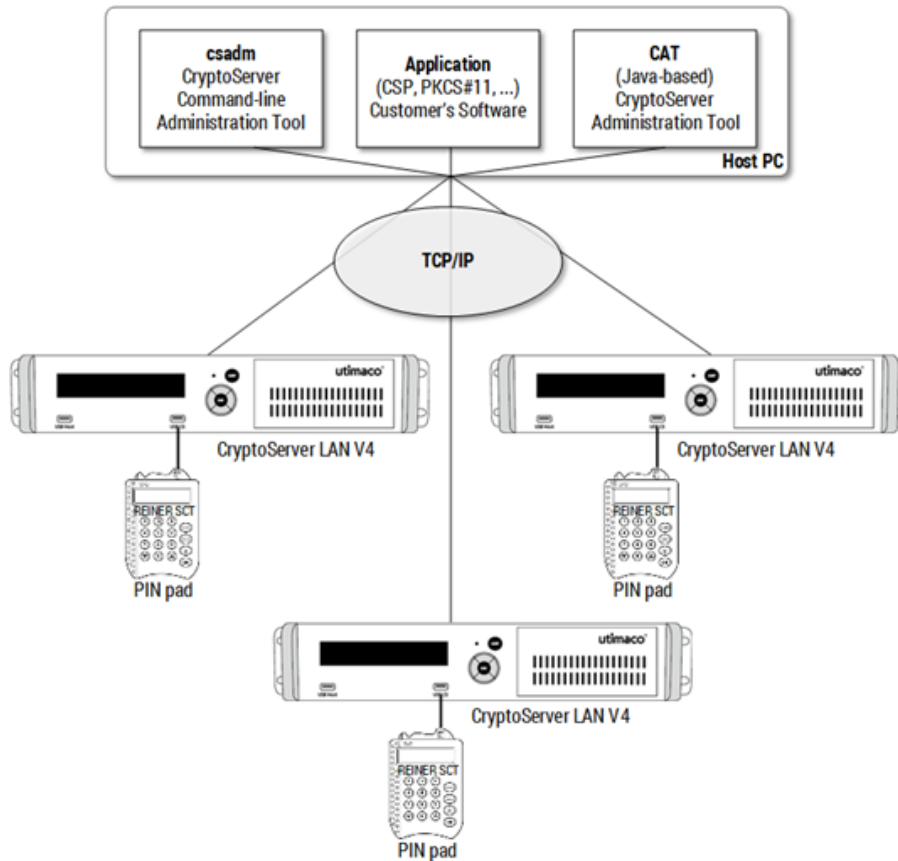


Figure 13 : Example for a system with three CryptoServer LANs

The CryptoServer LAN can be administered over a network from a host computer.

The CryptoServer PCIe, which is mounted in the CryptoServer LAN, can be administered either from the host computer or directly on the CryptoServer LAN itself. To do so, a PIN pad and ten smartcards are included in the CryptoServer LAN deliverables.

The network connection allows you to install the administration tools and the applications on the same host computer. However, you can also install them on different host computers. Several CryptoServer LANs can be implemented in the same network and managed from the central host computer.

You find the list of all currently supported operating systems for the host computer in the section [System Requirements \(p. 128\)](#).

If you want to administer the CryptoServer over the network, connect the PIN pad directly to one of the serial or USB ports of the host computer. If you want to perform administration tasks locally on the CryptoServer LAN, connect the PIN pad directly to the device.



The smartcards and the PIN pad serving as a card reader are not included in the CryptoServer PCIe deliverables and must be acquired separately from the manufacturer Utimaco IS GmbH.

3.6 Clustering for Load Balancing and Failover

Several CryptoServer LAN appliances can be organized in a cluster either for load balancing or for failover purposes, see [Failover \(p. 72\)](#).

Configured to support failover, the CryptoServer cluster ensures high system availability and reliability.

In a load balancing cluster, CryptoServer LAN devices are configured to optimize the use of resources to reach higher system performance, and to avoid overloading a single device. A load balancing cluster also implicitly offers high availability in case some CryptoServer in the cluster fails, yet at the cost of reduced system performance compared to the performance of a fully functional cluster.



See [Load Balancing Mechanism \(p. 191\)](#) for configuration.

3.6.1 Load Balancing

Load balancing defines the ability to distribute connections over several CryptoServer LAN appliances which are all active (rather than one active and the others passive).

All CryptoServer LAN appliances in the cluster must be configured identically. Connections are established by the client-side software based on the "Least Connections" principle, i.e. a new connection is opened on the CryptoServer LAN with the least number of existing connections. For example, let us assume that each of the N CryptoServer LAN appliances as shown in the figure below has currently exactly one open connection (Connection 1 to N). When later on connection 2 is closed and subsequently the connection N+1 should be established, this connection will be opened on the CryptoServer LAN 2 that has the least number of connections at that time, namely no connections at all.

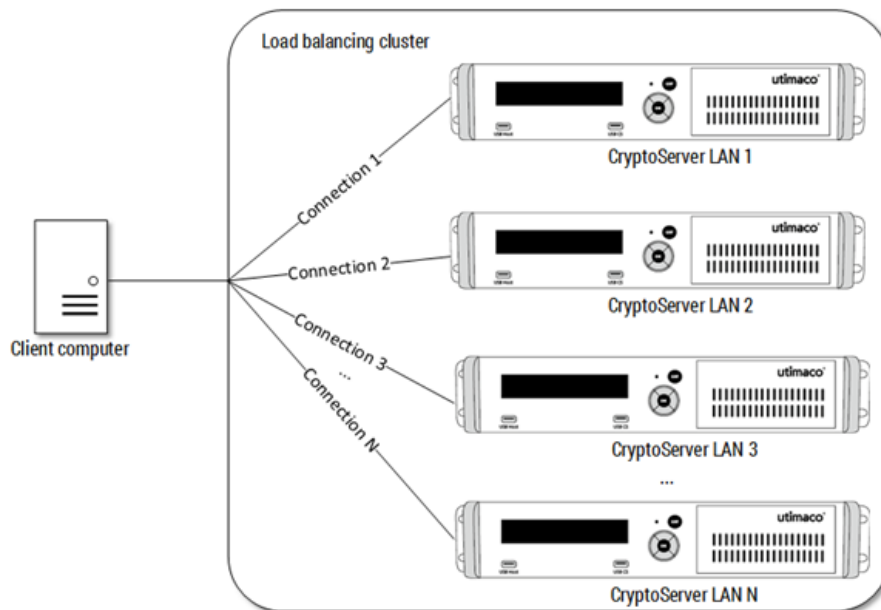


Figure 14 : Example of a load balancing cluster with N CryptoServer LANs

Load balancing has been tested with clusters of four CryptoServer LAN appliances. This is not due to limitations of CryptoServer or load balancing functionality. You may build clusters with more than four CryptoServer without any concerns.



Currently, load balancing is only supported by the CXI and PKCS#11 interfaces of the CryptoServer Se- and CSe- Series.

3.6.2 Failover

Failover defines the ability to automatically switch processing to a redundant or standby unit in a cluster, i.e. only the primary CryptoServer is active while other cluster members are passive. Usually, a cluster of two CryptoServer LAN appliances is deployed as a failover solution (see the figure below). When the primary CryptoServer LAN fails, the active connection(s) is (are) automatically routed to the passive CryptoServer LAN which continues to operate in the same manner as the failed device. The failover mechanism tries to reconnect to the primary CryptoServer in regular time intervals (fallback interval). In this context the primary CryptoServer can be the same device as before and which has become operable again; or it can be a passive CryptoServer which is used as a replacement for the

defect primary CryptoServer. On success the connections are switched back to the primary CryptoServer resp. transferred to the passive CryptoServer.

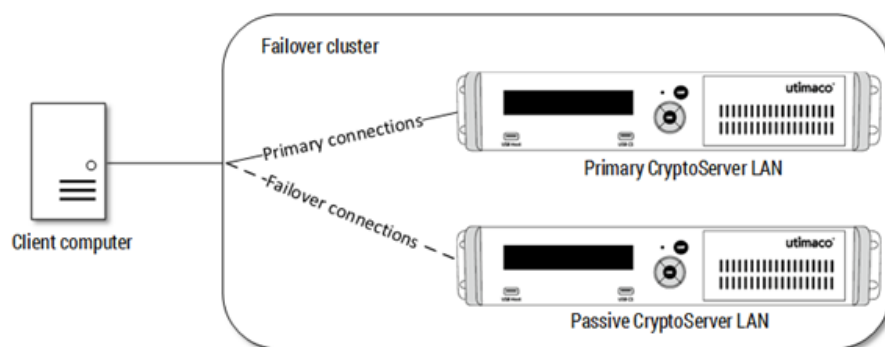


Figure 15 : Example for a failover cluster with two CryptoServer LANs

A cluster of more than two CryptoServer LAN appliances can be also deployed as a failover solution.

When choosing the fallback interval suitable for your system environment consider the following aspects to minimize the downtime of your failover cluster:

- If the passive CryptoServer LAN is available and preconfigured (identically to the primary CryptoServer LAN) in the same rack or in the local network, and just has to be switched on to be ready for use, a fallback interval of few minutes might be reasonable.
- If the passive CryptoServer LAN is locally available, but not in the same rack, and has to be configured identically to the primary CryptoServer LAN before connections can be rerouted to it, a fallback interval of few hours might be reasonable.
- If the passive CryptoServer is not locally available, can be delivered per express delivery and has to be configured identically to the primary CryptoServer before connections can be rerouted to it, a fallback interval of few days might be reasonable.
- If the primary CryptoServer has to be send back to the manufacturer Utimaco IS GmbH per [RMA request \(p. 283\)](#), and no passive CryptoServer is available, a fallback interval of several days might be reasonable.



The use of CryptoServer PCIe cards is only of limited use since normally for the replacement of a defect CryptoServer PCIe card the host computer must be shut down, and therefore the application and the CryptoServer cluster are completely out of operation.

3.6.3 Error Handling

Error-handling mechanisms for load balancing and failover are designed to minimize the possibility of disrupting the service availability of the CryptoServer cluster. The load balancing and failover mechanisms react to errors in a similar way:

- If an error indicates that the execution of the failed command might succeed on another device, e.g. in case of a timeout error, then a connection to the next CryptoServer is established, and the command is sent to that CryptoServer.
- If an error indicates that the execution of the failed command most likely will not succeed on another CryptoServer either, an error message is returned to the requesting application. This would be the case, e.g., if a key could not be found. The existing connection remains.

If a connection to a CryptoServer cannot be established, or a connection breaks and cannot be re-established, or any other error or problem dealing with load balancing and failover mechanism occurs, it is reported in the audit log.

3.6.4 Use of External and Internal Key Store

The load balancing and failover mechanisms support both internal and external key storage.

Internal key storage

- All cryptographic keys are stored in the cryptographic key database, `CXIKEY.db`, on the CryptoServer.



The load balancing and failover implementation provide an integrated key replication when **creating, importing, or restoring** keys, see [Key Replication and Internal Keystore \(p. 89\)](#).

- Cryptographic keys are stored within the tamper-protected area of the CryptoServer. In case any physical or logical attack has been detected by the sensors of the CryptoServer, an alarm is triggered and all cryptographic keys are automatically deleted. The number of cryptographic keys that can be stored inside the CryptoServer is limited by the storage capacity of the CryptoServer and depends on the key size and on the number and the size of the corresponding key properties.

For example, either approximately

- 3000 2048-bit RSA keys or
- 14000 ECDSA keys (NIST-P256)

can be stored in the internal keystore of the CryptoServer.

If you store 2048-bit RSA keys and ECDSA keys (NIST-P256) at the same time, the maximum number of keys to be stored is between 3000 and 14000.

The number for the 2048-bit RSA keys stated above applies if not only a private key but also the corresponding public key is stored. This is the usual case. If PKCS#11 is used, the private key and the public key are stored in separate objects. If the CSP/CNG provider is used, the private key and the public key are stored in the same object. If however only private keys are stored, either 4000 2048-bit RSA keys or 14000 ECDSA keys (NIST-P256) can be stored.

External key storage

- All cryptographic keys are stored in a database outside the CryptoServer

The following table gives an overview of the advantages and disadvantages to be considered when you decide to use either internal or external key storage in combination with load balancing or failover.

Key Storage	Advantages	Disadvantages
Internal	<ul style="list-style-type: none">▪ High security level since all keys are stored on the CryptoServer, i.e., additional protection by the HSM sensor, authenticated access▪ High performance	<ul style="list-style-type: none">▪ Manual key synchronization between all cluster members is required▪ Limited number of keys (several thousands, depending on the key length)

Key Storage	Advantages	Disadvantages
External	<ul style="list-style-type: none"> ▪ No manual key synchronization between cluster members is required ▪ High security level due to MBK (AES-256) protection of the external key storage ▪ Two types of external key storage <ul style="list-style-type: none"> • Flat file • Enterprise-level relational database (ODBC interface), starting with SecurityServer 4.40.0 <ul style="list-style-type: none"> • High availability through built-in replication • Performant storage and usage of millions of keys • Concurrent access from multiple applications/hosts 	<ul style="list-style-type: none"> ▪ Slightly lower performance than internal key storage

Table 11: Internal/External key storage – advantages and disadvantages

3.6.4.1 ODBC External Keystore

3.6.4.1.1 ODBC Preconditions

Windows

- SecurityServer-V4.55 or higher
- PostgreSQL or MSSQL database is accessible over the network
 - database hostname/ip
 - database TCP Port (default 5432)
 - database name, that should be used for the external key store
 - database user
 - database user password
- The PostgreSQL or MSSQL ANSI ODBC driver is available

Linux

- SecurityServer 4.60 or higher
- The PostgreSQL ODBC driver is available

3.6.4.1.2 PostgreSQL

PostgreSQL under Windows

1. Go to the `C:\Program Files\Utimaco\SecurityServer\Administration` directory:

```
cd "C:\Program Files\Utimaco\SecurityServer\Administration"
```

2. Add a new data source name (DSN) to the Windows Registry
`HKEY_CURRENT_USER\SOFTWARE\ODBC\ODBC.INI` using the `cxitool.exe` tool:

Command

```
cxitool.exe  
ConfigODBC=<dsnname>,<hostname>,<odbcpath>,<dbname>,<username>,<password>[,<  
registry>[,<port>]]
```

Example

```
cxitool.exe ConfigODBC="PostgreSQL_KEY_STORE",192.168.2.124,C:  
\my_dll_path\psqlodbc30a.dll,utimaco_db,db_user,utimaco,LOCAL,5432
```

3. Initialize the database schema:

```
cxitool.exe dbConnString="DSN=PostgreSQL_KEY_STORE" CreatedBSchema=postgres,  
force
```

4. Set up external key store for the PKCS#11 API.

Open the `C:\ProgramData\Utimaco\PKCS11_R3pkcs11_R3.cfg` file and add/change the following parameters:

Generated keys are stored in an external storage

```
KeysExternal = true
```

```
[KeyStorage]
```

Database via ODBC

```
KeyStorageType = ODBC  
KeyStorageConfig = "DSN=PostgreSQL_KEY_STORE"
```

Number of reconnection attempts to ODBC database.

```
KeyStorageReconnect = 5
```

5. Test if the keys can be stored in the ODBC external key storage.
Generate a new RSA key pair:

```
p11tool2.exe loginuser=12345678 PubKeyAttr=CKA_LABEL="My RSA Public Key",C  
KA_ID=0x52534132 PrvKeyAttr=CKA_LABEL="My RSA Private Key",CKA_ID=RSA2  
GenerateKeyPair=RSA
```

PostgreSQL under Linux



The examples below refer to Ubuntu.

1. Install PostgreSQL, unixODBC driver manager and driver for PostgreSQL.

Install the latest version of PostgreSQL.

```
sudo apt-get install postgresql
```

Install the unixODBC driver manager.

```
sudo apt-get install unixodbc
```

Install the ODBC driver for PostgreSQL.

```
sudo apt-get install odbc-postgresql
```



Usually the PostgreSQL database process are owned by the Linux user `postgres`. If the user has not been created by your install routines, please create the user manually. The `postgres` user is needed for the next step.

2. Create the database and user.

Open a SQL command line tool to create the database and users. Please refer to the PostgreSQL documentation on how to authenticate.

Change to the `postgres` user.

```
example-user@example-vb01:~$ su - postgres
```

As the `postgres` user run the `psql` command to open the **PostgreSQL interactive terminal** for the database administration.

```
postgres@example-vb01:~$ psql
```

There are different ways of creating databases. Please refer to the PostgreSQL documentation.

You can execute SQL statements or use some perl scripts which are installed with the `postgresql-all` package on Ubuntu.

Use the `createdb` script to create a database with the name `utimaco-test`, it will be used later for the ODBC config.

```
postgres@example-vb01:~$ createdb utimaco-test
```

Switch to the database `utimaco-test`.

```
postgres@example-vb01:~$ psql utimaco-test
```

Create the user `utimaco` with the password `utimaco`.

```
utimaco-test=# CREATE USER utimaco WITH ENCRYPTED PASSWORD 'utimaco';
CREATE USER
```

Define access privileges.

```
utimaco-test=# GRANT ALL PRIVILEGES ON DATABASE utimaco-test TO utimaco;
GRANT
```



The database and the database user are ready for ODBC connections.

3. Configure ODBC with the SecurityServer tools.

Example conditions:

- DSN `utimaco-dsn`
- database name `utimaco-test`
- user `utimaco`
- password `utimaco`
- PostgreSQL database is running on localhost (you can enter any IP address, if PostgreSQL runs on a remote system)
- ODBC driver lib location `/usr/lib/x86_64-linux-gnu/odbc/`

Command

```
cxitoool
ConfigODBC=<dsnname>,<hostname>,<odbcpath>,<dbname>,<username>,<password>
```

Example

```
example-user@example-vb01:~$ cxitoool ConfigODBC="utimaco-dsn,localhost,/  
usr/lib/x86_64-linux-gnu/odbc/psqlodbc.so,utimaco-test,utimaco,utimaco"
```




The command has just created the ODBC config files `.odbc.ini` and `.odbcinst.ini` in your home directory.



Run the `exitool ConfigODBC` command only once. Otherwise you have to manually edit the ODBC config files because of double entries.

The section `[ODBC Data Source\]` contains the chosen `utimaco-dsn` string. The driver points to `UtimacoODBCDriver` which must match the section `[UtimacoODBCDriver\]` in the file `.odbcinst.ini`.

```
example-user@example-vb01:~$ cat .odbc.ini
[ODBC Data Sources]
utimaco-dsn = auto generated ucapi external keystore dsn

[utimaco-dsn]
Driver      = UtimacoODBCDriver
Servername  = localhost
Database    = utimaco-test

example-user@example-vb01:~$ cat .odbcinst.ini
[UtimacoODBCDriver]
Driver=/usr/lib/x86_64-linux-gnu/odbc/psqlodbc.so
Debug=0
CommLog=1
UsageCount=1
```

4. Check the ODBC connection by using `isql` from the package `unixodbc`. If ODBC is configured correctly it will open a prompt to the database terminal. You can execute the SQL statement `SELECT 2 + 2;` for testing. Make sure to copy the `.odbc.ini` file to `/etc` because `isql` is using `/etc/odbc.ini` and `/etc/odbcinst.ini`.

```
# utimaco-dsn must be configured in /etc/odbc.ini
example-user@example-vb01:~$ isql -v utimaco-dsn utimaco utimaco
+-----+
| Connected!                                |
|                                           |
| sql-statement                            |
| help [tablename]                         |
| quit                                     |
+-----+
```

```

|
+-----+
SQL> SELECT 2 * 2;
+-----+
| ?column? |
+-----+
| 4         |
+-----+
SQLRowCount returns 1
1 rows fetched
SQL> quit

```

5. Set up the database schema for external keys with the cxitoool again. The command will create all necessary tables, columns and indexes.

Command

```
cxitoool dbConnString="DSN=Key Store" CreateDBSchema=postgres
```

Example

```
example-user@example-vb01:~$ cxitoool dbConnString="DSN=utimaco-dsn"
CreateDBSchema=postgres
```

Testing Connection

```
-----
Successful DB Connection
```

Testing Each Column Type Separately

```
-----
-- Testing hsm_keys ( id SERIAL PRIMARY KEY )
-- Testing hsm_keys ( key_blob BYTEA )
-- Testing hsm_keys ( key_size INTEGER )
-- Testing hsm_keys ( skey BYTEA )
-- Testing hsm_keys ( key_name VARCHAR(256) )
-- Testing hsm_keys ( key_group VARCHAR(256) )
-- Testing hsm_keys ( key_spec INTEGER )
-- Testing hsm_keys ( key_id BYTEA )
-- Testing hsm_keys ( unique_id BYTEA )
-- Testing hsm_keys ( unique_constrain BYTEA )
-- Testing hsm_str_attrs ( id SERIAL PRIMARY KEY )
-- Testing hsm_str_attrs ( att_id INTEGER )
-- Testing hsm_str_attrs ( att_value VARCHAR(256) )
-- Testing hsm_bin_attrs ( id SERIAL PRIMARY KEY )
-- Testing hsm_bin_attrs ( att_id INTEGER )

```

```
-- Testing hsm_bin_attrs ( att_value BYTEA )
-- Testing hsm_int_attrs ( id SERIAL PRIMARY KEY )
-- Testing hsm_int_attrs ( att_id INTEGER )
-- Testing hsm_int_attrs ( att_value INTEGER )

Create Tables and Indexes
-----
-- CREATE TABLE hsm_keys ( id SERIAL PRIMARY KEY , key_blob BYTEA ,
key_size INTEGER , skey BYTEA , key_name VARCHAR(256) , key_group VARCHAR(25
6) , key_spec INTEGER , key_id BYTEA , unique_id BYTEA , unique_constrain
BYTEA )
-- CREATE TABLE hsm_str_attrs ( id SERIAL PRIMARY KEY , att_id INTEGER ,
att_value VARCHAR(256) , key_id INTEGER REFERENCES hsm_keys )
-- CREATE TABLE hsm_bin_attrs ( id SERIAL PRIMARY KEY , att_id INTEGER ,
att_value BYTEA , key_id INTEGER REFERENCES hsm_keys )
-- CREATE TABLE hsm_int_attrs ( id SERIAL PRIMARY KEY , att_id INTEGER ,
att_value INTEGER , key_id INTEGER REFERENCES hsm_keys )
-- create unique index unique_id_property on hsm_keys (unique_constrain)

Integration Testing
-----
Testing Simple Insert:                Succeed
Testing Insert Duplicate Keys:        Succeed
Testing Insert Duplicate Keys (pkcs11): Succeed
Testing Find Keys By String Properties: Succeed
Testing Find Keys By Int32 Properties: Succeed
Testing Find Keys By Data Properties: Succeed
Testing Star Operator:                Succeed
Testing Get Key:                      Succeed
```



The cxitool binary has successfully created a database schema including everything necessary to store external keys in the database.

6. Configure PKCS#11 (External Key Store)

Update the PKCS#11 config file `cs_pkcs11_R3.cfg` and activate the settings for the external key store:

```
eample-user@example-vb01:~/Documents/HSM/4.60.0/Software/Linux/x86-64/
Crypto_APIs/PKCS11_R3/sample$ vi cs_pkcs11_R3.cfg

Logpath = /var/log/pkcs11
Logging = 4
```

```
[KeyStorage]
# Database via ODBC
KeyStorageType = ODBC
KeyStorageConfig = "DSN=utimaco-dsn"

# Number of reconnection attempts to ODBC database
KeyStorageReconnect = 5
```

7. Test the external key store via p11tool2.

```
example-user@example-vb01:~/Documents/HSM/4.60.0/Software/Linux/x86-64/
Administration$ p11tool2 loginuser=87654321 PubKeyAttr=CKA_LABEL="test-RSA-
pub",CKA_ID=0x52534133 PrvKeyAttr=CKA_LABEL="test-RSA-prv",CKA_ID=RSA3
GenerateKeyPair=RSA

30.01.2024 16:51:00.314 | [00024404:00024404]
initialize | I: CryptoServer PKCS#11 Library R3
version 1.30 (Sep 5 2023)
30.01.2024 16:51:00.314 | [00024404:00024404]
initialize | I: Configfile: /home/example-user/
Documents/HSM/4.60.0/Software/Linux/x86-64/Crypto_APIS/PKCS11_R3/sample/
cs_pkcs11_R3.cfg
30.01.2024 16:51:00.314 | [00024404:00024404]
initialize | I: Device List:
30.01.2024 16:51:00.314 | [00024404:00024404]
initialize | I: Device 3001@127.0.0.1
30.01.2024 16:51:00.314 | [00024404:00024404]
initialize | I: Multi-threaded access support
enabled: false
30.01.2024 16:51:00.314 | [00024404:00024404]
C_Initialize | T: leave C_Initialize()
30.01.2024 16:51:00.314 | [00024404:00024404]
C_OpenSession | T: enter C_OpenSession(slotID:
0x00000000, flags: 0x00000006, pApplication: 0, Notify: 0)
30.01.2024 16:51:00.319 | [00024404:00024404]
open_plugin | I: Opening KeyStorePlugin 'Ephemeral
Storage' (config: )
30.01.2024 16:51:00.319 | [00024404:00024404]
open_plugin | I: Opening KeyStorePlugin 'ODBC Storage'
(config: DSN=utimaco-dsn)
30.01.2024 16:51:00.319 | [00024404:00024404]
set_default_plugin_id | I: Set new default KeyStorePlugin 'ODBC
Storage'
30.01.2024 16:51:00.320 | [00024404:00024404]
C_OpenSession | T: leave C_OpenSession()
30.01.2024 16:51:00.320 | [00024404:00024404]
C_Login | T: enter C_Login(hSession: 0x00000001,
userType: 0x00000001)
```

```

30.01.2024 16:51:00.376 | [00024404:00024404]
C_Login | T: leave C_Login()
30.01.2024 16:51:00.376 | [00024404:00024404]
C_GenerateKeyPair | T: enter C_GenerateKeyPair(hSession:
0x00000001, pMechanism: CKM_RSA_PKCS_KEY_PAIR_GEN (0x0), pPublicKeyTemplate:
CK_ATTRIBUTE[4], pPrivateKeyTemplate: CK_ATTRIBUTE[3])
30.01.2024 16:51:01.216 | [00024404:00024404]
C_GenerateKeyPair | T: leave C_GenerateKeyPair()
30.01.2024 16:51:01.216 | [00024404:00024404]
C_Logout | T: enter C_Logout(hSession: 0x00000001)
30.01.2024 16:51:01.224 | [00024404:00024404]
C_Logout | T: leave C_Logout()
30.01.2024 16:51:01.224 | [00024404:00024404]
C_CloseSession | T: enter C_CloseSession(hSession:
0x00000001)
30.01.2024 16:51:01.224 | [00024404:00024404]
C_CloseSession | T: leave C_CloseSession()
30.01.2024 16:51:01.224 | [00024404:00024404]
C_Finalize | T: enter C_Finalize(pReserved: 0)
30.01.2024 16:51:01.224 | [00024404:00024404]
C_Finalize | T: leave C_Finalize()

```



Config Errors

If you have not configured ODBC correctly, e.g. typo in DSN or the database or user does not exist, the tools cxitool and p11tool2 may core dump:

```

example-user@example-vb01:~$ p11tool2 loginuser=87654321
PubKeyAttr=CKA_LABEL="test-RSA-pub",CKA_ID=0x52534134
PrvKeyAttr=CKA_LABEL="test-RSA-prv",CKA_ID=RSA4 GenerateKeyPair=RSA
free(): double free detected in tcache 2
Aborted (core dumped)

```

3.6.4.1.3 Microsoft SQL Server

1. Go to the `C:\Program Files\Utimaco\SecurityServer\Administration` directory:

```
cd "C:\Program Files\Utimaco\SecurityServer\Administration"
```

2. Add a new data source name (DSN) to the Windows Registry
`HKEY_CURRENT_USER\SOFTWARE\ODBC\ODBC.INI` using the `cxitool.exe` tool.

Command

```
cxitoool.exe
ConfigODBC=<dsnname>,<hostname>,<odbcpath>,<dbname>,<username>,<password>[ ,
<registry>[,<port>]]
```

Example

```
cxitoool.exe ConfigODBC="DSN=MSSQL_KEY_STOR",192.168.2.124,C:
\my_dll_path\psqlodbc30a.dll,utimaco_db,db_user,utimaco,LOCAL,5432
```

3. Initialize the database schema:

```
cxitoool.exe dbConnString="DSN=MSSQL_KEY_STORE" CreateDBSchema=mssql,force
```

4. Set up external key store for the PKCS#11 API.

Open the `C:\ProgramData\Utimaco\PKCS11_R3pkcs11_R3.cfg` file and add/change the following parameters:

Generated keys are stored in an external storage

```
KeysExternal = true
[KeyStorage]
```

Database via ODBC

```
KeyStorageType = ODBC
KeyStorageConfig = "DSN=MSSQL_KEY_STORE"
```

Number of reconnection attempts to ODBC database

```
KeyStorageReconnect = 5
```

5. Test if the keys can be stored in the ODBC external key storage.
Generate a new RSA key pair:

```
p11tool2.exe loginuser=12345678 PubKeyAttr=CKA_LABEL="My RSA Public Key",CKA_ID=0x52534132 PrvKeyAttr=CKA_L
```

EKM Extra Requirement

- Can only work with HKEY_LOCAL_MACHINE variables also the `cxitool ConfigODBC` command must be called with the `SYSTEM` parameter.

Example

```
cxitool.exe ConfigODBC="DSN=MSSQL_KEY_STOR",192.168.2.124,C:\my_dll_path\psqlodbc30a.dll,utimaco_db,db_user,utimaco,SYSTEM,5432
```

- EKM runs in the context of SQL Server, which runs under a different user, but network mounts are usually per user. Therefore, network drives have to be mounted by SQLServer using 'xp_cmdshell'.
 - ODBC path must be readable by SQL Server: either by putting it on a local drive or on a correctly mounted network driver.



See also *Microsoft SQL Server EKM Provider with Utimaco SecurityServer Integration Guide*, visit <https://support.hsm.utimaco.com/support/documentation/integration-guides> for more details.

3.6.4.2 Copy External Key Store

Copy external key store with the `cxitool`.

Syntax

```
cxitool [Dev=<dev>] <auth> SrcKeyStoreType=<srckeystoretype>  
SrcKeyStoreParam=<srckeystoreparam> KeyStoreType=<keystoretype>  
KeyStoreParam=<keystoreparam> CopyKeyStore
```

Authentication	<p>This command must be authenticated by one of the following user roles:</p> <ul style="list-style-type: none"> ▪ Key manager, level 2 in user group 1, 00000020, or level 2 in user group 0, 00000002 (default) ▪ Key user, level 2 in user group 0, 00000002 <p>Ensure that the authenticated user is assigned (<code>CXI_GROUP</code> user attribute) to the key group(s) specified by the <code><group></code> parameter. For more information, see section <i>Cryptographic Keys and Key Groups</i> in the <i>CryptoServer - cxitool Manual</i>. A user is assigned to a key group on creation. Use the <code>csadm ListUser</code> command or open Manage > User in the CryptoServer Administration Tool (CAT) to verify the key group the user is assigned to. For details, see the <i>CryptoServer – csadm Manual</i> and the <i>CryptoServer – CAT Manual</i>. Key groups must not be confused with user groups.</p>
-----------------------	---

Parameter	Description
<code><dev></code>	Device address of the CryptoServer or the CryptoServer LAN. This parameter can be omitted if the device address has been set as the CRYPTOSERVER environment variable.
<code><auth></code>	Authentication commands See CryptoServer - cxitool Manual <code>LogonPass</code> and <code>LogonSign</code> .
<code><srckeystoretype></code>	Either <code>SDB</code> (for legacy <code>.sdb</code> files), <code>EKM-SDB</code> (for <code>.sdb</code> files of EKM provider), <code>P11-SDB</code> (for <code>.P11</code> files of the P11-R2 provider) or <code>ODBC</code>
<code><srckeystoreparam></code>	<ul style="list-style-type: none"> ▪ For <code>KeyStoreType SDB</code>, <code>EKM-SDB</code> and <code>P11-SDB</code>: A path to an external key store file: <code>/path/to/store.sdb</code> ▪ For <code>KeyStoreType ODBC</code>: The data source name as defined in the format "<code>DSN=DATA SOURCE NAME</code>" or the complete ODBC connection string
<code><keystoretype></code>	Either <code>SDB</code> (for legacy <code>.sdb</code> file) or <code>ODBC</code>
<code><keystoreparam></code>	<ul style="list-style-type: none"> ▪ For <code>KeyStoreType SDB</code>: A path to an external key store file: <code>/path/to/store.sdb</code> ▪ For <code>KeyStoreType ODBC</code>: The data source name as defined in the format "<code>DSN=DATA SOURCE NAME</code>" or the complete ODBC connection string

Example	<pre>cxitool Dev=PCI:0 LogonPass=CXI_USER,ask SrcKeyStoreType=EKM-SDB SrcKeyStoreParam=C: \tmp\EKM\cssqlekm.sdb KeyStoreType=ODBC KeyStoreParam="DSN=Key Store" CopyKeyStore</pre>
----------------	--

3.6.4.3 Key Replication and Internal Keystore

The load balancing and failover implementation provides an integrated key replication when using the internal keystore::

- Key replication, is only available when **creating**, **importing**, or **restoring** keys.
- PKCS#11 must be used, see [Configuration Settings for the Use with PKCS#11](#) (p. 0)
- Load balancing must be configured properly, see [Load Balancing Mechanism](#) (p. 191)
- The MBK must be the same on all devices in the cluster.
- If an error occurs on a device, this is indicated in the log file with the respective device specifier, for example, 4002@192.168.0.143 or /dev/cs2.0.2).
- If an error occurs on a device, the function returns an error.
- If an error occurs, the cluster is in a problematic state.



There is no synchronization/restoration of the state when devices for which replication failed (for whatever reason) are working again and are accessed. This can lead to individual keys only being available on specific devices.

If a key is generated on a device where a previous replication failed, it may be that this key can no longer be replicated to other devices. This can lead to an operation with a key handle on device A using a different key than on device B, because the key handle points to different devices in A and B.

3.7 System Keys

This section describes the system keys, how they are generated, who is responsible for their storage and how they are used.

The user authentication keys except the Default Administrator Key are not described here.

3.7.1 Master Key

The Master Key is used to encrypt all other keys (including the MBK) and sensitive data stored on the CryptoServer.

The customer will never use Master Key directly. The CryptoServer itself is responsible for the generation, usage and erasure (in case of alarm) of this key.

The Master Key is CryptoServer-individual and will never leave the CryptoServer.

Type	The Master Key of the CryptoServer is a 32-byte AES key (default) or a 24-byte Triple DES key (in case that the firmware module AES is not loaded).
Generation	The key is generated automatically and randomly inside the CryptoServer every time the CryptoServer is brought into operation. The CryptoServer security module itself is responsible for usage and life cycle of this key.
Usage	The Master Key is needed to encrypt the data stored inside the CryptoServer. This is necessary to be able to store sensitive data also in memory areas which are not sensor-protected, i.e., which will not be erased within a very short time after an alarm has occurred. These are, for example, the flash device, NV-RAM and SD-RAM (CryptoServer CSe: DDR2 RAM). 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 firmware module provides an internal public interface for secure data storage to other application firmware modules. DB will use the Master Key in full length.
Life cycle	The Master Key cannot be read out, exported, imported; saved or restored, it is generated inside the CryptoServer. The Master Key is inaccessible to everyone, even to the CryptoServer owner. It is implemented exclusively within the CryptoServer itself. After generation, the key is stored inside the CryptoServer in the sensor-protected Key-RAM. The Master Key cannot be exported; hence it will never be available outside the CryptoServer. The life cycle of this key ends when an alarm occurs on the CryptoServer. If any alarm cause (temperature, foil, voltage, etc.) is detected by CryptoServer's sensor, the memory area in which the key is stored is automatically erased. A new Master Key is generated during the next boot process after successful execution of the ResetAlarm command. Additionally, the Master key is erased by executing the Clear command. Any Clear command erases the Master Key by generating and storing a new one (the previous one is overwritten).

Table 12: Details about the Master Key

3.7.2 Default Administrator Key

The Default Administrator Key is the user authentication key for the default administrator user ADMIN that is delivered with the CryptoServer.

After shipping the Default Administrator Key ought to be used as soon as possible 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 can find the key for your version (incl. private part) in one of the following keyfiles in your product bundle:

```
\Software\Windows\Administration\key\ADMIN.key ,
\Software\Linux\Administration\key\ADMIN.key ,
```

and on the delivered smartcards.

The following table contains detailed information about the Default Administrator Key.

Type	The Default Administrator Key is an RSA key of at least 1024 bit size.
Generation	<p>The public part of the Default Administrator Key is generated by the manufacturer Utimaco IS GmbH, outside the CryptoServer and is loaded onto the CryptoServer before the CryptoServer has been delivered to you. The default administrator user ADMIN who already has been set up on the CryptoServer must then use this user authentication key to authenticate initial administration tasks. It can also be used to authenticate other specific CryptoServer users.</p> <p>The Default Administrator Key is shown in the output of the csadm GetCardInfo command as follows:</p> <pre>RSA-Key: Utimaco IS GmbH / Init-Dev-1-Key</pre> <p>or</p> <pre>RSA-Key: Utimaco IS GmbH / ADMIN-Key</pre> <p>As an alternative to the csadm command, you can use the Java-based GUI CryptoServer Administration Tool (CAT): Authentication Token > Smartcard > Info > Show Info</p> <p>The key is manufacturer-specific, but it is a default key and therefore neither customer-individual nor CryptoServer-individual.</p> <p>As part of CryptoServer shipment, the manufacturer hands over the key (public and private part) to the customer.</p>
Usage	<p>The key is needed to offer first administration possibilities:</p> <p>The key can be used by default administrator user ADMIN to authenticate the security relevant administration commands (commands, for example, for firmware and user management, reset alarm, set CryptoServer time) as long as the CryptoServer is still in factory default setting.</p> <div data-bbox="563 1659 624 1715">  </div> <p>Because the <code>ADMIN.key</code> is a default administrator key that does not provide any individual protection, we strongly recommend replacing the default authentication token of the 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 as soon as possible.</p>

Life cycle	<p>The key is imported into the CryptoServer by the manufacturer as part of the production process.</p> <p>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 (<code>csadm ChangeUser</code> command, which has to be authenticated by the ADMIN himself) or deleted (<code>csadm DeleteUser</code> command, which can only be performed after sufficient further users with administrator rights have been created before). As an alternative to the <code>csadm</code> commands, you can use CAT > Manage User > Change Token/Password or CAT > Manage User > Delete User.</p> <p>One of these options should be done by the customer as soon as possible after receiving the CryptoServer.</p> <p>A second copy of the public key is stored in the flash device (<code>init.key</code> file). This copy can neither be changed nor deleted, but it is used to restore the default administrator user ADMIN with his Default Administrator Key as authentication token (using the <code>csadm Clear = DEFAULT</code> (<code>ClearFactoryDefaults</code>) command, or CAT > Manage > Clear to Factory Settings). Even in case of an alarm this copy of the key is not deleted.</p>
-------------------	--

Table 13: Details about the Default Administrator Key ADMIN.key

3.7.3 Module Signature Key

The Module Signature Key is used only by the CryptoServer to verify the integrity and authenticity of the firmware modules, i.e., its origin is by the manufacturer. This system key is not used by the customer.

Type	The Module Signature Key is an RSA key of at least 4096-bit size.
Generation	It is generated by the manufacturer, outside the CryptoServer.
Usage	<p>The private part of the Module Signature Key is used by the manufacturer for the signature of any MMC (Module Manufacturer Container). An MMC contains the executable of a CryptoServer firmware module. The signature is used to protect the integrity and authenticity of the MMC during load process, i.e., its origin by the manufacturer.</p> <p>The public part of the Module Signature Key is stored inside the CryptoServer. During firmware load process, this key or the customer-specific Alternative Module Signature Key is used to verify the MMC signature.</p> <p>The private part of the Module Signature Key is used as well by the manufacturer for the signature of any Signed License File. Whenever an <code>*.slf</code> file is loaded, the public part of the Module Signature Key is used to verify the SLF signature, and thus to verify the integrity and authenticity of the SLF, i.e., its origin by the manufacturer.</p>

Life cycle	<p>The private part of the Module Signature Key is stored in a safe environment at the manufacturer's site, where only authorized personnel has access. The public part of the key is imported into the CryptoServer by the manufacturer as part of the production process. It is stored in the file system area of the CryptoServer flash file (<code>mdl.sig.key</code>).</p> <p>The Module Signature Key can neither be exported nor changed nor deleted by the customer.</p> <p>In case of an alarm, or a <code>Clear</code> command being executed, the key is not deleted.</p>
-------------------	--

Table 14: Details about the Module Signature Key

3.7.4 Alternative Module Signature Key



You cannot manage the Alternative Module Signature Key with CAT.

The Alternative Module Signature Key is optional. It has to be used only if the customer would like to develop and load his own CryptoServer firmware modules or if he wants to use firmware modules that do not come from the manufacturer/~Utimaco.

Type	The Alternative Module Signature Key is an RSA key of variable length.
Generation	It is generated by the customer, e.g., by using the <code>csadm</code> command <code>GenKey</code> , outside the CryptoServer. The <code>csadm</code> command <code>SaveKey</code> can be used to store the new key on a smartcard. We recommend creating at least one backup copy of the smartcard/keyfile containing the Alternative Module Signature Key with the <code>csadm</code> command <code>SaveKey</code> . The customer is responsible for using and storing the key pair.
Usage	<p>In case that this key shall be used:</p> <p>The private part of the Alternative Module Signature Key is used for the signature of the MMC (Module Manufacturer Container) of any CryptoServer firmware module that the customer develops of his own. The signature is used to protect the integrity and authenticity of the MMC during load process, i.e., its origin by the customer.</p> <p>The public part of the Alternative Module Signature Key is stored inside the CryptoServer. During firmware load process, this key or the manufacturer's Module Signature Key will be used to verify the MMC signature.</p>

Life cycle	<p>The public part of the Alternative Module Signature Key <code>mdlsigalt.key</code> may be imported on the CryptoServer with the <code>csadm LoadAltMdlSigKey</code> command. In this case this command has to be authenticated by a user with sufficient administrator and user management rights (authentication level 2 in user groups 6 and 7).</p> <p>If this key has been imported into the CryptoServer, it remains stored inside CryptoServer's flash device (file <code>mdlsigalt.key</code>) and will be used for MMC signature verification (in parallel to the manufacturer's Module Signature Key, file <code>mdlsig.key</code>).</p> <p>The key can be changed/overwritten (by loading another <code>mdlsigalt.key</code> file, see above) and deleted (with command <code>DeleteFile</code>).</p> <p>The Alternative Module Signature Key cannot be exported from the CryptoServer, and it is not be deleted in case of an alarm.</p>
-------------------	--

Table 15: Details about the Alternative Module Signature Key

3.7.5 HSM Authentication Key



The HSM Authentication Key cannot be managed with CAT. It is used for Mutual Authentication, see [Mutual Authentication \(p. 38\)](#).

The HSM Authentication Key (HSMAuthKey) is used by the CryptoServer to authenticate itself to a user in the initialization phase of a Secure Messaging session (secure channel).

Type	An HSMAuthKey is a 3072-bit RSA key.
Generation	An HSMAuthKey is generated when a Secure Messaging session is initiated or the <code>csadm GetHSMAuthKey</code> command is performed.
Usage	<p>The private and the public part of the HSMAuthKey are securely stored inside the FLASH memory of the CryptoServer in the <code>authkey.db</code> database. The private part of the HSMAuthKey is encrypted with the Master Key of the CryptoServer and cannot be exported outside the HSM. The public part of the key can be retrieved with the dedicated <code>csadm GetHSMAuthKey</code> command.</p> <p>To use the HSMAuthKey for verifying the authenticity of a Secure Messaging session originating from the CryptoServer, some additional configuration on the administration computer is required. The administration computer is the computer where the software provided with the CryptoServer has been previously installed.</p>

Life cycle	The HSMAuthKey resp. the <code>authkey.db</code> is deleted when an alarm has occurred or a <code>csadm Clear</code> command has been performed. It is a device-specific RSA key that cannot be modified or deleted by any CryptoServer user.
-------------------	---

Table 16: Details about the HSM Authentication Key



An HSM Authentication Key (`authkey.db` file) can neither be backed up nor restored. It is accessible after an MBK rollover.

3.7.6 Firmware Encryption Key



You cannot manage the Firmware Encryption Key with CAT.

By default, firmware modules are not encrypted. Only if the customer would like to store and load encrypted firmware modules, the Firmware Encryption Key shall be used. The usage of the Firmware Encryption Key is optional. It is also called Module Encryption Key.

Type	The Firmware Encryption Key is an RSA key of variable length.
Generation	The generation shall be done by the customer, outside the CryptoServer. The customer is responsible for using and storing the key pair, and for loading the private part of the key into the CryptoServer.

Usage	<p>The usage of the Firmware Encryption Key is optional. It shall be used only in case that the customer wants to load and store encrypted firmware modules. In case that this key shall be used:</p> <p>The public part of the Firmware Encryption Key, which remains outside the CryptoServer, is used by the developer to encrypt the executable firmware module. To be more precise, the public part of the Firmware Encryption Key is used as a Key Encryption Key (KEK) for a 32-byte AES key. The raw binary firmware module itself is encrypted with this AES key. The AES-encrypted firmware module, the KEK-encrypted AES key (encryption according to PKCS#1 block 02) and some key type information are embedded in the MMC structure.</p> <p>The encrypted executable, wrapped in a MMC/MTC structure, may be loaded into the CryptoServer. After being loaded it will be decrypted by using the private part of the Firmware Encryption Key which is stored inside the CryptoServer.</p> <p>Even if the private part of the Firmware Encryption Key is loaded the CryptoServer accepts also cleartext firmware modules for load.</p>
Life cycle	<p>The private part of the Firmware Encryption Key may be imported on the CryptoServer with the <code>csadm LoadFWDecKey</code> command. This command has to be authenticated by a user with sufficient administrator rights (authentication level 2 in user group 6).</p> <p>The private part of the key will be stored in an appropriate key database <code>fw_dec_key.db</code>, encrypted with Master Key. The export of the key is not possible.</p> <p>The key can be changed/overwritten (by loading another private key part with command <code>LoadFWDecKey</code>) and deleted (command <code>DeleteFile</code>). Both commands have to be authenticated by a user with sufficient administrator rights (authentication level 2 in user group 6).</p> <p>The (encrypted) key will also be deleted in case of an alarm and with all Clear commands..</p>

Table 17: Details about the Firmware Encryption Key

3.7.7 Audit Log Signature Key



You cannot manage the Audit Log Signature Key with CAT.



The `csadm GenerateAuditLogKey` command is supported as of the CryptoServer version 4.30 and for ADM version $\geq 3.0.26.0$ and $4.6.1.0 \leq$ SMOS version < 5.0 or SMOS version $\geq 5.6.1.0$.

The Audit Log Signature Key is used to create a signed audit log file. This key is stored in the `auditkey.db` database.

Type	<p>Depending on the <code><mode></code> parameter on creation, the audit log signature key is either an RSA key or an ECDSA key.</p> <ul style="list-style-type: none"> ▪ 3072-bit RSA key with the public exponent 0x10001, PKCS#1 RSA PSS signature with SHA-256 and a 32-byte salt length ▪ ECDSA with NIST P-256 curve and SHA-256
Generation	<p>The Audit Log Signature Key can be created only by performing the <code>csadm GenerateAuditLogKey</code> command.</p>
Usage	<p>The key can be used as input for the <code>csadm GetSignedAuditLog</code> and <code>csadm VerifySignedAuditLog</code> commands.</p> <p>If you want to use a different Audit Log Signature Key, use the <code>csadm DeleteFile=</code> command to delete the <code>auditkey.db</code> database/key. The database is also deleted if an alarm has occurred, or if an External Erase or a clear has been performed.</p> <p>The <code>auditkey.db</code> database can only contain one Audit Log Signature Key.</p> <p>If you want to use a different Audit Log Signature Key, use the <code>csadm DeleteFile=</code> command to delete the database/key and perform the <code>csadm GenerateAuditLogKey</code> command to generate a new key in a new <code>auditkey.db</code> database.</p> <p>Consider that if you delete the <code>auditkey.db</code> database, you cannot sign any audit log files with the key stored in this database anymore and that audit log files previously signed with this key cannot be verified unless you have backed up the <code>auditkey.db</code> database.</p> <p>The <code>auditkey.db</code> database can be backed up and restored using the <code>csadm BackupDatabase</code> and <code>csadm RestoreDatabase</code> commands.</p> <p>If the device is part of a cluster, ensure that the generated Audit Log Signature Key is distributed in the entire cluster by using the <code>csadm BackupDatabase</code> command and the <code>csadm DatabaseRestore</code> command.</p> <p>The <code>csadm GenerateAuditLogKey</code> command is supported in Operational Mode only. Perform the <code>csadm GetState</code> command to retrieve the mode of the device.</p>

Life cycle	<p>The Audit Log Signature Key is permanently deleted when an alarm has been triggered, a clear has been performed or a <code>csadm DeleteFile=auditkey.db</code> command has been performed. Therefore, you must store a backup copy of this key outside the device performing the <code>csadm BackupDatabase</code> command and restore it.</p> <p>The Audit Log Signature Key is permanently deleted when an alarm has been triggered or a clear has been performed.</p>
-------------------	---

Table 18: Details about the audit log signature key

3.7.8 Master Backup Key (MBK)

The CryptoServer provides secure storage for secret data and private keys. This includes, for example, the keys used by the PKCS#11, CSP/CNG, JCE and other interfaces. As any perceived attack causes the CryptoServer to permanently delete all the sensitive data and private keys stored on it, we strongly recommend backing up this data or keys so they can be reimported (restored) once the alarm has been resolved. To ensure this backup copy of the sensitive data or private keys can be stored securely even outside the CryptoServer, it is encrypted with the Master Backup Key (MBK).



Secret data and cryptographic keys stored inside the CryptoServer are encrypted with the Master Key and not with the MBK.

Type	The MBK is an AES key with a key length of 16, 24 or 32 byte.
Generation	<p>For generating an MBK you can use the <code>csadm MBKGenerateKey</code> command as described in <i>MBKGenerateKey</i> in the <i>CryptoServer – csadm Manual</i> or you can use CAT as described in <i>Generating an MBK</i> in the <i>CryptoServer - CAT Manual</i>. The MBK is generated by the MBK firmware module. It is split into n key shares that are exported and stored on n smartcards protected with a PIN or in n keyfiles protected with a password. Only m out of n key shares are needed to reconstruct the MBK. Splitting the MBK into several key shares is one of CryptoServer's most important security functions because it enforces the MBK to be used by several people (at least two), who then share responsibility for its use.</p> <p>If you generate the MBK using the menu control buttons of a CryptoServer LAN you store the MBK only on a smartcard, see the <i>CryptoServer LAN V5 – Administration Manual</i>.</p>

Usage

The CryptoServer can store up to 256 MBKs in MBK slot number 0 to 255. However, the default MBK slot number on the CryptoServer is slot 3. This MBK slot is used by our applications. The MBK slot number must not be confused with the PKCS#11 slot number. The purpose of an MBK is to protect the backups and externally stored secret data and private keys from unauthorized access even if they have to be stored elsewhere. These backup copies of secret data or private keys are normally stored on the CryptoServer and are used for user authentication.

The MBK firmware module provides the following functions for MBK management:

- **Generation and storage of an MBK**
This function generates a 256-bit (32-byte) AES key as an MBK within the secure environment of the CryptoServer. To ensure the highest possible levels of quality and "randomness" the hardware random number generator of the CryptoServer is used for the MBK generation. The generated MBK is then split into shares and stored on smartcards or in keyfiles, but it is not stored yet on the CryptoServer. The MBK must be imported into the CryptoServer in a separate step.
- **Import of an MBK**
This function imports an MBK from the smartcards or keyfiles to the CryptoServer. For backward compatibility, importing of 128-bit (16-byte) DES keys is also supported (for csadm versions < 2.5.3 and CAT versions < 2.2.5.2 only).

There are two methods for managing an MBK:

- **Local MBK Management**
In this case, the PIN pad is connected directly to the CryptoServer. As the CryptoServer writes the MBK shares directly to the smartcards, they are neither stored in the host computer's RAM nor transferred via a network. However, if you use this method, you cannot store the MBK in a keyfile. Local MBK management can be performed by both the csadm command-line tool and the CryptoServer LAN menu control buttons.
- **Remote MBK Management**
In this situation, the MBK can be managed over a network on a CryptoServer LAN. After the MBK has been generated on the CryptoServer, its shares are transferred to the host computer, where the administration tools have been installed. Afterwards, they are stored on smartcards or in keyfiles. If the MBK shares are stored on smartcards, the PIN pad is connected to the host computer.

For detailed information on managing an MBK, see:

- If you use **csadm**, see *Commands for Managing the Master Backup Keys* in the *CryptoServer – csadm Manual*.
- If you use the CryptoServer Administration Tool (**CAT**), see *Generating an MBK* in the *CryptoServer - CAT Manual*.
- If you use the menu control buttons on the front panel of the CryptoServer LAN V5, see *Performing MBK Management on the CryptoServer LAN* in the *CryptoServer LAN V5 – Administration Manual*.

Life cycle	<p>The MBK is permanently deleted when an alarm has been triggered. Therefore, you must store a backup copy of this key outside the CryptoServer, for example:</p> <ul style="list-style-type: none"> ▪ On a set of at least two smartcards protected with a PIN. ▪ On a secure USB flash drive or a computer as a set of keyfiles that are protected with a strong password. For security reasons, each MBK share must be assigned to a different person and stored on a different place. <p>The authorized users with the Administrator role shall ensure that the smartcards where the MBK-shares are stored on or the MBK keyfiles and the backup copies of the MBK are securely stored and appropriately protected against unauthorized access, loss and damage. The PIN codes for these smartcards and the passwords for the keyfiles must be kept secret.</p>
-------------------	--

Table 19: Details about the Master Backup Key

3.7.9 Tenant Backup Keys (TBK)

As of SecurityServer/CryptoServer SDK 4.10, the Master Backup Key can be additionally used to derive individual backup keys, called Tenant Backup Keys (TBK). The TBKs ensure that key backups and keys stored outside a CryptoServer, and used for example by the PKCS#11, CSP/CNG or CXI API, are protected in an individual way. For example, in a cloud environment the CryptoServer serves as the root of trust for more than one PKCS#11 applications simultaneously. Each application uses a single PKCS#11 slot on the CryptoServer for its individual key management. The CryptoServer considers each of these applications as a tenant. Thus, the key backup of a slot is protected with the slot-individual TBK and can only be accessed by the authorized users of that PKCS#11 slot.



csadm and the Java-based GUI CryptoServer Administration Tool (CAT) always use the CryptoServer's MBK. They do not use TBKs.

The individual protection of external keys and key backups by using TBKs is only provided by P11CAT, p11tool2 and cxitool, and requires individual configuration.

Cryptographic keys that are not assigned to any key group never use a TBK but the MBK.

The expressions "PKCS#11 slot", "key group", "cryptographic key group" and "CXI key group" are equivalent to each other.



Secret data and cryptographic keys stored inside the CryptoServer are encrypted with the Master Key and not with the MBK or the TBK.

Type

The TBK is an AES key with a key length of 16, 24 or 32 byte.

Generation

If you want to use a group-specific Tenant Backup Key (TBK) instead of a Master Backup Key (MBK), the following steps must be performed.

Preparation step (optional): Set a password and initiate hashing this password

A TBK is derived from an MBK. Consider using a hashed password to derive a TBK from an MBK. To set a password and to initiate hashing this password, perform one of the following steps:

- For PKCS#11 applications:
Set `CKA_CFG_SLOT_BACKUP_PASS_HASH` to the password by using the `p11tool2 SecureSlotPass` command.
Example: `p11tool2 Slot=0 LoginSO=123456 SecureSlotPass=123456`
To verify the result, perform the `p11tool2 ... GetSlotConfig=CKA_CFG_SLOT_BACKUP_PASS_HASH` command. This command only shows whether the password has been set or not. As an alternative, perform the `p11tool2 ... GetSlotConfig=*` command. This command shows all configuration objects.
The password cannot be set by opening **P11CAT > Config Management > Slot Configuration > CKA_CFG_SLOT_BACKUP_PASS_HASH**. It only shows whether the password has been set or not.
- For applications using Utimaco's CXI interface:
Set `SecureGroupPass` to the password by using `cxitool`, for example, for the `SLOT_0000` group:
`cxitool Dev=3001@127.0.0.1 LogonPass=SO_0000,123456 Group=SLOT_0000 SecureGroupPass=123456`
To verify the result, perform the `cxitool ... Group=... GetConfig` command.

There is no way to set a password and to initiate hashing this password for all PKCS#11 slots/CXI key groups in one step.



If a passphrase shall be used for the derivation of TBKs, it has to be defined prior to enabling the usage of TBKs (see generation step below). The usage of TBKs in turn must be configured before the corresponding PKCS#11 slot/CXI group on the CryptoServer gets operational. Otherwise, existing external keys and key backups become inaccessible.



Changing the `CKA_CFG_SLOT_BACKUP_PASS_HASH` attribute for a slot that is currently in use causes previously generated external keys and their backups to become inaccessible. If you use SecurityServer/CryptoServer SDK 4.10 when creating the external key and their backups and you try to restore them with a changed individual passphrase, the error message "invalid mac of key blob" (error code: 0xB0680026) is created. This applies as well if you have upgraded to SecurityServer/CryptoServer SDK 4.20 or later before trying to restore the external key or key backups. However, if you upgrade to SecurityServer/CryptoServer SDK 4.20 before creating the external key or key backup and you try to restore them with a changed individual passphrase, the error message "wrong TBK passphrase for this key blob" (error code: 0xB0680081) is created.



If external keys and backups have been previously created with SecurityServer 4.10 or CryptoServer SDK 4.10 or earlier and if these external keys and backups have been created by using an MBK, using a TBK now makes these previously created external keys and backups inaccessible.

Generation step: Enable the usage of TBKs and generate a TBK

TBKs are not used by default. The CryptoServer can be configured on demand to use them. This is done by setting dedicated configuration attributes. Further individualization of the TBKs can be achieved by defining individual passphrases for their derivation (see the preparation step above).

The usage of TBKs can be enabled globally for the CryptoServer or only for a specific CryptoServer PKCS#11 slot/CXI key group:

- The global use of TBKs means that TBKs are used for every key and each PKCS#11 slot/CXI key group. To enforce this, the CryptoServer requires one or more authenticated CryptoServer users with user management permissions (min. authentication status 20000000) or the ADMIN to set the following global configuration attribute:
 - For PKCS#11 applications:
 Set `CKA_CFG_SECURE_SLOT_BACKUP` to `CK_TRUE` by using the `p11tool2 SetGlobalConfig` command or **P11CAT > Config Management > Global Configuration > CKA_CFG_SECURE_SLOT_BACKUP > CK_TRUE**.
 Example: `p11tool2 Login=ADMIN,C:/keys/init_prv.key SetGlobalConfig=CKA_CFG_SECURE_SLOT_BACKUP,CK_TRUE`
 To verify the result, perform the `p11tool2 GetGlobalConfig` command.

- For applications using Utimaco's CXI interface:
Set `SecureGroupBackup` to true by using `cxitool`, for example:
`cxitool LogonSign=ADMIN,"C:\Program Files\Utimaco\SecurityServer\Administration\ADMIN.key"`
`SetConfig=SecureGroupBackup,true`
To verify the result, perform the `cxitool GetConfig` command without using the `Group` parameter.
- The PKCS#11 slot/key group-specific use of TBKs means that TBKs can be used only for selected PKCS#11 slots/CXI key groups on the CryptoServer. To enforce this, the CryptoServer requires the corresponding PKCS#11 slot/CXI key group administrator(s), so-called Security Officer(s) (SO), with permissions in the user group 2 (min. authentication status 00000200) to set the configuration attributes mentioned above for the specific PKCS#11 slot(s)/CXI key group.
 - For PKCS#11 applications:
Set `CKA_CFG_SECURE_SLOT_BACKUP` to `CK_TRUE` by using the `p11tool2 SetSlotConfig` command or **P11CAT > Config Management > Slot Configuration > CKA_CFG_SECURE_SLOT_BACKUP > CK_TRUE**.
Example: `p11tool2 LoginSO=ask`
`SetSlotConfig=CKA_CFG_SECURE_SLOT_BACKUP,CK_TRUE`
To verify the result, perform the `p11tool2 GetSlotConfig` command.
 - For applications using Utimaco's CXI interface
Set `SecureGroupBackup` to true by using `cxitool`, for example, initiating the usage of a TBK for the `SLOT_0000` group:
`cxitool Dev=3001@127.0.0.1 LogonPass=SO_0000,123456`
`Group=SLOT_0000 SetConfig=SecureGroupBackup,true`
To verify the result, perform the `cxitool Group=... GetConfig` command.



If you want to use a TBK, the following applies: If you use SecurityServer/CryptoServer SDK 4.10, create an external key or key backup by using an MBK, then enable the usage of TBKs by setting the `CKA_CFG_SECURE_SLOT_BACKUP` configuration attribute to the `CK_TRUE` value, trying to restore the external key or key backup fails and the external key and key backups become inaccessible. The error message "invalid mac of key blob" (error code: 0xB0680026) is created. This applies as well if you have upgraded to SecurityServer/CryptoServer SDK 4.20 or later before trying to restore the external key or key backups. However, if you upgrade to SecurityServer/CryptoServer SDK 4.20 before creating the external key or key backup using an MBK, restoring them with a TBK succeeds.



Ensure that you have enabled the usage of TBKs according to your security policy before your CryptoServer production environment gets operational. Otherwise, existing external keys and key backups become inaccessible.

Configuring the CryptoServer to use TBKs as described above implicitly generates a TBK in a cache of the CryptoServer. This TBK is derived from the MBK that currently is in MBK slot 3. Note down the name of this MBK because if you generate an external key or a key backup with this TBK and you want to restore this external key or key backup, the same MBK must be in MBK slot 3. The only exception from this rule is after an MBK rollover. In this case, the same MBK must be in MBK slot ≥ 3 . The default MBK slot number on the CryptoServer where the MBK is stored on import is 3. The MBK slot number must not be confused with the PKCS#11 slot number.

A TBK or TBK shares cannot be stored in a file on a file system or on a smartcard.

There is no explicit TBK generating command that a user can perform. Neither can a TBK be generated by pressing a menu control button of a CryptoServer LAN. A TBK is always generated implicitly as described above.

Usage

As already mentioned above, the individual protection of external keys and key backups by using TBKs is only provided by P11CAT, p11tool2 and cxitool, and requires individual configuration.

The purpose of a TBK is to protect PKCS#11 slot-specifically the backups and externally stored secret data and private keys from unauthorized access even if they have to be stored elsewhere. These backup copies of secret data or private keys are normally stored on the CryptoServer and are used for user authentication.

The following actions/commands use a TBK:

Backing up/restoring a cryptographic key or configuration

- For PKCS#11 applications
 - `p11tool2`
 - `p11tool2 BackupInternalKeys`
 - `p11tool2 RestoreInternalKeys`
 - `p11tool2 BackupExternalKeys`
 - `p11tool2 RestoreExternalKeys`
 - `p11tool2 BackupConfig`
 - `p11tool2 RestoreConfig`
 - P11CAT
 - P11CAT > **Backup/Restore > Backup/Restore Keys > Backup Internal Keys**
 - P11CAT > **Backup/Restore > Backup/Restore Keys > Backup External Keys**
 - P11CAT > **Backup/Restore > Backup/Restore Keys > Restore Key Backup to External Key Store**
 - P11CAT > **Backup/Restore > Backup/Restore Keys > Restore Key Backup to Internal Key Store**
 - P11CAT > **Backup/Restore > Backup/Restore Config > Backup Slot Configuration Object**
 - P11CAT > **Backup/Restore > Backup/Restore Config > Restore Slot Configuration Object**
- For applications using Utimaco's CXI interface
 - `cxitool BackupKey`

- `cxitool RestoreKey`

Generating external cryptographic keys

- For PKCS#11 applications
As a precondition for using external cryptographic keys in p11tool2 or P11CAT, `KeysExternal = true` must have been set in the `cs_pkcs11_R3.cfg` file.
 - p11tool2
 - `p11tool2 GenerateKeyPair`
 - `p11tool2 GenerateKey`
 - To enforce external cryptographic keys, perform the `p11tool2 ... SetGlobalConfig=CKA_CFG_ENFORCE_EXT_KEYS,CK_TRUE` command or the `p11tool2 ... SetSlotConfig=CKA_CFG_ENFORCE_EXT_KEYS,CK_TRUE` command.
 - To verify the result, perform one of the following commands:
 - `p11tool2 ... GetGlobalConfig=CKA_CFG_ENFORCE_EXT_KEYS`
 - `p11tool2 ... GetGlobalConfig=*`
 - `p11tool2 ... GetSlotConfig=CKA_CFG_ENFORCE_EXT_KEYS`
 - `p11tool2 ... GetSlotConfig=*`
 - P11CAT
 - P11CAT > **Object Management > Generate > Generate Key > Generate**
 - P11CAT > **Object Management > Generate > Generate Key Pair > Generate**
 - P11CAT > **Object Management > Generate > Generate Key from File > Generate**
 - P11CAT > **Object Management > Generate > Generate Key Pair from File > Generate**
 - To enforce external cryptographic keys, perform P11CAT > **Config Management > Global Configuration > CKA_CFG_ENFORCE_EXT_KEYS > CK_TRUE** or P11CAT > **Config Management > Slot Configuration > CKA_CFG_ENFORCE_EXT_KEYS > CK_TRUE**.
- For applications using Utimaco's CXI interface
 - `cxitool ... Group=... KeyStore=<filename> GenerateKey`
To enforce external cryptographic keys, the `cxitool ... Group=... SetConfig=EnforceExtKeys,true` command must have been performed. To verify the result, perform the `cxitool ... Group=... GetConfig` command.


Life cycle	<p>A TBK is only available in a cache of the CryptoServer. A TBK is derived from the MBK available in MBK slot 3. If the CryptoServer is restarted (for example, due to the <code>csadm Restart</code> command) or another MBK is imported into MBK slot 3 (for example, due to the <code>csadm ... MBKImportKey=3</code> command), the TBK is deleted and a new TBK is generated in the cache. The only exception from this rule is after an MBK rollover. In this case, the old MBK the TBK has been derived from must be in MBK slot ≥ 3. Each time the CryptoServer is started anew, the TBK is derived anew from the MBK in MBK slot 3 (or ≥ 3 after an MBK rollover) and is stored in the cache.</p> <hr/> <div style="display: flex; align-items: center;">  <p>We highly recommend storing a backup copy of the MBK the TBK has been derived from outside the CryptoServer. This avoids the external keys and key backups that have been created using this TBK to become inaccessible.</p> </div> <hr/> <p>Examples to store a backup copy of an MBK:</p> <ul style="list-style-type: none"> ▪ On a set of at least two smartcards protected with a PIN. ▪ On a secure USB flash drive or a computer as a set of keyfiles that are protected with a strong password. For security reasons, each MBK share must be assigned to a different person and stored on a different place. <p>The authorized users with the Administrator role shall ensure that the smartcards where the MBK-shares are stored on or the MBK keyfiles and the backup copies of the MBK are securely stored and appropriately protected against unauthorized access, loss and damage. The PIN codes for these smartcards and the passwords for the keyfiles must be kept secret.</p>
-------------------	---

Table 20: Details about the Tenant Backup Key

3.8 Cryptographic Interfaces

This section gives a brief introduction to the CryptoServer's most important cryptographic interfaces. The corresponding libraries for these interfaces are provided on the SecurityServer product CD.

3.8.1 Cryptographic eXtended Interface (CXI)

The Cryptographic eXtended Interface (CXI) is a proprietary interface developed by Utimaco IS GmbH. CXI provides all CryptoServer cryptographic functions via a user-friendly interface, which has a low protocol overhead and is easy to integrate into customer-specific applications.

You can use the Cryptographic User user to administer all other cryptographic interfaces described later on in this section via this CXI interface. You can control all other cryptographic interfaces and their functions within the CryptoServer via the CXI interface. In

this respect the CXI interface plays a more prominent role than the other cryptographic interfaces.

On the CryptoServer itself, CXI functions are provided by the CXI firmware module, which is a component of the SecurityServer firmware package. You can use the CAT administration program to create a Cryptographic User which functions both as a CXI user and as a user for all other cryptographic interfaces. The CAT provides the appropriate permission profile for the Cryptographic User.

3.8.2 PKCS#11

PKCS#11 is the standard used to define a programming interface for security tokens such as smartcards or hardware security modules. From the PKCS#11 viewpoint, a security token is a device that stores objects and can perform cryptographic functions. These objects may be keys or certificates. In addition, the objects can each have different attributes, which not only define how you handle them but may also limit the areas in which they can be used.

With version 3.20 and later of the SecurityServer/CryptoServer SDK product CD only the PKCS#11 R2 (PKCS#11 R3 as of SecurityServer/CryptoServer SDK 4.40.0) implementation is supplied for which a dedicated administration tool P11CAT and a separated manual are available.

The PKCS#11 functions on the CryptoServer are provided by the CXI firmware module which is also a part of the SecurityServer firmware package.

3.8.3 Microsoft CryptoAPI and Cryptography API: Next Generation (CNG)

Software developers can use the Microsoft cryptographic application programming interface (CryptoAPI) to call cryptographic functions from their Windows-based applications. These functions are provided by what is known as the Cryptographic Service Provider (CSP).

The CryptoAPI contains all functions necessary for encrypting and decrypting data (with both symmetrical and asymmetrical keys) and for using and managing digital certificates for authentication purposes.

Cryptography API: Next Generation (CNG) is Microsoft's latest cryptographic platform. In CNG the provider concept has been extended further than in CryptoAPI. CNG also supports newer cryptographic algorithms, for example elliptic curves. At present, both CryptoAPI and CNG are being supported, to enable existing applications to be upgraded step-by-step.

The CSP/CNG cryptographic interface is accessed via the CXI firmware module. Depending on the CryptoServer CSP version different configuration settings are necessary:

- For CryptoServer CSP 1.x provided on SecurityServer and CryptoServer SDK product CD up to 4.01, you do not need to set up a Cryptographic User for CSP/CNG on the

CryptoServer. If you have installed the device or the machine via the Control panel applet, a user is created on the CryptoServer automatically. This machine, or the CSP/CNG user, has the same authentication status as the Cryptographic User in the CXI firmware module.

- For CryptoServer CSP 2.x provided on SecurityServer and CryptoServer SDK product CD version 4.10 and later, the CSP and CNG interfaces are configured via a dedicated configuration file, and a CSP/CNG user, with the same permissions as the Cryptographic User (min. 00000002) must be created manually.

The CryptoAPI and CNG functions are made available to the applications that call them by a CryptoServer Cryptographic Service Provider (CSP) which serves both these interfaces. Depending on the version of the SecurityServer and the CryptoServer SDK product CD, not only the CryptoServer CSP itself is provided, but also a dedicated configuration file, `cs_cng.cfg`, that can be used to configure the CSP/CNG cryptographic interface. Additionally, a key management tool is available: The command-line utility CNG tool for managing CNG cryptographic keys.

3.8.4 Java Cryptography Extension (JCE)

The Java Cryptography Extension (JCE) is a collection of Java packages that implement cryptographic procedures for Java applications. These include encrypting and decrypting data and generating keys. These functions are made available by the JCE provider.

On the CryptoServer, these JCE functions are made available by the CXI firmware module which is a component of the SecurityServer firmware package. You can use the CAT administration tool to create a Cryptographic User that is also a user for JCE. The CAT provides the appropriate authorization profile for the Cryptographic User.

3.8.5 OpenSSL

OpenSSL is a free implementation of the SSL/TLS protocol for Open Source environments which also offers a range of more specialized functions for certificate management and for different cryptographic functions. These functions are grouped together in the OpenSSL command-line tool.

The integration of OpenSSL into the CryptoServer is done indirectly via an OpenSSL PKCS#11 wrapper. If you want to integrate OpenSSL indirectly via a PKCS#11 wrapper, use the PKCS#11 administration tool P11CAT to initialize a PCKS#11 token or slot.

3.8.6 Extensible Key Management (EKM)

An SQL Server provides functions for data encryption and EKM (Extensible Key Management). The Microsoft Cryptography API service provider is used for encryption and key generation. Encryption keys for encrypting data and keys are created in temporary key containers, and must be exported by the service provider before they can be saved in the database.

This approach enables SQL Server to be used for key management with an encryption key hierarchy and key security.

In CryptoServer, the EKM functions are provided by the CXI firmware module which is part of the SecurityServer firmware package.

Use the CAT administration program to create a Cryptographic User which is also a user for EKM. The CAT provides the appropriate authorization profile for the Cryptographic User.

3.9 Supported Algorithms

3.9.1 CXI API

The CXI API supports the following algorithms:

<i>Algorithm</i>	<i>Key length [bit]</i>	<i>Mode</i>	<i>Chaining mode</i>	<i>Padding</i>
DES	56, 112, 168	Encrypt, Decrypt	ECB, CBC	None, PKCS#5, ISO7816, Random
		MAC	CBC, CBC-Retail	None, Zero, PKCS#5, ISO7816, Random
		Wrap, Unwrap	ECB, CBC	None, PKCS#5, ISO7816, Random
AES	128, 192, 256	Encrypt, Decrypt	ECB, CBC, GCM, OFB, CTR, CCM, KWP	None, PKCS#5, ISO7816, Random
		MAC	CBC	None, Zero, PKCS#5, ISO7816, Random
			CMAC	None, Zero, PKCS#5, ISO7816, Random
			GMAC	None
		Wrap, Unwrap	ECB, CBC, OFB, KWP	None, PKCS#5, ISO7816, Random
RSA	512 .. 16384 (delta = 1 bit)	Encrypt, Decrypt	-	None, PKCS#1-v1_5, PKCS#1-OAEP

Algorithm	Key length [bit]	Mode	Chaining mode	Padding
		Sign, Verify	-	PKCS#1-v1_5, X9.31, PKCS#1-PSS
		Wrap, Unwrap	-	None, PKCS#1-v1_5, PKCS#1-OAEP
ECDSA ECDH	Brainpool: 160 ... 512 NIST: 163 .. 571 Secp: 112 .. 571 FRP256v1	Encrypt, Decrypt (ECIES)	-	-
		Sign, Verify (according to ANSI X9.62; format: Raw or ASN.1)	-	-
EdDSA	curve25519	Sign, Verify	-	-
DSA, DH DH_PKCS see info box below table	P: ≥ 512 Q: ≥ 160	Sign, Verify	-	-
RAW (Generic Secret)	80 ... 8192 (delta = 8 bit)	Hash Computation (HMAC) Sign, Verify (HMAC) Derive Key	-	-

Table 21: Key algorithms supported by the CXI API



DH_PKCS mandatorily only contains the prime P and the generator G as domain parameters. Q is optional. DH mandatorily contains all three DSA domain parameters P, Q and G.

If random padding bytes are needed, always the deterministic random number generator (DRNG) is used.

The following hash algorithms are supported:

Algorithm	Hash length [bit]
MD5 (non-FIPS only)	128
RIPEMD-160 (non-FIPS only)	160
SHA-1	160

<i>Algorithm</i>	<i>Hash length [bit]</i>
SHA-224	224
SHA-256	256
SHA-384	384
SHA-512	512
SHA3-224	224
SHA3-256	256
SHA3-384	384
SHA3-512	512

Table 22: Hash algorithms supported by the CXI API

3.9.2 CSP/CNG API

The CSP/CNG API supports the following algorithms:

<i>Algorithm</i>	<i>Key length/Curves [bit]</i>	<i>Interface</i>	
		<i>CSP</i>	<i>CNG</i>
DES	56, 112 and 168	✓	✓
AES	128, 192 and 256	✓	✓
RSA	512 – 16.384 (delta = 8 bit)	✓	✓
ECDSA	NIST-P256, NIST-P384, NIST-P521	✗	✓
ECDH	NIST-P256, NIST-P384, NIST-P521	✗	✓

Table 23: Key algorithms supported by the CSP/CNG API

<i>Algorithm</i>	<i>Key length /Curves [bit]</i>	<i>Interface</i>	
		<i>CSP</i>	<i>CNG</i>
MD5 (non-FIPS only)	128	✓	✓
RIPEMD-160 (non-FIPS only)	160	✓	✓
SHA-1	160	✓	✓
SHA-224	224	✓	✓
SHA-256	256	✓	✓
SHA-384	384	✓	✓
SHA-512	512	✓	✓

Table 24: Hash algorithms supported by the CSP/CNG API

3.9.3 B.1 PKCS#11 API

3.9.3.1 PKCS#11 Defined Mechanisms

The table shows the PKCS#11 standard mechanisms provided by the CryptoServer and the PKCS#11 API functions supporting them:

<i>Mechanism</i>	<i>Functions</i>						
	<i>Encrypt & Decrypt</i>	<i>Sign & Verify</i>	<i>SR & VR¹</i>	<i>Digest</i>	<i>Gen. Key/Key Pair</i>	<i>Wrap & Unwrap</i>	<i>Derive</i>
CKM_RSA_PKCS_OAEP	✓ ²					✓	
CKM_RSA_PKCS_KEY_PAIR_GEN					✓		
CKM_RSA_X9_31_KEY_PAIR_GEN					✓		
CKM_RSA_PKCS	✓ ²	✓ ²	✓			✓	
CKM_RSA_PKCS_PSS		✓ ²					
CKM_RSA_X_509	✓ ²	✓ ²	✓				
CKM_RSA_X9_31		✓ ²					
CKM_MD5_RSA_PKCS		✓					
CKM_SHA1_RSA_PKCS		✓					
CKM_SHA224_RSA_PKCS		✓					
CKM_SHA256_RSA_PKCS		✓					
CKM_SHA384_RSA_PKCS		✓					
CKM_SHA512_RSA_PKCS		✓					
CKM_SHA3_224_RSA_PKCS*		✓					
CKM_SHA3_256_RSA_PKCS*		✓					
CKM_SHA3_384_RSA_PKCS*		✓					
CKM_SHA3_512_RSA_PKCS*		✓					
CKM_RIPEMD160_RSA_PKCS		✓					
CKM_SHA1_RSA_PKCS_PSS		✓					
CKM_SHA224_RSA_PKCS_PSS		✓					
CKM_SHA256_RSA_PKCS_PSS		✓					
CKM_SHA384_RSA_PKCS_PSS		✓					
CKM_SHA512_RSA_PKCS_PSS		✓					
CKM_SHA3_224_RSA_PKCS_PSS*		✓					

Mechanism	Functions						
	Encrypt & Decrypt	Sign & Verify	SR & VR¹	Digest	Gen. Key/Key Pair	Wrap & Unwrap	Derive
CKM_SHA3_256_RSA_PKCS_PSS*		✓					
CKM_SHA3_384_RSA_PKCS_PSS*		✓					
CKM_SHA3_512_RSA_PKCS_PSS*		✓					
CKM_SHA1_RSA_X9_31		✓					
CKM_DSA		✓ ²					
CKM_DSA_SHA1		✓					
CKM_DSA_SHA224		✓					
CKM_DSA_SHA256		✓					
CKM_DSA_SHA384		✓					
CKM_DSA_SHA512		✓					
CKM_DSA_SHA3_224*		✓					
CKM_DSA_SHA3_256*		✓					
CKM_DSA_SHA3_384*		✓					
CKM_DSA_SHA3_512*		✓					
CKM_DSA_KEY_PAIR_GEN					✓		
CKM_DSA_PARAMETER_GEN					✓		
CKM_DH_PKCS_KEY_PAIR_GEN					✓		
CKM_DH_PKCS_DERIVE							✓
CKM_X9_42_DH_KEY_PAIR_GEN					✓		
CKM_X9_42_DH_PKCS_PARAMETER_GEN					✓		
CKM_X9_42_DH_DERIVE							✓
CKM_EC_KEY_PAIR_GEN (CKM_ECDSA_KEY_PAIR_GEN)					✓		
CKM_ECDSA		✓ ²					
CKM_ECDSA_SHA1		✓					
CKM_ECDSA_SHA224*		✓					
CKM_ECDSA_SHA256*		✓					
CKM_ECDSA_SHA384*		✓					
CKM_ECDSA_SHA512*		✓					
CKM_ECDH1_DERIVE							✓
CKM_ECDH1_COFACTOR_DERIVE							✓
CKM_GENERIC_SECRET_KEY_GEN					✓		

Mechanism	Functions						
	Encrypt & Decrypt	Sign & Verify	SR & VR¹	Digest	Gen. Key/Key Pair	Wrap & Unwrap	Derive
CKM_AES_KEY_GEN					✓		
CKM_AES_ECB	✓					✓	
CKM_AES_CBC	✓					✓	
CKM_AES_CBC_PAD	✓					✓	
CKM_AES_CTR	✓						
CKM_AES_CCM	✓						
CKM_AES_GCM	✓						
CKM_AES_MAC_GENERAL		✓					
CKM_AES_MAC		✓					
CKM_AES_CMAC		✓					
CKM_AES_GMAC*		✓					
CKM_AES_OFB	✓					✓	
CKM_AES_KEY_WRAP	✓ ²					✓	
CKM_AES_KEY_WRAP_PAD	✓ ²					✓	
CKM_AES_KEY_WRAP_KWP	✓ ²					✓	
CKM_DES_KEY_GEN					✓		
CKM_DES_ECB	✓					✓	
CKM_DES_CBC	✓					✓	
CKM_DES_CBC_PAD	✓					✓	
CKM_DES_MAC_GENERAL		✓					
CKM_DES_MAC		✓					
CKM_DES2_KEY_GEN					✓		
CKM_DES3_KEY_GEN					✓		
CKM_DES3_ECB	✓					✓	
CKM_DES3_CBC	✓					✓	
CKM_DES3_CBC_PAD	✓					✓	
CKM_DES3_MAC_GENERAL		✓					
CKM_DES3_MAC		✓					
CKM_DES_ECB_ENCRYPT_DATA							✓
CKM_DES_CBC_ENCRYPT_DATA							✓
CKM_DES3_ECB_ENCRYPT_DATA							✓
CKM_DES3_CBC_ENCRYPT_DATA							✓
CKM_AES_ECB_ENCRYPT_DATA							✓

<i>Mechanism</i>	<i>Functions</i>						
	<i>Encrypt & Decrypt</i>	<i>Sign & Verify</i>	<i>SR & VR¹</i>	<i>Digest</i>	<i>Gen. Key/Key Pair</i>	<i>Wrap & Unwrap</i>	<i>Derive</i>
CKM_AES_CBC_ENCRYPT_DATA							✓
CKM_MD5				✓			
CKM_MD5_HMAC_GENERAL		✓					
CKM_MD5_HMAC		✓					
CKM_MD5_KEY_DERIVATION							✓
CKM_SHA_1				✓			
CKM_SHA_1_HMAC_GENERAL		✓					
CKM_SHA_1_HMAC		✓					
CKM_SHA1_KEY_DERIVATION							✓
CKM_SHA224				✓			
CKM_SHA224_HMAC_GENERAL		✓					
CKM_SHA224_HMAC		✓					
CKM_SHA224_KEY_DERIVATION							✓
CKM_SHA256				✓			
CKM_SHA256_HMAC_GENERAL		✓					
CKM_SHA256_HMAC		✓					
CKM_SHA256_KEY_DERIVATION							✓
CKM_SHA384				✓			
CKM_SHA384_HMAC_GENERAL		✓					
CKM_SHA384_HMAC		✓					
CKM_SHA384_KEY_DERIVATION							✓
CKM_SHA512				✓			
CKM_SHA512_HMAC_GENERAL		✓					
CKM_SHA512_HMAC		✓					
CKM_SHA512_KEY_DERIVATION							✓
CKM_SHA3_224*				✓			
CKM_SHA3_224_HMAC_GENERAL*		✓					
CKM_SHA3_224_HMAC*		✓					
CKM_SHA3_224_KEY_DERIVATION*							✓
CKM_SHA3_256*				✓			
CKM_SHA3_256_HMAC_GENERAL*		✓					
CKM_SHA3_256_HMAC*		✓					
CKM_SHA3_256_KEY_DERIVATION*							✓

Mechanism	Functions						
	Encrypt & Decrypt	Sign & Verify	SR & VR¹	Digest	Gen. Key/Key Pair	Wrap & Unwrap	Derive
CKM_SHA3_384 [*]				✓			
CKM_SHA3_384_HMAC_GENERAL [*]		✓					
CKM_SHA3_384_HMAC [*]		✓					
CKM_SHA3_384_KEY_DERIVATION [*]							✓
CKM_SHA3_512 [*]				✓			
CKM_SHA3_512_HMAC_GENERAL [*]		✓					
CKM_SHA3_512_HMAC [*]		✓					
CKM_SHA3_512_KEY_DERIVATION [*]							✓
CKM_RIPEMD160				✓			
CKM_RIPEMD160_HMAC_GENERAL		✓					
CKM_RIPEMD160_HMAC		✓					
CKM_XOR_BASE_AND_DATA							✓
CKM_CONCATENATE_BASE_AND_KEY							✓
CKM_CONCATENATE_BASE_AND_DATA							✓
CKM_CONCATENATE_DATA_AND_BASE							✓
CKM_EXTRACT_KEY_FROM_KEY							✓
CKM_UTC_AES_KEY_WRAP	✓ ²					✓	
CKM_UTC_AES_KEY_WRAP_PAD	✓ ²					✓	
CKM_UTC_AES_KEY_WRAP_KWP [*]	✓ ²					✓	

Table 25: List of supported mechanisms defined in the PKCS#11 standard

1	SR = SignRecover, VR = VerifyRecover
2	Single-part operations only
3	Single-part sign operations only
4	Wrap only
*	As specified in PKCS #11 Cryptographic Token Interface Current Mechanisms Specification 3.00 Draft Version

Public Object Support

Currently only CKK_RSA and CKK_EC public objects (CKA_PRIVATE == CK_FALSE) are supported for the following operations:

```
C_GetAttributeValue
C_EncryptInit
C_Encrypt
C_EncryptUpdate
C_DecryptInit
C_Decrypt
C_DecryptUpdate
C_SignInit
C_Sign
C_SignUpdate
C_VerifyInit
C_Verify
C_VerifyUpdate
C_WrapKey
C_UnwrapKey
```

3.9.4 JCE API

The JCE API supports the following algorithms:

Algorithm	Padding Modes
SHA1withRSA	PKCS1, PSS
SHA224withRSA	PKCS1, PSS
SHA256withRSA	PKCS1, PSS
SHA384withRSA	PKCS1, PSS
SHA512withRSA	PKCS1, PSS
SHA3-224withRSA	PKCS1, PSS
SHA3-256withRSA	PKCS1, PSS
SHA3-384withRSA	PKCS1, PSS
SHA3-512withRSA	PKCS1, PSS
MD5withRSA	PKCS1, PSS
SHA1withECDSA	-
SHA224withECDSA	-
SHA256withECDSA	-
SHA384withECDSA	-
SHA512withECDSA	-
SHA3-224withECDSA	-
SHA3-256withECDSA	-

Algorithm	Padding Modes
SHA3-384withECDSA	-
SHA3-512withECDSA	-
MD5withECDSA	-
SHA1withDSA	-

Table 26: Signatures algorithms supported by the JCE API

Algorithm	Padding Modes
SHA1withRSA	PKCS1, PSS
SHA224withRSA	PKCS1, PSS
SHA256withRSA	PKCS1, PSS
SHA384withRSA	PKCS1, PSS
SHA512withRSA	PKCS1, PSS
SHA3-224withRSA	PKCS1, PSS
SHA3-256withRSA	PKCS1, PSS
SHA3-384withRSA	PKCS1, PSS
SHA3-512withRSA	PKCS1, PSS
SHA1withECDSA	-
SHA224withECDSA	-
SHA256withECDSA	-
SHA384withECDSA	-
SHA512withECDSA	-
SHA3-224withECDSA	-
SHA3-256withECDSA	-
SHA3-384withECDSA	-
SHA3-512withECDSA	-

Table 27: Signatures algorithms supported by the JCE API

KeyPairGenerator

Algorithm
RSA
EC
DSA

Table 28: KeyFactory algorithms supported by the JCE API

Algorithm
RSA
EC

Table 29: KeyFactory algorithms supported by the JCE API

KeyFactory

Algorithm
RSA
EC
DSA

Table 30: KeyFactory algorithms supported by the JCE API

Algorithm
RSA
EC

Table 31: KeyFactory algorithms supported by the JCE API

SecretKeyFactory

Algorithm
DES
DESede
AES

Table 32: SecretKeyFactory algorithms supported by the JCE API

Algorithm
AES

Table 33: SecretKeyFactory algorithms supported by the JCE API

KeyStore

Algorithm
CryptoServer

Table 34: KeyStore algorithms supported by the JCE API

SecureRandom

Algorithm
SHA1PRNG
CryptoServer

Table 35: SecureRandom algorithms supported by the JCE API

KeyGenerator

Algorithm
DES
DESede
AES

Table 36: KeyGenerator algorithms supported by the JCE API

Algorithm
AES

Table 37: KeyGenerator algorithms supported by the JCE API

Cipher

Algorithm	Block mode	Padding mode
DES	ECB, CBC	NONE, PKCS5, ISO10126
DESede	ECB, CBC	NONE, PKCS5, ISO10126
AES	ECB, CBC, OFB	NONE, PKCS5, ISO10126
AES	GCM, CCM	NONE
RSA	-	NONE, PKCS1, OAEP

Table 38: Cipher algorithms supported by the JCE API

Algorithm	Block mode	Padding mode
AES	ECB, CBC, OFB	NONE, PKCS5, ISO10126
AES	GCM, CCM	NONE
RSA	-	NONE, PKCS1, OAEP

Table 39: Cipher algorithms supported by the JCE API

MAC

Algorithm	Block mode	Padding mode
DES	CBC	NONE,
DESwithPKCS5Padding	CBC	PKCS5
DESwithISO10126Padding	CBC	ISO010126
AES	CBC	NONE
AESwithPKCS5Padding	CBC	PKCS5
AESwithISO10126Padding	CBC	ISO010126

Table 40: MAC algorithms supported by the JCE API

Algorithm	Mode
DES	HmacMD5
DES	HmacSHA1
DES	HmacSHA224
DES	HmacSHA256
DES	HmacSHA384
DES	HmacSHA512
DES	HmacSHA3-224
DES	HmacSHA3-256
DES	HmacSHA3-384
DES	HmacSHA3-512
DES	HmacRMD160
AES	HmacMD5
AES	HmacSHA1
AES	HmacSHA224
AES	HmacSHA256
AES	HmacSHA384
AES	HmacSHA512
AES	HmacSHA3-224
AES	HmacSHA3-256
AES	HmacSHA3-384
AES	HmacSHA3-512
AES	HmacRMD160

Table 41: MAC and hash algorithms supported by the JCE API

Algorithm	Block mode	Padding mode
AES	CBC	NONE

Algorithm	Block mode	Padding mode
AESwithPKCS5Padding	CBC	PKCS5
AESwithISO10126Padding	CBC	ISO010126

Table 42: MAC algorithms supported by the JCE API

Algorithm	Mode
AES	HmacMD5
AES	HmacSHA1
AES	HmacSHA224
AES	HmacSHA256
AES	HmacSHA384
AES	HmacSHA512
AES	HmacSHA3-224
AES	HmacSHA3-256
AES	HmacSHA3-384
AES	HmacSHA3-512
AES	HmacRMD160

Table 43: MAC and hash algorithms supported by the JCE API

MessageDigest

Mode
MD5
SHA-1
SHA-224
SHA-256
SHA-384
SHA-512
SHA3-224
SHA3-256
SHA3-384
SHA3-512
RMD-160

Table 44: Message digest modes supported by the JCE API


<i>Mode</i>
SHA-1
SHA-224
SHA-256
SHA-384
SHA-512
SHA3-224
SHA3-256
SHA3-384
SHA3-512




Table 45: Message digest modes supported by the JCE API

3.10 Built-in Elliptic Curves

The device offers a collection of elliptic curves, which can be used for ECDSA signature creation, EdDSA signature creation and ECDH key agreement. Each curve is specified by elliptic curve domain parameters and is identified by a name.

The following table lists all built-in Elliptic Curves by their name, denotes the bit size of their domain parameters (i.e. the key size) and references the specification where the respective curve and its domain parameters are defined.

Elliptic Curve calculations are implemented in firmware on CryptoServer CSe-Series models CSe10 and CSe100, and on Se-Series models Se12 and Se52. On CryptoServer Se-Series models Se500 and Se1500 numerous common Elliptic Curves are implemented in hardware while other less common Elliptic Curves are implemented in firmware. Elliptic Curves that are implemented in hardware on CryptoServer Se-Series models Se500 and Se1500 are marked by a  in the last column of the table below.

<i>Name(s)</i>	<i>Size</i>	<i>Defined in</i>	<i>HW implementation on Se500 / Se1500</i>
secp112r1	112	[SEC2]	
secp112r2	112	[SEC2]	
sect113r1	113	[SEC2]	
sect113r2	113	[SEC2]	
secp128r1	128	[SEC2]	
secp128r2	128	[SEC2]	
sect131r1	131	[SEC2]	
sect131r2	131	[SEC2]	
brainpoolP160r1	160	[BP]	

Name(s)	Size	Defined in	HW implementation on Se500 / Se1500
brainpoolP160t1	160	[BP]	✓
secp160k1	160	[SEC2]	
secp160r1	160	[SEC2]	
secp160r2	160	[SEC2]	
NIST-K163 / sect163k1	163	[FIPS186-4], [ANSI-X9.62], [SEC2]	
sect163r1	163	[SEC2]	
NIST-B163 / sect163r2	163	[FIPS186-4], [ANSI-X9.62], [SEC2]	
brainpoolP192r1	192	[BP]	✓
brainpoolP192t1	192	[BP]	✓
NIST-P192 / secp192r1	192	[FIPS186-4], [ANSI-X9.62], [SEC2]	✓
secp192k1	192	[SEC2]	
sect193r1	193	[SEC2]	
sect193r2	193	[SEC2]	
brainpoolP224r1	224	[BP]	✓
brainpoolP224t1	224	[BP]	✓
NIST-P224 / secp224r1	224	[FIPS186-4], [ANSI-X9.62], [SEC2]	✓
secp224k1	224	[SEC2]	
NIST-K233 / sect233k1	233	[FIPS186-4], [ANSI-X9.62], [SEC2]	
NIST-B233 / sect223r1	233	[FIPS186-4], [ANSI-X9.62], [SEC2]	
sect239k1	239	[SEC2]	
curve25519	255	[RFC 7748] with Errata ID 4730	
edwards25519	255	[RFC 7748] with Errata ID 4730	
brainpoolP256r1	256	[BP]	✓
brainpoolP256t1	256	[BP]	✓
NIST-P256 / secp256r1	256	[FIPS186-4], [ANSI-X9.62], [SEC2]	✓
secp256k1	256	[SEC2]	
FRP256v1	256	[ANSSI]	
sm2p256v1	256	[SM2]	No

Name(s)	Size	Defined in	HW implementation on Se500 / Se1500
NIST-K283 / sect283k1	283	[FIPS186-4], [ANSI-X9.62], [SEC2]	
NIST-B283 / sect283r1	283	[FIPS186-4], [ANSI-X9.62], [SEC2]	
brainpoolP320r1	320	[BP]	✓
brainpoolP320t1	320	[BP]	✓
brainpoolP384r1	384	[BP]	✓
brainpoolP384t1	384	[BP]	✓
NIST-P384 / secp384r1	384	[FIPS186-4], [ANSI-X9.62], [SEC2]	✓
NIST-K409 / sect409k1	409	[FIPS186-4], [ANSI-X9.62], [SEC2]	
NIST-B409 / sect409r1	409	[FIPS186-4], [ANSI-X9.62], [SEC2]	
curve448	448	[RFC 7748] with Errata ID 4730	
edwards448	448	[RFC 7748] with Errata ID 4730	
brainpoolP512r1	512	[BP]	✓
brainpoolP512t1	512	[BP]	✓
NIST-P521 / secp521r1	521	[FIPS186-4], [ANSI-X9.62], [SEC2]	✓
NIST-K571 / sect571k1	571	[FIPS186-4], [ANSI-X9.62], [SEC2]	
NIST-B571 / sect571r1	571	[FIPS186-4], [ANSI-X9.62], [SEC2]	

Table 46: Built-in Elliptic curves

4 Setup

You can install the software you need to run the CryptoServer on a host computer with Windows or Unix/Linux. This includes:

- The CryptoServer administration tools, csadm and CAT. These tools can be used to administer both the CryptoServer PCIe and the CryptoServer LAN.
- The CryptoServer Crypto APIs (JCE, CXI, CSP/CNG, PKCS#11)
- The CryptoServer Simulator
- The complete documentation of the CryptoServer and the CryptoServer LAN

Before installing the software, you need to make some preparations.

4.1 System Requirements

- CPU: Intel x86/x64, AMD x86/x86-64
- Hard disk capacity: at least 120 Mbyte
- RAM: more than 12 Mbyte
- A free PCIe expansion slot, if you are using a CryptoServer CSe-Series or Se-Series Gen2
 - The computer in which you want to mount the CryptoServer PCIe card must be sited in a cool, well-ventilated place.
 - Do not place it near sources of heat or in direct sunlight.
 - You must ensure that the expansion slot in which CryptoServer is mounted in the ventilation airflow of the computer.
 - The CryptoServer PCIe card should only be inserted below other plug-in cards that produce heat.
 - Ensure that you keep one expansion slot free between all other heat-producing plug-in cards, or other CryptoServer PCIe cards.
- Network card: Ethernet 10/100/1000 Mbit/s for the connection to the CryptoServer LAN
- Operating systems: You find the current and complete list of supported operating systems in the CS_PD_SecurityServer_SupportedPlatforms.pdf document on the

SecurityServer/CryptoServer SDK product CD in the ...\\Documentation\\Product Details directory.

- CD-ROM disc drive
- Current Java versions for Windows/Linux

The following table lists the operating systems supported by SecurityServer 6.0.0.

Operating System	CryptoServer	u.trust Anchor
Windows x64		
Windows 10	✓	✓
Windows 11 (Pro)	✓	✓
Windows Server 2016	✓	✓
Windows Server 2019	✓	✓
Windows Server 2022 (Standard)	✓	✓
Linux x64		
Red Hat Enterprise Linux 8	✓	✓
Red Hat Enterprise Linux 9	✓	✓
SUSE Linux Enterprise Server 12	✓	✓
SUSE Linux Enterprise Server 15	✓	✓
Ubuntu 20.04 LTS	✓	✓
Ubuntu 22.04 LTS	✓	✓



32-bit versions of cryptographic providers (e.g. PKCS#11 provider, CNG provider) are still shipped with SecurityServer 6.0.0. In addition, 32-bit versions of selected commandline tools (e.g. csadm, cxtool) for Linux are still shipped with SecurityServer 6.0.0. Please notice that support for 32 bit has been declared as legacy, and shipment of such 32-bit cryptographic provider and command line tools will be discontinued in future.

4.2 Preparations

If you are using a CryptoServer LAN, you must first you must first integrate it into your network. In addition, you must have assigned an IP address to this device. You can find detailed instructions on how to bring your CryptoServer LAN into service in the *CryptoServer LAN Operating Manual* provided on the SecurityServer product CD.

If you are using a CryptoServer PCIe card, you must have it mounted and installed the driver for the card.

The following manuals describe how to install the driver for the PCIe card:

- CryptoServer PCIe - Operating Manual – CSe-Series
- CryptoServer PCIe - Operating Manual – Se-Series Gen2

4.3 Installation on Windows

The CryptoServer host software and the CryptoServer Simulator support the operating systems listed in the [CS_PD_SecurityServer_SupportedPlatforms.pdf](#) document provided on the SecurityServer/CryptoServer SDK product CD in the [... \Documentation\Product Details](#) directory.

On a 64-bit computer, only a 64-bit Java SE Runtime Environment is supported by the CryptoServer, and on a 32-bit computer, only a 32-bit Java SE Runtime Environment is supported.



If you use a 64-bit computer, perform the following command in a command-line to verify whether a 64-bit version of the Java SE Runtime environment has been installed on your computer:

```
java -d64 -version
```

Output example:

```
java version "1.7.0_51"  
Java(TM) SE Runtime Environment (build 1.7.0_51-b13)  
Java HotSpot(TM) 64-Bit Server VM (build 24.51-b03, mixed mode)
```

If you use a 32-bit computer, verify whether a 32-bit Java version has been installed:

```
java -d32 -version
```



Before you start the installation, check that you have installed the current version of the Java SE Runtime Environment on your computer.

We recommend you uninstall the old version of the Java SE Runtime Environment from your computer first. Restart your computer before you install the current version of the Java SE Runtime Environment.

4.3.1 Installing the Host Software and CryptoServer Simulator

Prerequisites

To run the GUI tools, your Java installation will need to support unlimited crypto. If your system does not have a JRE, download one from <http://openjdk.java.net/Next> install the corresponding Java security policy files, for example "UnlimitedJCEPolicyJDK11.zip" from the openjdk.java.net¹ website. Extract them and copy the *.jar files to your /lib/security directory.

This section describes the installation using the GUI. If you want to perform a silent installation, see [Installing the Host Software and CryptoServer Simulator Without User Interaction](#) (p. 131).

1. In the highest folder level of the product bundle, click the file `SecurityServer-<version number>.msi`. For versions earlier than 4.40.0, the file is called `CryptoServerSetup-<version number>.exe`.



If necessary, you will be prompted to install the Microsoft runtime environment (VCRedist). Click OK to confirm the corresponding dialog box.

2. In the installation wizard click the **Next**.
3. In the **Select Destination Location** dialog use the **Browse...** button to select a different directory for installing the software or confirm the default installation directory.
4. Click **Next >**.

The **Installation type** dialog box opens (only for CryptoServer 4.40.0 and later).

- *Default installation*

The CryptoServer administration tools (CAT, csadm etc.), the CryptoServer

¹ <http://openjdk.java.net>

documentations and the CryptoServer Simulator are installed.

- *User-defined installation*

This installation type lets you specify the components to be installed.

- *Complete installation*

This corresponds to selecting all items of the user-defined installation except for PCIe driver and PIN Pad driver.

Select the installation type of your choice.

5. The Select Components dialog box opens. For CryptoServer 4.40.0 and later, only in case of the user-defined installation.
6. Select the components you want to install. By default, **Simulator** (only for CryptoServer 4.40.0 and later), **Administration** and **Documentation** are selected.



To install the CryptoServer Simulator for Windows, select the corresponding check box here.

7. Click **Next**.
The Select Start Menu Folder dialog box opens. Here you can select the directory in which the shortcut used to start the program is to be stored.
8. Confirm the default directory or click the **Browse** button to change the directory for the shortcut.



If you click Don't create a Start Menu folder, no shortcut is created in the Windows Start menu.

9. Click **Next**.
In the **Select Additional Tasks** dialog box you can decide whether to create a desktop icon or not. By default, **Create a desktop icon** is selected.
10. Click **Next**.
In the **Ready to Install** dialog box opens you can see what you have selected or specified in the previous dialog boxes.
11. Click **Install**.



During installation a special setup window opens in which you are prompted to download the Java(TM) Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files from the Oracle web site. These files are required to work with the Utimaco CryptoServer JCE Provider.

12. To use the JCE interface, click **Yes** in the **Setup** window.

After completing the installation, the **Completing the CryptoServer Setup Wizard** dialog box opens. Here you can specify whether the CAT should be launched automatically after successful software installation. By default, the **Launch CryptoServer Administration** option is selected.

13. Click the **Finish** to complete the software installation.

You have now finished installing the CryptoServer software. CAT starts now by default and the Settings dialog box is displayed.

If CAT starts with the `invalid auth. key` error message, either delete the `CS_AUTH_KEYS` environment variable or update it by performing the `csadm GetHSMAuthKey` command, see *GetHSMAuthKey* in the *CryptoServer – csadm Manual* for details.

4.3.2 Installing the Host Software and CryptoServer Simulator Without User Interaction



This section applies to 64-bit Windows operating systems only.



This section applies to CryptoServer 4.40.0 and later.

The installation is done by performing a command with the following syntax:

```
msiexec /i SecurityServer-<version>.msi <parameters> /qn
```

`SecurityServer-<version>.msi` is the installation file you find in the topmost directory of the product CD. `<parameters>` are optional parameters specifying the optional components to be installed.

Parameter	Description
	<p>If no parameter is used, a default installation is performed. The administration tools (csadm, CAT etc.), the documentations and the CryptoServer Simulator are installed.</p> <p>This corresponds to the parameter <code>INSTALLLEVEL= 3</code>.</p> <p>This corresponds to the preselected items Administration (CryptoServer Administration Tools), Documentation (CryptoServer Documentation) and Simulator (CryptoServer Simulator) in the GUI of the user-interactive installation, see the step to select components in Installing the Host Software and CryptoServer Simulator (p. 131).</p>
ADMINISTRATION_ONLY	<p>This corresponds to selecting only Administration (CryptoServer Administration Tools) and Documentation (CryptoServer Documentation) in the GUI of the user-interactive installation, see the step to select components in Installing the Host Software and CryptoServer Simulator (p. 131).</p>
DOCUMENTATION_ONLY	<p>This corresponds to selecting only Documentation (CryptoServer Documentation) in the GUI of the user-interactive installation, see the step to select components in Installing the Host Software and CryptoServer Simulator (p. 131).</p>
PKCS11=1	<p>This corresponds to selecting PKCS#11 (PKCS#11 R3 – Cryptographic Token Interface) in the GUI of the user-interactive installation, see the step to select components in Installing the Host Software and CryptoServer Simulator (p. 131).</p>
PKCS11_32=1	<p>This corresponds to selecting PKCS#11 > PKCS#11_32bit (PKCS#11 R3 – Cryptographic Token Interface 32 bit) in the GUI of the user-interactive installation, see the step to select components in Installing the Host Software and CryptoServer Simulator (p. 131).</p> <p>This includes PKCS11=1.</p>
CXI=1	<p>This corresponds to selecting CXI > CXI_C (CXI – Cryptographic eXtended Interface) in the GUI of the user-interactive installation, see the step to select components in Installing the Host Software and CryptoServer Simulator (p. 131).</p>
CXI_32=1	<p>This corresponds to selecting CXI > CXI_C > CXI_C_32bit (CXI– Cryptographic eXtended Interface 32 bit) in the GUI of the user-interactive installation, see the step to select components in Installing the Host Software and CryptoServer Simulator (p. 131).</p> <p>This includes CXI_C.</p>
CXI_JAVA=1	<p>This corresponds to selecting CXI > CXI_Java (CXI– Cryptographic eXtended Interface Java) in the GUI of the user-interactive installation, see the step to select components in Installing the Host Software and CryptoServer Simulator (p. 131).</p>

Parameter	Description
CSPCNG=1	This corresponds to selecting CSP/CNG (CSP/CNG – Cryptographic Service Provider (Microsoft)) in the GUI of the user-interactive installation, see the step to select components in Installing the Host Software and CryptoServer Simulator (p. 131) .
CSPCNGSRV=1	This corresponds to selecting CSP/CNG > CSP Login Service (CSP/CNG Interactive Log-in Support) in the GUI of the user-interactive installation, see the step to select components in Installing the Host Software and CryptoServer Simulator (p. 131) . This includes CSP/CNG.
CSPCNG_32=1	This corresponds to selecting CSP/CNG > CSP/CNG_32bit (CSP/CNG – Cryptographic Service Provider (Microsoft) – 32 bit) in the GUI of the user-interactive installation, see the step to select components in Installing the Host Software and CryptoServer Simulator (p. 131) . This includes CSP/CNG.
JCE=1	This corresponds to selecting JCE (JCE Provider) in the GUI of the user-interactive installation, see the step to select components in Installing the Host Software and CryptoServer Simulator (p. 131) .
JCE2=1	This corresponds to selecting JCE2 (JCE2 Provider) in the GUI of the user-interactive installation, see the step to select components in Installing the Host Software and CryptoServer Simulator (p. 131) .
EKM=1	This corresponds to selecting EKM (EKM – Extensible Key Management) in the GUI of the user-interactive installation, see the step to select components in Installing the Host Software and CryptoServer Simulator (p. 131) .
OPENSSL=1	This corresponds to selecting OpenSSL (OpenSSL PKCS11 engine) in the GUI of the user-interactive installation, see the step to select components in Installing the Host Software and CryptoServer Simulator (p. 131) .
PCIEDRV=1	This corresponds to selecting Drivers/CryptoServer (CryptoServer PCIe driver) in the GUI of the user-interactive installation, see the step to select components in Installing the Host Software and CryptoServer Simulator (p. 131) . This corresponds as well to performing the instructions in <i>Installation of the CryptoServer Driver Software > Installation on Windows Operating Systems > Installing the Driver</i> , in the <i>CryptoServer PCIe CSe-Series Operating Manual</i> and the <i>CryptoServer PCIe Se-Series Gen2 Operating Manual</i> . This includes the PCIe driver and the CryptoServer driver.
PPDRV=1	Installation of the drivers for the PIN pads „REINER SCT cyberJack“ and „Utimaco cyberJack one“. This corresponds to selecting Drivers/cyberJack in the GUI of the user-interactive installation, see the step to select components in Installing the Host Software and CryptoServer Simulator (p. 131) . This corresponds as well to performing the instructions in 3.3.6, "Installing the PIN Pad Driver".

Parameter	Description
PPD=1	This corresponds to selecting PPD (PPD – PIN Pad Daemon) in the GUI of the user-interactive installation, see the step to select components in Installing the Host Software and CryptoServer Simulator (p. 131) . For details about the PIN pad daemon, see <i>Using a Local PIN Pad for a Remote CryptoServer</i> in the <i>CryptoServer – csadm Manual</i> .
INSTALLLEVEL=3	This corresponds to selecting Administration (CryptoServer Administration Tools), Documentation (CryptoServer Documentation) and Simulator (CryptoServer Simulator) in the GUI of the user-interactive installation, see the step to select components in Installing the Host Software and CryptoServer Simulator (p. 131) . These items are selected by default.
INSTALLLEVEL=1000	This corresponds to selecting all items except for PCIe Driver and PIN Pad driver in the GUI of the user-interactive installation, see the step to select components in Installing the Host Software and CryptoServer Simulator (p. 131) . This corresponds to a complete installation. This corresponds as well to using the following collection of parameters: PKCS11=1 PKCS11_32=1 CXI=1 CXI_32=1 CXI_JAVA=1 CSPCNG=1 CSPCNGSRV=1 CSPCNG_32=1 JCE=1 JCE2=1 EKM=1 OPENSSL=1 PPD=1
INSTALLLEVEL=1001	This corresponds to selecting all items including PCIe Driver and PIN Pad driver in the GUI of the user-interactive installation, see the step to select components in Installing the Host Software and CryptoServer Simulator (p. 131) . This corresponds as well to using the following collection of parameters: PKCS11=1 PKCS11_32=1 CXI=1 CXI_32=1 CXI_JAVA=1 CSPCNG=1 CSPCNGSRV=1 CSPCNG_32=1 JCE=1 JCE2=1 EKM=1 OPENSSL=1 PCIEDRV=1 PPD=1 PPDRV=1
APPDIR	Absolute installation path of the host software and the CryptoServer Simulator. Default : C:\Program Files\Utimaco\SecurityServer If the installation path contains at least one space, quotation marks at the beginning and the end of the installation path are mandatory. Example: APPDIR="C:\My SecurityServer path" Using APPDIR corresponds to selecting the installation path in the GUI of the user-interactive installation, see Installing the Host Software and CryptoServer Simulator (p. 131) .

Table 47: Parameters for an installation without user-interaction

All listed parameters are optional. Any combinations of the parameters are supported. Omit the parameters that should not be used. Additional msixec parameters like /l for logging can be used as well.

Examples:

- Default installation (administration tools, documentations and the CryptoServer Simulator)

```
msiexec /i SecurityServer-<version>.msi /qn
```

or

```
msiexec /i SecurityServer-<version>.msi INSTALLLEVEL=3 /qn
```

- Default installation plus the PKCS#11 R3 cryptographic token interface

```
msiexec /i SecurityServer-<version>.msi PKCS11=1 /qn
```

- Complete installation

```
msiexec /i SecurityServer-<version>.msi INSTALLLEVEL=1000 /qn
```

- Complete installation plus the installation of the PCIe driver

```
msiexec /i SecurityServer-<version>.msi INSTALLLEVEL=1000 PCIEDRV=1 /qn
```

- Complete installation plus the installation of the PIN pad driver

```
msiexec /i SecurityServer-<version>.msi INSTALLLEVEL=1000 PPDRV=1 /qn
```

- Complete installation plus the installation of the PCIe driver and the PIN pad driver

```
msiexec /i SecurityServer-<version>.msi INSTALLLEVEL=1001 /qn
```

4.3.3 Setting up Java Cryptography Extension (JCE) for Windows

If you want to use the JCE interface you need the Java(TM) Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files.

Download the appropriate Java(TM) Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files for the Java version installed on your computer.



Before downloading the Java(TM) Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files, check which Java version is installed on your computer by typing `java -version` in a command line.

1. Load the appropriate ZIP file `jce_policy-x.zip` onto your computer.
2. Extract `jce_policy-x.zip` on your computer.
From the `jce_policy-x.zip` file you require the following files:
 - `local_policy.jar`
 - `US_export_policy.jar`
3. Copy them to the following directory:
`C:\Program Files\Java\jre\lib\security`



The installation of the Java(TM) Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files is completed



In the `C:\Program Files\Utimaco\SecurityServer\Software\JCE\CryptoServer.cfg` file, the `ConnectionTimeout` parameter specifies the maximum time in milliseconds to wait before the connection establishment is aborted if the device is not responding.

In practice, the timeout can reach approximately $2 \cdot n \cdot T$.

Legend:

- n: Number of HSMs specified by the `Device` parameter
- T: The timeout value specified by the `ConnectionTimeout` parameter

In the same file, the `Timeout` parameter specifies the maximum time in milliseconds to wait for the answer from CryptoServer after sending a command.

In practice, the timeout can reach approximately $2 \cdot n \cdot T$.

Legend:

- n: Number of HSMs specified by the `Device` parameter
- T: The timeout value specified by the `Timeout` parameter

4.3.4 Installing the PIN Pad Driver

To be able to use the delivered PIN pad for the CryptoServer administration the appropriate driver must be manually installed on the host computer where the host software has already been installed.



When performing the `csadm LogonSign` command, the following error message might be shown:

```
Error B91D0003
  PIN pad API
  no device found
```

In this case, install the PIN pad driver.



When administering the CryptoServer (for example, by performing the `csadm LogonSign` command), the following error message might be shown:

```
Error B9000405
  CryptoServer API
  authentication layer for CryptoServer
  error in signature function
```

In this case, update the PIN pad driver.

4.3.4.1 Prerequisites:

- You have uninstalled any previous PIN pad driver versions on the host computer



In case the PIN pad REINER SCT cyberJack (as shown in Figure 1, section “[PIN Pads \(p. 47\)](#)”) was delivered to you and there is a version of Utimaco's PIN pad driver already installed on

the host computer and listed in the Windows Device Manager as LibUSB Devices, then no driver update or new driver installation is required.

- You have installed the product CD or you have inserted it into the computer's CD-ROM drive.



Under no circumstances install the "Base components" USB driver for Windows from the company REINER SCT.

If your Windows operating system already has REINER SCT's "Base Components" driver installed, it is essential that you remove it before you install the Utimaco IS GmbH driver. The official REINER SCT "Base Components" driver and the driver from Utimaco IS GmbH cannot be used in parallel.



Under no circumstances run the "cyberJack Device Manager" from the REINER SCT Base components, and perform a firmware update of the PIN pad driver.

Installing a firmware update on the PIN pad renders this PIN pad unusable with CryptoServer administration tools and cryptographic libraries. After a firmware update, it is not possible to revert to the PIN pad firmware as shipped by Utimaco IS GmbH.

4.3.4.2 Installing on Windows 10 and later

1. Connect the USB plug of the PIN pad with the host computer.
If you are prompted in a separate dialog box to download the PIN pad driver from the Internet or to let Windows install it automatically, close this dialog box without selecting any of these options.
2. Start the device manager.
 - a. Open the **Run** dialog by pressing the Windows key and then simultaneously pressing the **R** key.

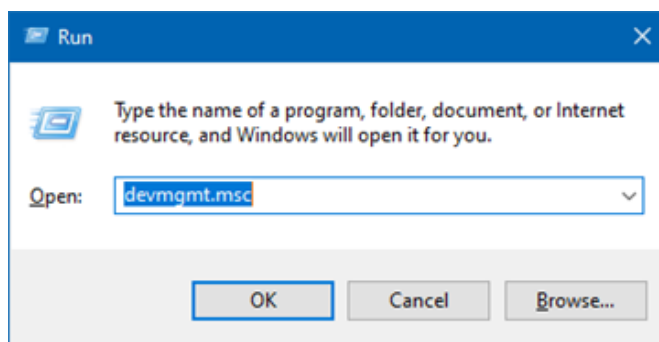


Figure 16 : Run dialog box

- b. Enter `devmgmt.msc` into the text field and press **ENTER**.
The Windows Device Manager opens.

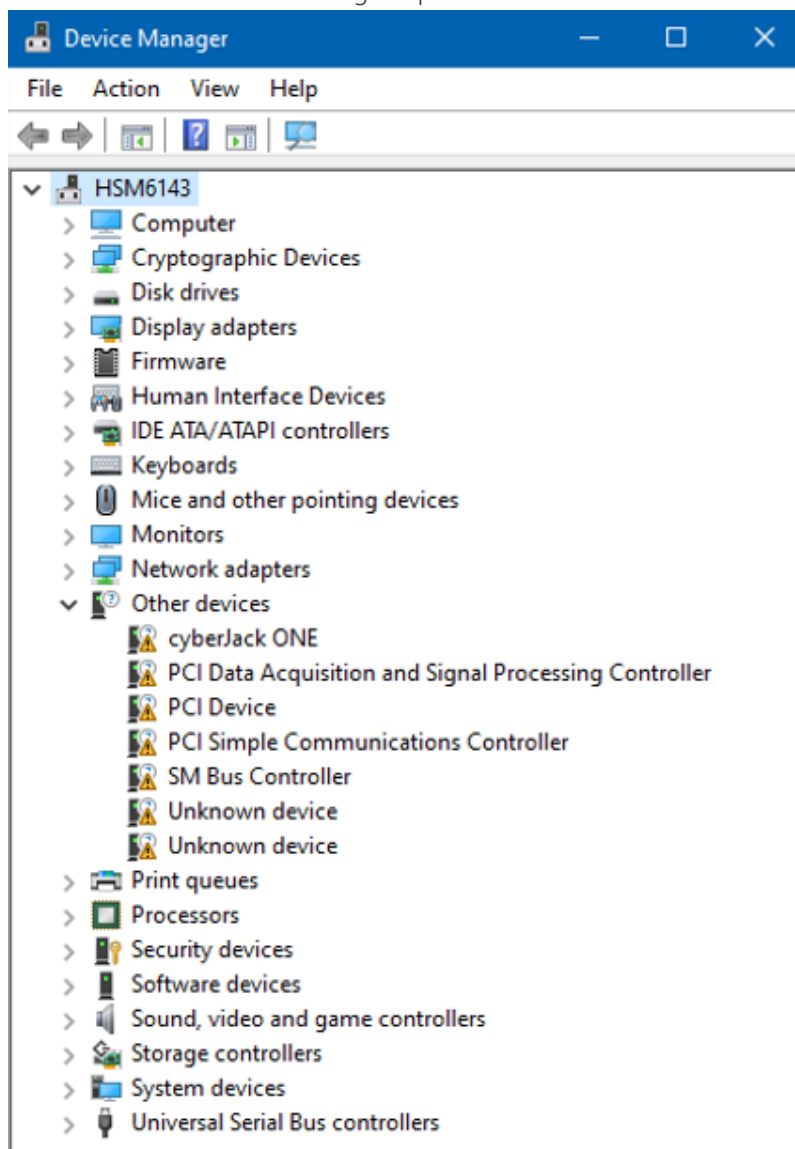


Figure 17 : Windows Device Manager

3. Double click the **Other Devices** entry in the displayed list.
4. Depending on the PIN pad you are using, proceed as follows:
 - a. If the REINER SCT cyberJack PIN pad shown in Figure 1 has been supplied to you, double-click **cyberJack e-com(a)**.
 - b. If the Utimaco cyberJack one PIN pad shown in Figure 2 has been supplied to you, double-click **cyberJack ONE**.

The <PIN pad type> Properties dialog box opens.

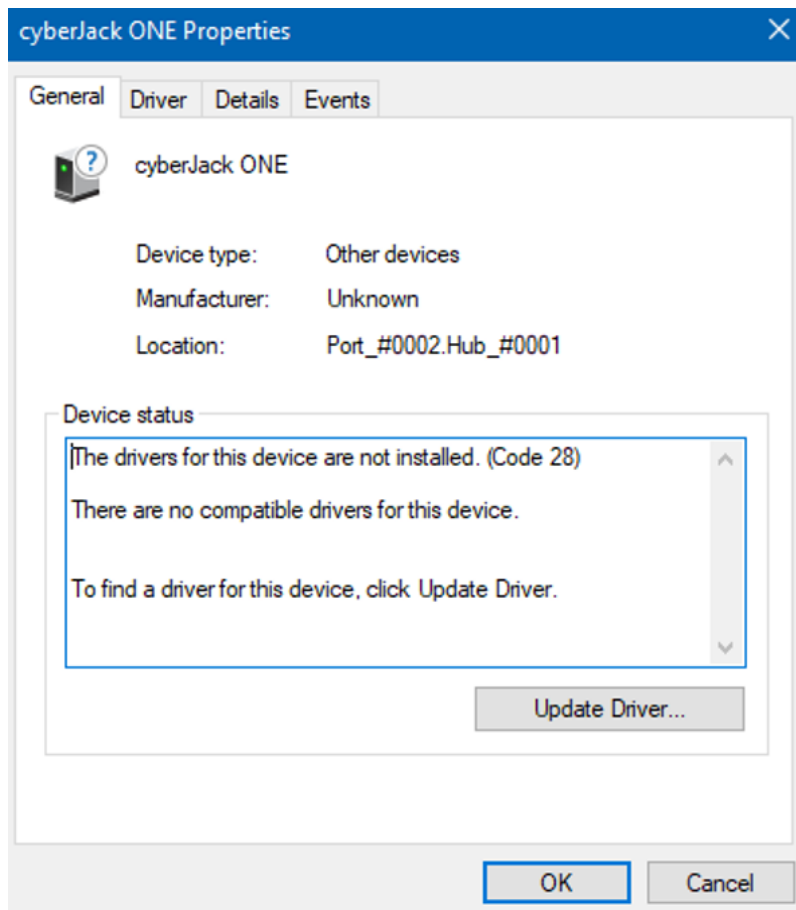


Figure 18 : Dialog box for PIN pad driver installation

5. Click **Update Driver**.
The **Update Driver Software - <PIN pad type>** dialog box opens.

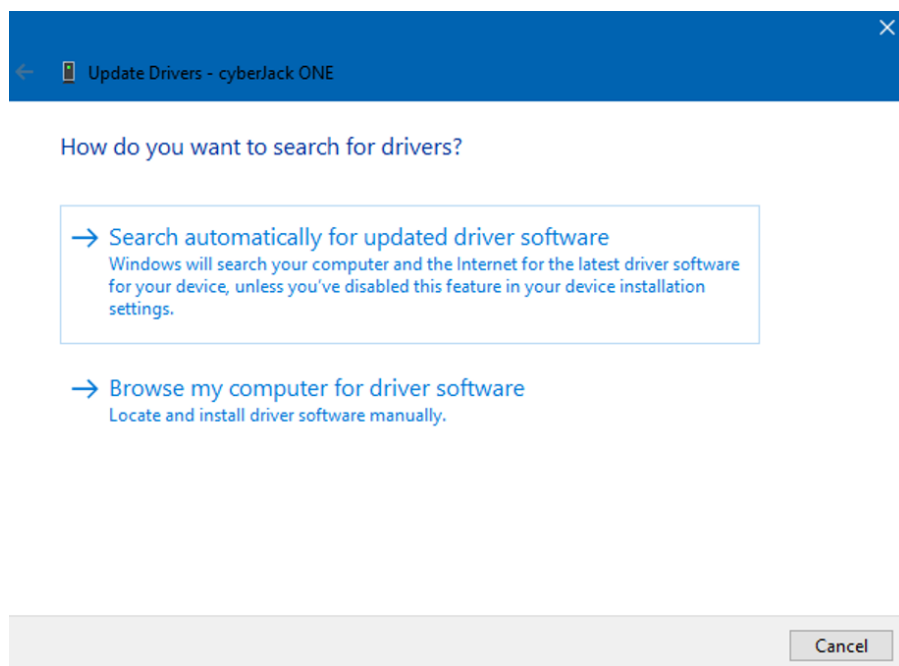


Figure 19 : Dialog box for choosing the PIN pad driver installation

6. Click **Browse my computer for driver software**.

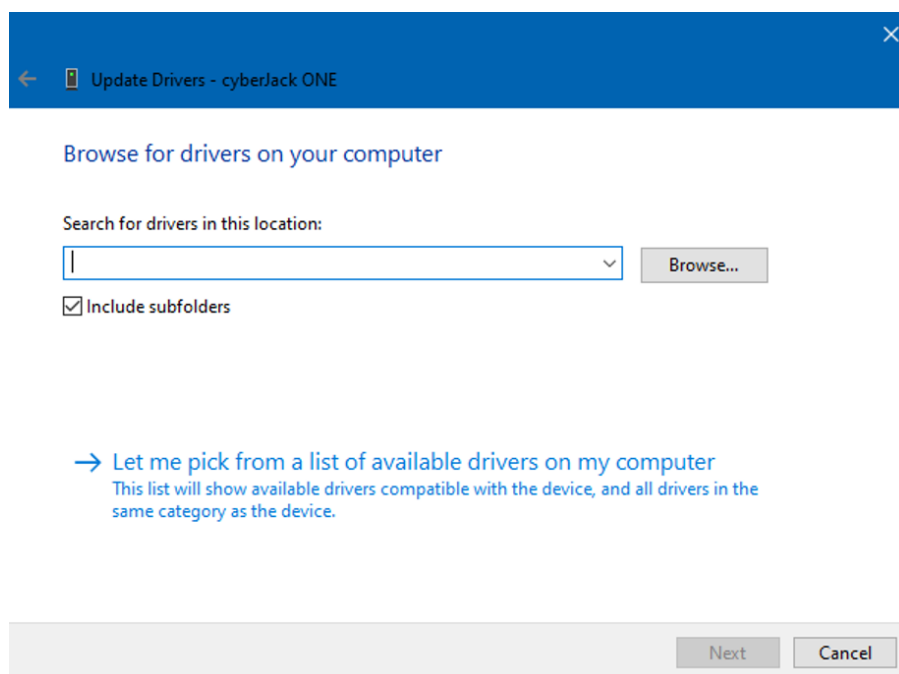


Figure 20 : Dialog box for browsing for drivers

7. Click **Browse** and select the PIN pad driver.
If the installation of the PIN pad driver had been selected in the installer, you find the driver in `C:\Program Files\Utimaco\SecurityServer\cyberJack`.

As an alternative, you find the driver software on the product CD delivered by the Utimaco IS GmbH here: `\Software\Windows\Tools\cyberJack\Driver\`

8. Click **Next**.
9. If a **Windows Security Warning** dialog opens, click **Install**.
The PIN pad driver is now being installed. When completed, the successful driver installation is confirmed in a separate dialog box.
10. Click **Close** in the **<PIN pad type> Properties** dialog box.

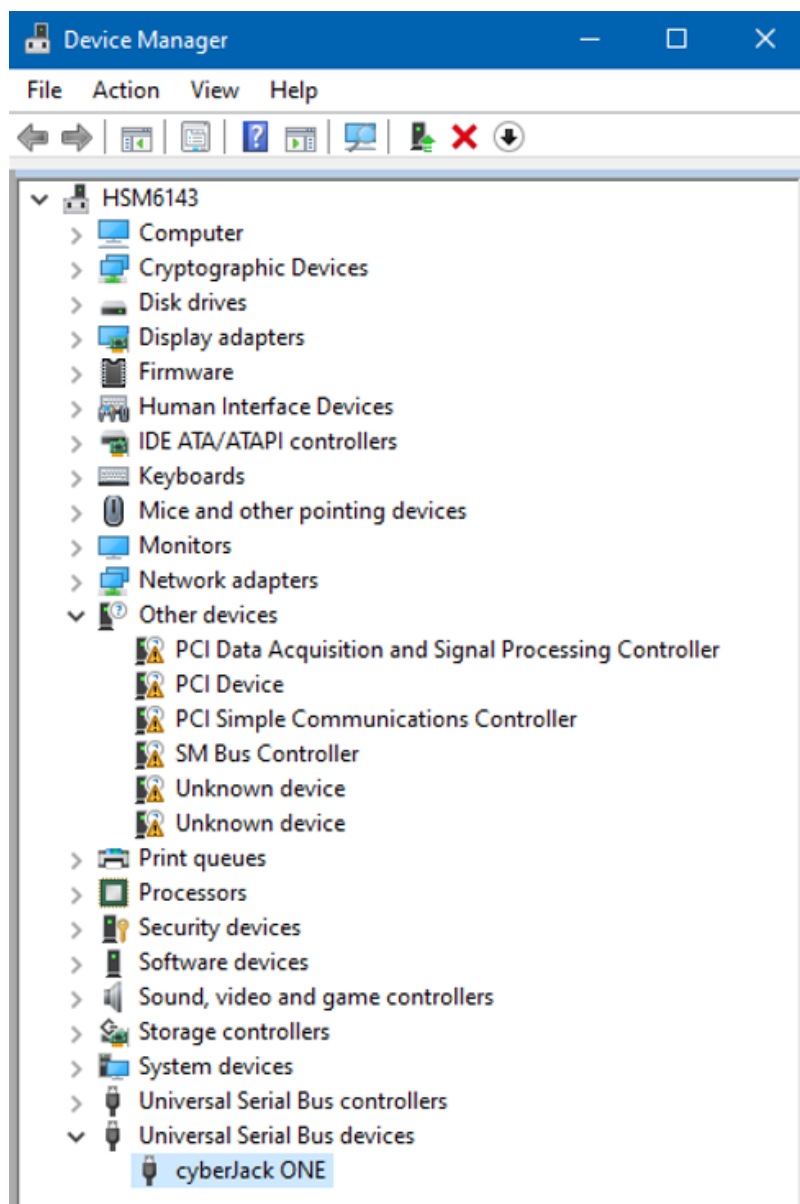


Figure 21 : Windows Device Manager with installed driver



The PIN pad is now displayed with its name – **cyberJack e-com(a)** or **cyberJack ONE** - in the Windows **Device Manager** as a subentry of the **Universal Serial Bus devices** entry.

4.4 Installation on Linux

This chapter describes how to install the CryptoServer host software on a computer that is running a Linux operating system.

4.4.1 Installing csadm

1. If there is no `~/bin` directory present on your system, create it in your user directory:

```
mkdir ~/bin
```

2. Copy the csadm relevant for your operating system (32-bit or 64-bit) into the `~/bin` directory.

An example for Linux 64-bit:

```
cp <mount point of the product CD>/Software/Linux/Administration/csadm ~/bin
```

The `.../Software/Linux/Administration/` directory is to be found in the general SecurityServer bundle. The corresponding `.../Software/Linux/x86-32/Administration` directory for 32 bit is to be found in the 32-bit add-on bundle for the SecurityServer.

3. Ensure that you have write and execute permissions for the CryptoServer administration tool csadm.

```
chmod -R u+w+x ~/bin
```

4. Add the `~/bin` directory to the path in the user configuration file in the shell that is being used.

In our example we have used a bash as the shell, i.e., open the `~/.bashrc` file, and then add the following line to it:

```
export PATH=$PATH:~/bin
```

5. Save the changes and close the `~/.bashrc` file.

4.4.2 Installing CAT



CAT can be used on Linux 64-bit systems:



On a 64-bit computer, only a 64-bit Java SE Runtime Environment is supported by the CryptoServer.

If you use a 64-bit computer, perform the following command in a command line to verify whether a 64-bit version of the Java SE Runtime environment has been installed on your computer:

```
java -d64 -version
```

Output example:

```
java version "1.7.0_51"  
Java(TM) SE Runtime Environment (build 1.7.0_51-b13)  
Java HotSpot(TM) 64-Bit Server VM (build 24.51-b03, mixed mode)
```



To be able to use the CAT, install first the appropriate Sun Java Runtime Environment for the Java version available on your computer.

Perform the following command in a command line to retrieve the installed Java version:

```
java -version
```

Use the package manager for your Unix/Linux system to install the Oracle Java Runtime Environment.

To install the CAT, follow these steps:

1. Copy the `cat.jar` file into your user directory.

```
cp <path to the Product CD>/Software/Linux/Administration/cat.jar ~/
```

2. Start CAT with the following command:

```
java -jar ~/cat.jar
```

4.4.3 Installing the CryptoServer Simulator

You find the list of all currently supported operating systems in the `CS_PD_SecurityServer_Supported_Platforms.pdf` document on the SecurityServer product CD in the `...\Documentation\Product Details` directory.

4.4.3.1 Prerequisites

If you are using a 64-bit Linux operating system, you must have installed `gcc-multilib` on it. To perform the installation of `gcc-multilib`, use the commands described in the next table and consider that you must have root permissions to do so. The root permissions are only needed for this installation but installing other components should be done as a non-privileged user.

Operating system	Installation command
Ubuntu 20.04 LTS and 22.04 LTS	<code>apt install gcc-multilib</code>
SLES 12 and 15	<code>zypper install glibc-32bit</code>
RHEL 8 and 9	<code>yum install glibc.i686 glibc-devel.i686 libstdc++-devel.i686</code> (libgcc.i686 is installed as a dependency.)

To run the GUI tools, your Java installation will need to support unlimited crypto. If your system does not have a JRE, download one from <http://openjdk.java.net/Next> install the corresponding Java security policy files, for example, "UnlimitedJCEPolicyJDK11.zip" from the openjdk.java.net² website. Extract them and copy the `*.jar` files to your `\lib\security` directory.

4.4.3.2 Installation

To install the CryptoServer Simulator:

² <http://openjdk.java.net>

1. Create a new directory in your home directory, for example:

```
mkdir -p ~/Utimaco/CryptoServer/Simulator
```

2. Copy the CryptoServer Simulator files provided on the SecurityServer product CD to the new directory.

```
cp -r /<mount point of SecurityServer product CD>/Software/Linux/Simulator/*  
~/Utimaco/CryptoServer/Simulator
```

3. Define the environment variable CRYPTOSERVER which is needed for the use of the CryptoServer Simulator:

```
export CRYPTOSERVER=3001@localhost
```



Add this command to your `~/.bashrc`, `~/.profile` or `~/.zshrc` according to your shell to call it automatically.

4. Ensure that you have write and execute permissions for the CryptoServer Simulator files.

```
chmod -R u+w+x ~/Utimaco/CryptoServer/Simulator/sim5_linux
```

5. Start the CryptoServer Simulator:

```
~/Utimaco/CryptoServer/Simulator/sim5_linux/bin/cs_sim.sh
```



The `-h` option in this sh file starts a small server enabling multiple connections (see 2.22.9, "Multiple Connections"). The `-o` option starts the SMOS operating system.

This starts the **Utimaco CryptoServer SDK5 – Simulator** command-line application. Do not close this application until you have finished using the simulator.

4.4.4 Setting up Java Cryptography Extension (JCE) for Linux

If you want to use the JCE interface, you need the Java(TM) Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files that can be downloaded from the Oracle website. These files are required for working with the Utimaco's CryptoServer JCE Provider.



Check the Java version installed on your computer by performing the "java -version" command before you start downloading the Java(TM) Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files.

- Load the appropriate ZIP file `jce_policy-x.zip` onto your computer.
 - Extract `jce_policy-x.zip` on your computer.
- From the `jce_policy-x.zip` file you require the following files:

```
- local_policy.jar
- US_export_policy.jar
```

The following example shows the path for Java 6 on a computer with a Linux operating system and 64-bit architecture.

```
/usr/lib/jvm/java-6-openjdk-amd64/jre/lib/security/
US_export_policy.jar
/usr/lib/jvm/java-6-openjdk-amd64/jre/lib/security/local_policy.jar
```

- Copy the policy files shown above to the appropriate directory on your Linux system.



In the `.../Software/JCE/CryptoServer.cfg` file, the `ConnectionTimeout` parameter specifies the maximum time in milliseconds to wait before the connection establishment is aborted if the device is not responding.

In practice, the timeout can reach approximately $2 \cdot n \cdot T$.

Legend:

- n: Number of HSMs specified by the `Device` parameter
- T: The timeout value specified by the `ConnectionTimeout` parameter

In the same file, the `Timeout` parameter specifies the maximum time in milliseconds to wait for the answer from CryptoServer after sending a command.

In practice, the timeout can reach approximately $2 \cdot n \cdot T$.

Legend:

- n: Number of HSMs specified by the `Device` parameter
- T: The timeout value specified by the `Timeout` parameter

4.4.5 Configuring the PIN Pad

There is no PIN pad driver installation required for host computers with a Linux operating system. However, for using an PIN pad delivered by Utimaco on a Linux host computer, the definition of a special udev rule is necessary.

- For using the REINERSCT cyberJack PIN pad:

```
echo 'SUBSYSTEM=="usb", ATTRS{idVendor}=="0c4b", ATTR{idProduct}=="0400",  
MODE="666"' > /lib/udev/rules.d/z80_cyberjack.rules
```

- For using the Utimaco cyberJack One PIN pad:

```
echo 'SUBSYSTEM=="usb", ATTRS{idVendor}=="0c4b", ATTR{idProduct}=="0600",  
MODE="666"' > /lib/udev/rules.d/z80_cyberjack.rules
```

- For using both PIN pads:

```
echo 'SUBSYSTEM=="usb", ATTRS{idVendor}=="0c4b", MODE="666"' > /lib/udev/  
rules.d/z80_cyberjack.rules
```

4.5 Replacing the Default Administrator ADMIN

The default administrator ADMIN is present on the device upon delivery, see [ADMIN, the Default Administrator \(p. 60\)](#).

Since anyone who has the user authentication key on the smartcard or as a keyfile can log in to the CryptoServer as the default administrator ADMIN and manage the device on their own, the user ADMIN should as soon as possible replace their Default Administrator Key `ADMIN.key`. In case the two-person rule is required for the administration tasks, the user ADMIN must be replaced by two (or more) new users with the necessary user permissions.

Replacing the default administrator ADMIN

A new authentication token to be used for replacing the default authentication token can be generated as described in *Generating a User Authentication Key* in the [CryptoServer - csadm Manual \(p. 284\)](#).

Users that assume the role of administrator can use the RSA signature, ECDSA signature or RSA smartcard authentication mechanisms, since only such users are not automatically deleted by an alarm, see [An Alarm and Its Consequences \(p. 38\)](#), or an External Erase, see [External Erase \(p. 44\)](#).

The authentication token replacement process can be executed with either csadm, see *ChangeUser* in [CryptoServer - csadm Manual \(p. 284\)](#), or with CAT, see *Changing a User Authentication Token* in [CryptoServer - CAT Manual \(p. 284\)](#).

Replacing the default administrator ADMIN with two or more users

If the default user ADMIN shall be replaced by two or more users, the sum of the permissions of the newly created users who use a RSA or ECDSA key as authentication token has to be at least 21000000. Only if this permission is reached, the device allows the deletion of the default user ADMIN. The reason for this precaution is that in any situation the device must remain administrable. Whenever an alarm occurs on the device, see [An Alarm and Its Consequences \(p. 38\)](#), or when a Clear is performed, see [Clear \(p. 42\)](#), all users with password authentication mechanism are deleted from the user database. The remaining users should be able to setup the device 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.

This therefore enables a two-person rule for the CryptoServer's administration tasks. Two new administrators are created who in total have the same permissions in user groups 6 and 7 as the default administrator ADMIN previously had on their own. The corresponding role-based user profiles have been set up in the CAT for this purpose, see *User Profiles* in the [CryptoServer - CAT Manual \(p. 284\)](#).

After this, the default administrator ADMIN can be deleted. From that point onwards, two people will be required (two-person rule) to perform the CryptoServer administration tasks.



Additional administrator users with password mechanism may be created, but they will be deleted in case of an alarm.

At any later date, the device will only allow to delete a user with user management rights if the following two conditions are fulfilled:

- In the specific user group 7 the sum of the permissions of all remaining users who use a public key as authentication token is still above the minimum permission level of 2.

- In the specific user group 6 the sum of the permissions of all remaining users who use a public key as authentication token is still above the minimum permission level of 1.

These are the necessary conditions to retain an administrable device. Otherwise, the command for user deletion will be rejected with an error code.



In an emergency case, when the administration key is lost (e.g. smartcard(s) with private part of customer-individual administrator key is lost), the device can be reset by executing the `csadm Clear=DEFAULT` command. In this case, the device deletes the user database and creates a new one that contains the default ADMIN user described above.

5 Configuration

5.1 Setting the CRYPTOSERVER Variable

For setting up and administering the CryptoServer with the command-line tool `csadm`, the address of the device to connect to is required. This address is specified for every command in the `Dev=` command parameter, see *Syntax of csadm* in the *u.trust Anchor - csadm Manual*.

Example:

```
csadm Dev=3001@194.168.4.107 GetState
```

Optionally and for more convenience, the address of the u.trust Anchor cHSM can be set permanently in a CRYPTOSERVER environment variable. In this case, the `Dev=` parameter in the `csadm` command can be omitted.

The CRYPTOSERVER environment variable is not a default for the following parameters:

- Device parameter in the `cs_pkcs11_R2.cfg` file
This parameter is used for actions initiated by P11CAT or p11tool2. For details, see *CryptoServer – PKCS#11 P11CAT Manual*
- Device parameter in the `cs_cng.cfg` file
This parameter is used for actions initiated by CSP/CNG. For details, see *CryptoServer - CryptoServer CSP and CNG Key Storage Provider*
- Device parameter in the `csxlan.conf` file
This parameter is used for some communication situations.



The commands in this section are only examples. Depending on your shell, other commands might be suitable.

Setting CRYPTOSERVER

- For the current command line

Example:

```
export CRYPTOSERVER=192.168.1.1
```

- For all future command-lines

- a. Append an export command to the `.bashrc`



It is very important that you use `>>` in the following command. It appends the preceding text to the specified file. Do not use `>`, because it overwrites the entire file.

Example:

```
echo export CRYPTOSERVER=192.168.1.1 >> ~/.bashrc
```

- b. Apply the changes to the current command-line as well by performing the following command.

Example: `. ~/.bashrc`

Verifying CRYPTOSERVER

Perform one of the following commands to verify the value of the CRYPTOSERVER environment variable.

- For the current command line

Example 1: `env | grep CRYPTOSERVER`

Example 2: `printenv | grep CRYPTOSERVER`

Example 3: `echo $CRYPTOSERVER`

Deleting CRYPTOSERVER

- For the current command-line

Example:

```
export CRYPTOSERVER=
```

- For all future command-lines

Remove the following line from the `~/.bashrc` file.

Example:

```
export CRYPTOSERVER=192.168.1.1
```

5.2 Using a Local PIN Pad for a Remote CryptoServer

This chapter describes in detail how to use local PIN pads with a remote CryptoServer, see [Scenario 3: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN Without Blocking Firewall](#) (p. 32) and [Scenario 4: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN With Blocking Firewall](#) (p. 33).



Using a local PIN pad for a remote CryptoServer can only be applied by using command-line tools such as `csadm`, `p11tool2` and `cxtool` and the PKCS#11 CryptoServer Administration Tool (P11CAT). The CryptoServer Administration Tool (CAT) supports this feature as of SecurityServer 4.30/CryptoServer SDK 4.30.

5.2.1 Requirements

The following requirements must be fulfilled:

- Authentication mechanism
Using a local PIN pad for a remote CryptoServer is not supported for the RSA smartcard authentication mechanism, see [Authentication Mechanisms](#) (p. 54). However, if a new user should be created and this new user should use RSA smartcard authentication after his creation, using a local PIN pad for a remote CryptoServer is supported during the execution of the `csadm AddUser` command, see *AddUser* in the [CryptoServer - csadm Manual](#) (p. 284).
- Local computer
 - A PIN pad must be connected to the local computer via an USB.
 - The PIN pad driver must have been installed, see [Installing the PIN Pad Driver](#) (p. 139).
 - The needed user authentication key for authentication must be available on a smartcard. Use the `csadm GetCardInfo` command to verify this, see *GetCardInfo* in the [CryptoServer - csadm Manual](#) (p. 284).
 - The user authentication key for the default ADMIN user with permission mask 22000000 has been provided on each smartcard delivered by Utimaco IS GmbH.

- For a connection to a remote CryptoServer with a firewall, an SSH client, for example, PuTTY, must have been installed. For [Scenario 3: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN Without Blocking Firewall \(p. 32\)](#), any similar terminal software client, for example, Remote Desktop Connection, is sufficient.

- PIN pad daemon files

These files are:

- `exe` for Windows and `ppd` for Linux: PIN pad daemon executable
- `cfg` : Configuration file
- `pwd` : Password file

As of SecurityServer 4.30/CryptoServer SDK 4.30, the PIN pad daemon files are provided in the `\Software\ppd` directory within the product bundle. If the CryptoServer installer file for Windows is used, `CryptoServerSetup-<version>.exe`, provided on the SecurityServer product CD, and the PPD check box is selected in the installer window, the `ppd.exe` file is copied to the `<install_dir>\Administration` directory and the `ppd.cfg` file and the `ppd.pwd` file are copied to the `...\ProgramData\Utimaco\PPD` directory during the installation.

For SecurityServer/CryptoServer SDK versions earlier than 4.30, retrieve the PIN pad daemon files from the support team of Utimaco, see [Contact Address for Support Queries \(p. 283\)](#).

- Remote computer
 - Either a CryptoServer PCIe version 4.10 or later has been installed or a connection to a CryptoServer LAN version 4.10 or later is provided.
 - The CryptoServer drivers and the needed CryptoServer tools (for example, `csadm`) have been installed.
 - Network client

If Windows runs on the remote computer, a network client should be installed to be able to verify a connection to a certain port on the local computer. Use, for example, Telnet or PuTTY. Telnet can be enabled by selecting **Start > Control Panel > Programs > Programs and Features > Turn Windows features on or off > Telnet** and clicking **OK**. If you use PuTTY, it must be downloaded from the internet and installed.

On Linux, netcat or ncat can be used. Check the package manager of your Linux distribution for the package to be installed.

- SSH daemon
 - For [Scenario 3: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN Without Blocking Firewall \(p. 32\)](#), any terminal server software similar to SSH, for example, Remote Desktop Connection, is sufficient.
 - For [Scenario 4: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN With Blocking Firewall \(p. 33\)](#), an SSH daemon must have been installed. For a remote Windows computer, you have to install an SSH daemon (for example, OpenSSH) on the remote computer. Installing and operating this SSH daemon is out of scope of this documentation.
- Network
 - For [Scenario 3: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN Without Blocking Firewall \(p. 32\)](#), the port of the terminal software (for example, port 22 for ssh) and port 6070 (default; configurable) must not be blocked by a firewall between the local computer and the remote computer.
 - For [Scenario 4: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN With Blocking Firewall \(p. 33\)](#), port 22 must not be blocked by a firewall between the local computer and the remote computer.

5.2.2 Configuring the PIN Pad Daemon (PPD)

When started, the PIN pad daemon reads its configuration values from the `ppd.cfg` configuration file. To make this happen automatically, that means, without the configuration file being specified in the start command, the configuration file must be provided in one of the following directories (`<configuration path>`).

- Windows

`%USERPROFILE%\AppData\Local\Utimaco`

`%USERPROFILE%\AppData\Local\Utimaco\PPD`

`%ALLUSERSPROFILE%\Utimaco`

`%ALLUSERSPROFILE%\Utimaco\PPD`

- **Linux**

`$HOME` (the home directory of the user)

`$HOME/ppd`

`/etc`

`/etc/ppd`

In the configuration file, lines starting with `#` are comments. There are two sections, `[Global]` and `[Server]`. The following items are configured in this file.

- **PIN Pad Daemon Log File**

The PIN pad daemon has its own log file. `LogFile` defines the path to this file.

Sample log entries:

Starting the PIN pad daemon including the configuration parameters

```
PPD is starting...  
...  
PPD is running
```

Terminating the PIN pad daemon.

```
PPD is stopping...
```

Failed authentication attempt in a command that has been received by using the PIN pad daemon.

```
authentication failed
```

Error message of the PIN pad API due to a command that has been received by using the PIN pad daemon.

```
Error B91D501F  
  
PIN pad API  
  
USB  
  
errno = 31
```

A connection has been established to perform a command.

Example: TCP connection from 192.123.123.123:49333 accepted on port 6070

Default configuration path value:

- Windows

<TEMP>\ppd.log, that means, typically C:\Users\<user name>\AppData\Local\Temp\ppd.log

Linux

<TEMP>/ppd.log, that means, typically /tmp/ppd.lo

- **Logging level**

LogLevel defines which information is logged in the log file defined by the Logfile parameter.

1: Error

2: Warning

3: Info

4: Trace

Default value: 2

- **Log file size**

LogSize is the maximum size of the log file in byte before a new log file is created.

Default value: 8000000

- **Authentication mechanism**

The PIN pad API on the remote computer communicates with the local computer over a TCP connection (default; This can be changed to UDP). If the `AuthMech` parameter in the configuration file is set to 1, this communication is encrypted with secure messaging with an AES-256 key in CBC Mode. In this case, exchanging the session key is authenticated using SHA-512 and with the password defined in the password file that is defined by the Passfile parameter in the configuration file.

If `AuthMech` is set to 0, no secure messaging is done at all.



The authentication mechanism mentioned here must not to be confused with the authentication mechanisms of the device.

▪ Password File

The `Passfile` parameter only applies if the `AuthMech` parameter has been set to 1.

The `Passfile` parameter indicates the path to the password file. This file contains only the Password parameter indicating the password for authenticating the exchange of the session key for secure messaging.

The maximum length of the password is 79. The password must not contain any blanks. The hashtag character (#) in the password is not supported, since it will be interpreted as the beginning of a comment. Other special characters in the password, for example, the following ones, are supported:

Colon: :

Dot: .

Hyphen: -

Comma: ,

The Passfile parameter's default value is `<configuration path>\ppd.pwd` on Windows and `<configuration path>/ppd.pwd` on Linux.

In the cases of the following list, no authentication is done for exchanging the session key of secure messaging, that means, an anonymous session is used:

The password file defined by the Passfile parameter cannot be found.

The Password parameter in the password file is empty.

If authentication is done for exchanging the session key for secure messaging, the password defined in the password file must be entered when a command is performed by a tool/API. If no authentication is done, a password must not be entered.

▪ Port Number

Port number the PIN pad daemon listens on for incoming connections

For a connection of a local PIN Pad to local CryptoServer Tools/APIs and a remote CryptoServer LAN, this port must be opened in all firewalls between the local computer

and the remote computer. For [Scenario 3: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN Without Blocking Firewall \(p. 32\)](#), it is not necessary that this port is opened in the firewalls because the firewalls are tunneled by using SSH.

Default value: **6070**

- **Protocol**

The protocol to be used for the communication between the PIN pad daemon on the local computer and the CryptoServer tools/APIs on the remote computer.

Possible values: `TCP` or `UDP`

Default value: `TCP`

- **IPv6**

IPv6 indicates whether IPv6 is supported. The PIN pad daemon is able to use IPv6 with either TCP or UDP. We recommend using TCP with IPv4.

Possible values: **0**: Disabled, **1**: Enabled

Default value:

- **Timeout**

Timeout in seconds until an idle connection between the PIN pad daemon and the CryptoServer tool/API is terminated. A timeout value of 0 indicates that an idle connection is never terminated.

Default value: 600

Sample Configuration File

```
[Global]
```

```
# Log file [default: <TMP>/ppd.log]
```

```
# LogFile=/var/log/ppd.log
```

```
# Log level (0:NONE, 1:ERROR, 2:WARNING, 3:INFO, 4:TRACE) [default: 2]
```

```
LogLevel=3
```

```
# Maximum size the log file may grow before rotation [default: 8MB]
LogSize=1000000

# Authentication mechanism: 0: NONE 1:AUTH_MECH_SHA512 [default: 1]
# AuthMech=0

# Password file for authentication method AUTH_MECH_SHA512
# [default: <CONFIGPATH>/ppd.pwd]
# Passfile=/etc/ppd/ppd.pwd

[Server]
# Port number [default: 6070]
Port=6070
# Protocol, TCP or UDP [default: TCP]
Protocol=TCP

# Enable IPv6 (0: Disable, 1: Enable)
#IPv6=1 # [default: 0]

# Timeout until an idle connection is closed in seconds [default: 600]
IdleTimeout=60
```

5.2.3 Starting the PIN pad daemon on Windows

If the following requirements are met, the ppd.exe file is copied to the `C:\Program Files\Utimaco\SecurityServer\Administration` directory, the `ppd.cfg` file and the `ppd.pwd` file are copied to the `C:\ProgramData\Utimaco\PPD` directory during the installation:

- The SecurityServer/CryptoServer SDK version is 4.40 or later.

- A Windows operating system is used.

See the release notes for detailed information about the supported versions.

- The CryptoServer installer file, `CryptoServerSetup-<version>.exe`, provided on the SecurityServer product CD, is used for the CryptoServer installation.
- The PPD check box is selected in the installer window.

Perform the following steps to start the PIN pad daemon on Windows:

1. If the PPD was not installed via setup:

- Copy the `ppd.exe` file to an arbitrary target directory, for example, `C:\Utimaco\PPD`.
A path with blanks is not supported.
- Copy the `cfg` and `ppd.pwd` files to one of the directories where they are read automatically.
- Go to the target directory.
- Proceed with step 2.

2. If the PPD was installed via setup:

- Configure the PPD as described in section [Configuring the PIN Pad Daemon \(PPD\)](#) (p. 158)

- To run the PIN pad daemon once, perform the following command.

If the `ppd.cfg` configuration file is in one of the directories where it is automatically read:

```
ppd -foreground
```

Otherwise:

```
ppd -config=<configuration path> -foreground
```

`<configuration path>` is the path to the `ppd.cfg` configuration file.

Example:

```
ppd -config=C:\ProgramData\Utimaco\PPD\ppd.cfg -foreground
```

If you need help, perform the `ppd -help` command. It produces the following output:

```
@(#) ppd PinPAD Daemon 1.1.0 [X64] (Sep 14 2017)
```

```
valid parameters are:  
-config=<path> - explicitly determine config file path  
-install       - install PPD as service  
-remove       - remove service  
-foreground    - execute PPD without service installation  
-version       - show version information  
-help         - show this text
```

c. The following dialog opens when the PPD is run the first time.

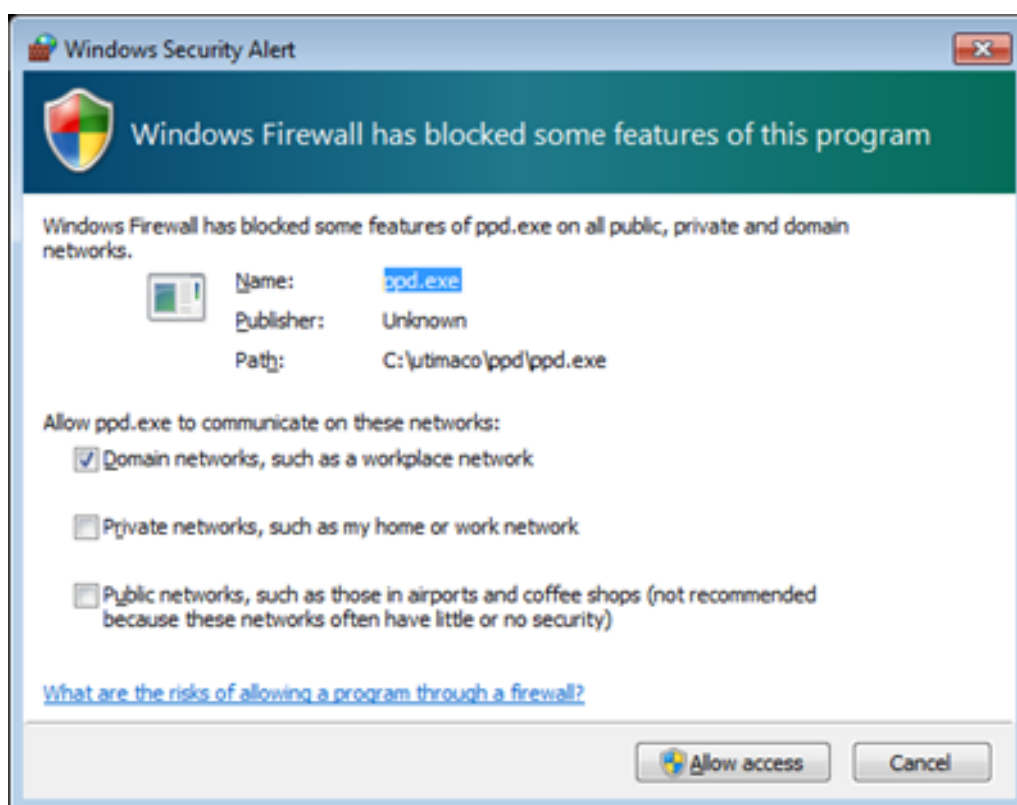


Figure 22 : Windows Firewall Warning

- d. Click **Allow access** to allow the firewall communication for the PIN pad daemon.
- e. If you want to install the PIN pad daemon as a service, execute the following substeps.
- f. Click **Start > All Programs > Accessories > Command prompt** with the right mouse button.
- g. Select **Run as administrator**.

- h. Perform the following command in this command-line. If the `ppd.cfg` configuration file is in one of the directories where it is automatically read:

```
ppd -config=<configuration path> -install
```

Example:

```
ppd -config=C:\ProgramData\Utimateco\PPD\ppd.cfg -install
```

A path with blanks is not supported.

Otherwise:

```
ppd -config=<configuration path> -install
```

`<configuration path>` is the path to the `ppd.cfg` configuration file.

Example:

```
ppd -config=C:\ProgramData\Utimateco\PPD\ppd.cfg -install
```

- i. Perform the `net start ppd` command in this command-line. Otherwise it will be started after system restart.
- j. Perform the `msc` command to verify that the service has started.

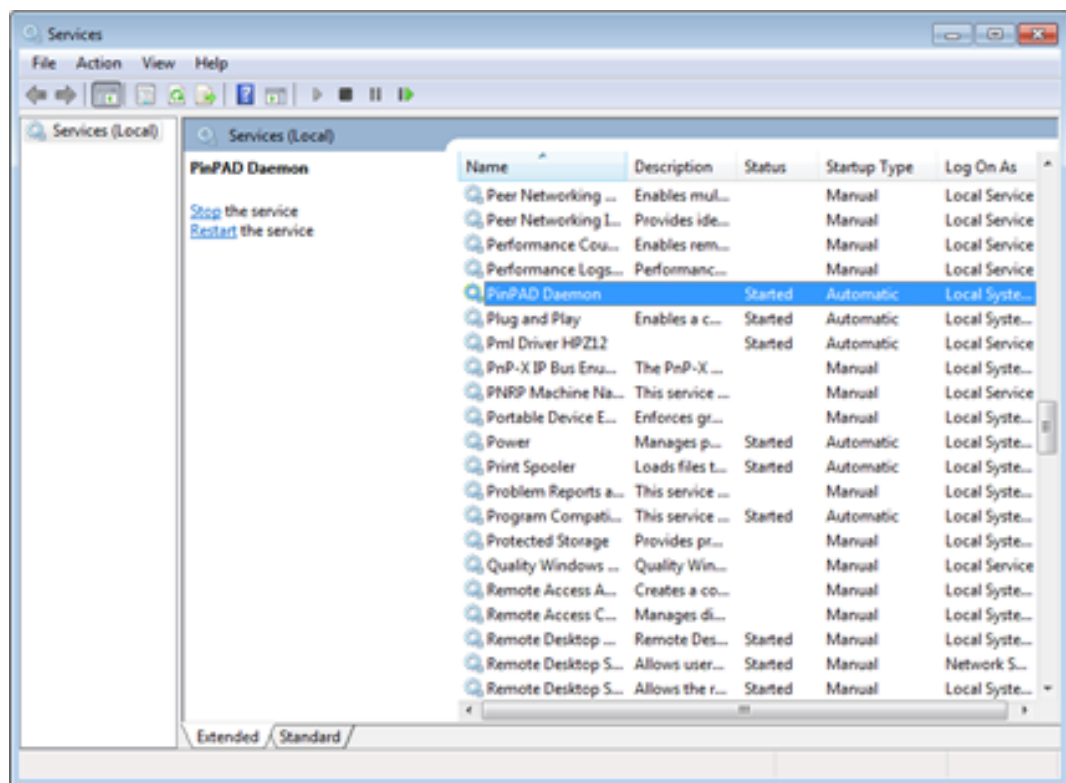


Figure 23 : Started PIN pad daemon in the services list

- k. If you want to manually stop this service at a later date, perform the `net stop ppd` command as an administrator.
- l. If you want to uninstall this service at a later date, perform the `ppd -remove` command as an administrator.

The start of the PIN pad daemon is logged in the logfile.

```

19.02.2018 16:09:25 | ppd_init          | @(#) ppd PinPAD Daemon 1.1.0 [X64]
(Sep 14 2017)
19.02.2018 16:09:25 | ppd_init          | I: configuration file : C:
\ProgramData\Utimaco\PPD\ppd.cfg
19.02.2018 16:09:25 | ppd_init          | I: LogFile           : C:
\ProgramData\Utimaco\PPD\ppd.log
19.02.2018 16:09:25 | ppd_init          | I: LogLevel          : 3
19.02.2018 16:09:25 | ppd_init          | I: LogSize           : 8388608
19.02.2018 16:09:25 | ppd_init          | I: AuthMech          : 1
19.02.2018 16:09:25 | ppd_init          | I: Passfile          : C:
\ProgramData\Utimaco\PPD\ppd.pwd
19.02.2018 16:09:25 | ppd_init          | I: Port              : 6070
19.02.2018 16:09:25 | ppd_init          | I: Protocol          : 6
19.02.2018 16:09:25 | ppd_init          | I: IPv6              : 0
19.02.2018 16:09:25 | ppd_init          | I: IdleTimeout       : 60
19.02.2018 16:09:25 | ppd_init          | I: PPD is starting...
19.02.2018 16:09:25 | ppd_main_task     | I: PPD is running

```

5.2.4 Starting the PIN Pad Daemon on Linux

To start the PIN pad daemon on Linux, perform the following steps.

1. Copy the `ppd` file to an arbitrary target directory, for example, `/usr/bin` or `/usr/sbin`. A path with blanks is not supported.
2. Copy the `ppd.cfg` and `ppd.pwd` files to one of the directories where they are read automatically (see Chapter 5.1.9.2, "Configuring the PIN Pad Daemon"). A path with blanks is not supported.
3. To run the PIN pad daemon in the foreground, perform the following command. If the `ppd.cfg` configuration file is in one of the directories where it is automatically read:
`ppd -foreground`

Otherwise:

```
ppd -config=<configuration path> -foreground
```

`<configuration path>` is the path to the `ppd.cfg` configuration file.

Example:

```
ppd -config=/etc/ppd/ppd.cfg -foreground
```

A path with blanks is not supported.

As an alternative, if you want to run the PIN pad daemon in the background, that is, as a daemon, perform the following command.

```
ppd -config=<configuration path>
```

See chapter [Configuring the PIN Pad Daemon \(PPD\)](#) (p. 158) for details about the configuration file.

If you need help, perform the `ppd -help` command. It produces the following output:

```
@(#) ppd PinPAD Daemon 1.1.0 [X64] (Sep 14 2017)

valid parameters are:
-config=<path> - explicitly determine config file path
-foreground   - execute PPD without service installation
-version      - show version information
-help         - show this text
```

4. The start of the PIN pad daemon is logged in the logfile.

```
19.02.2018 16:09:25 | ppd_init          | @(#) ppd PinPAD Daemon 1.1.0
19.02.2018 16:09:25 | ppd_init          | [X64] (Sep 14 2017)
19.02.2018 16:09:25 | ppd_init          | I: configuration file : /etc/pwd/
19.02.2018 16:09:25 | ppd_init          | ppd.cfg
19.02.2018 16:09:25 | ppd_init          | I: LogFile           : /etc/pwd/
19.02.2018 16:09:25 | ppd_init          | ppd.log
19.02.2018 16:09:25 | ppd_init          | I: LogLevel          : 3
19.02.2018 16:09:25 | ppd_init          | I: LogSize           : 8388608
19.02.2018 16:09:25 | ppd_init          | I: AuthMech          : 1
19.02.2018 16:09:25 | ppd_init          | I: Passfile          : /etc/pwd/
19.02.2018 16:09:25 | ppd_init          | ppd.pwd
19.02.2018 16:09:25 | ppd_init          | I: Port              : 6070
19.02.2018 16:09:25 | ppd_init          | I: Protocol          : 6
19.02.2018 16:09:25 | ppd_init          | I: IPv6              : 0
19.02.2018 16:09:25 | ppd_init          | I: IdleTimeout       : 60
19.02.2018 16:09:25 | ppd_init          | I: PPD is starting...
19.02.2018 16:09:25 | ppd_main_task     | I: PPD is running
```

5. It might be useful to provide start/stop/init scripts.

5.2.5 Connecting a Local PIN Pad on Windows to Remote CryptoServer Tools/APIs on Windows



This chapter describes only [Scenario 3: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN Without Blocking Firewall](#) (p. 32). To apply [Scenario 4: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN With Blocking Firewall](#) (p. 33), you have to install an SSH daemon (for example, OpenSSH) on the remote computer. Installing and operating this SSH daemon is out of scope of this documentation.

This chapter describes how to connect a local Windows computer to a remote Windows computer so that the local PIN pad can be used to authenticate a command on the remote computer.

1. Click **Start > All Programs > Accessories > Remote Desktop Connection** on the local computer.
2. In the Computer field, enter the IP address or name of the remote computer.
3. Click **OK**.
4. Enter the username.
5. Click **OK**.
6. Perform the following substeps to verify whether the connection works without problems between the local computer and port on the remote computer that has been defined in the configuration file.
 - a. Start the network client, for example, Telnet or PuTTY, on the remote computer. The following steps describe the usage of PuTTY.
 - b. Enter the name or the IP address of the local computer under **Host Name (or IP address)**
 - c. In the **Port** field, enter the port that is configured in the configuration file.
 - d. Under **Connection type**, select **Raw**.

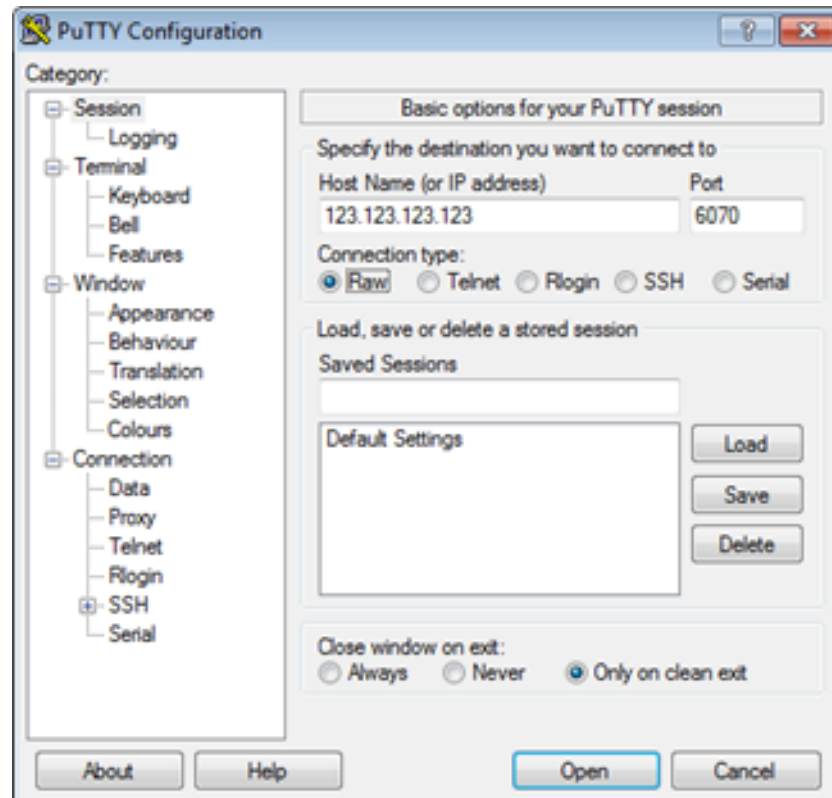


Figure 24 : PuTTY – Connection verification

- e. Click **Open**.
- f. On the local computer, open the PIN pad daemon log file (default name: `ppd.log`)

If a new line according to the following example has been generated, this means that a connection between PuTTY on the remote computer and the PIN pad daemon (port 6070 in the example) on the local computer has been established.

```
19.02.2018 16:09:27 | ppd_main_task | I: TCP connection from
127.0.0.1:50003 accepted on port 6070
```

The connection between these two ports has been marked in red in the following figure.

7. Now you are logged in from the local computer to the remote computer and the connection has been verified. In a command-line on the remote computer, you can perform any tool/API command that needs authentication by a key stored on the smartcard in the PIN pad connected to the local computer.
As a first test, let `csadm` display which keys are stored on the smartcard.

- a. Open a command-line on the remote computer.
- b. In this command-line, execute a command according to the following example.

```
csadm GetCardInfo=:cs2:cjo:USB0@1.1.1.1/mypwd
```

Consider one special aspect of the csadm `GetCardInfo` command. This command only needs an installed csadm on the remote computer but it does not need any contact to a device.

The PIN pad display shows the following:

```
Insert Smartcard
```

```
Press OK/Cancel
```

- c. Insert the smartcard into the PIN pad.
- d. Press **OK** on the PIN pad.

Example output in the command-line:

```
RSA-Key:    Utimaco IS GmbH / Init-Dev-1-Key
```

```
ECC-Key:    EcKey
```

```
RSA-Backup: KeyInfo02 #1
```

```
ECC-Backup: EcKey #1
```

```
MBK
```

8. A second test retrieves the authentication status (permission mask) of the default administration user, ADMIN. His user authentication key has been provided on every smartcard delivered by Utimaco IS GmbH.

- a. Perform a command according to the following example.

```
csadm Dev=PCI:0 LogonSign=ADMIN,:cs2:cjo:USB0@1.1.1.1 GetAuthState
```

The PIN pad display shows the following:

```
Insert Smartcard
```

```
Press OK/Cancel
```

- b. Insert the smartcard.
- c. Press **OK** on the PIN pad.

Example output:

```
current AUTH state: 22000000
```

```
user: ADMIN
```

5.2.6 Connecting a Local PIN Pad on Windows to Remote CryptoServer Tools/APIs on Linux

This chapter describes how to connect a local Windows computer to a remote Linux computer so that the local PIN pad can be used to authenticate a command on the remote computer.

1. Start the SSH client, for example, PuTTY, on the local computer.

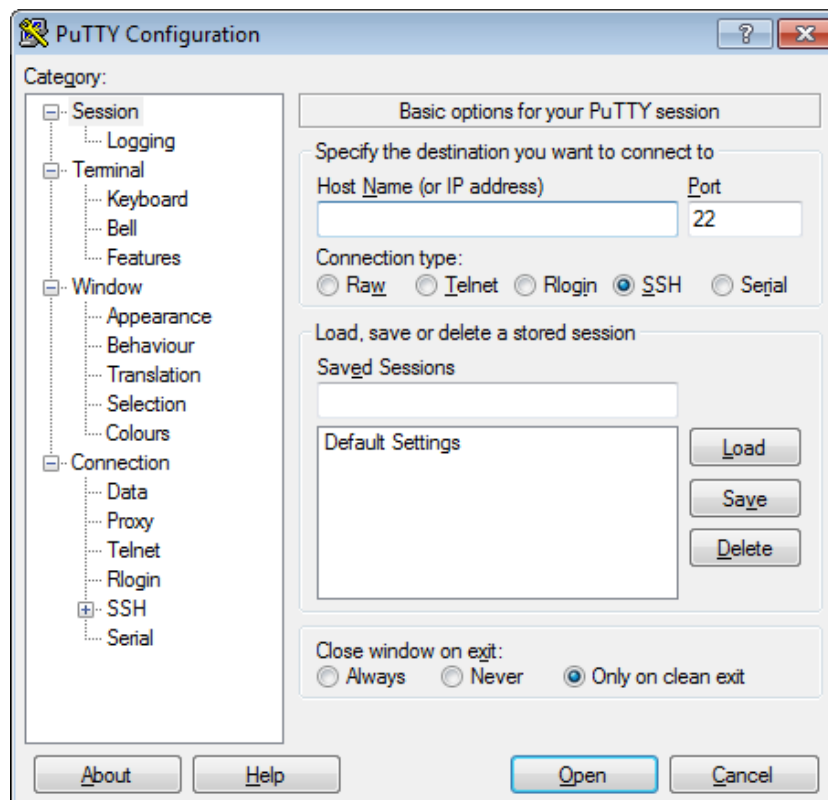


Figure 25 : PuTTY started

2. In the Host Name (or IP address) field, enter the name or IP address of the remote computer.

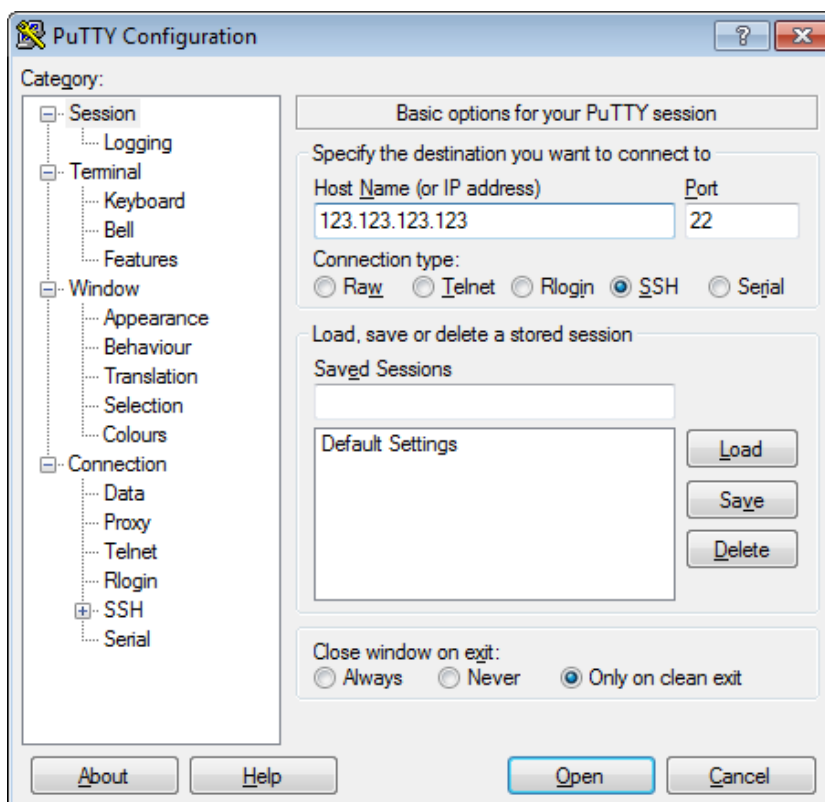


Figure 26 : PuTTY – IP address entered

3. Ensure that the value in the **Port** field is 22.
4. If you want to enter the name of the user to be logged in on the remote computer, open **Connection > Data** in the **Category** field, and enter it in the **Auto-login username** field.

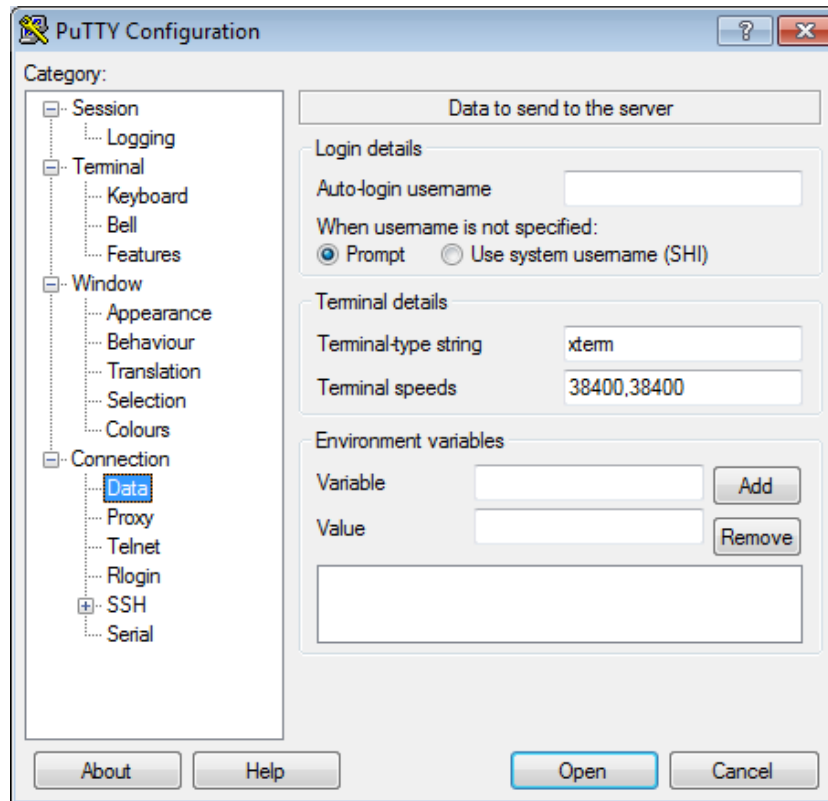


Figure 27 : PuTTY - Username

5. If [Scenario 4: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN With Blocking Firewall \(p. 33\)](#) is applied to e.g. tunnel firewalls between the local computer and the remote computer, perform the following substeps.
 - a. In the Category field, open **Connection > SSH > Tunnels**.

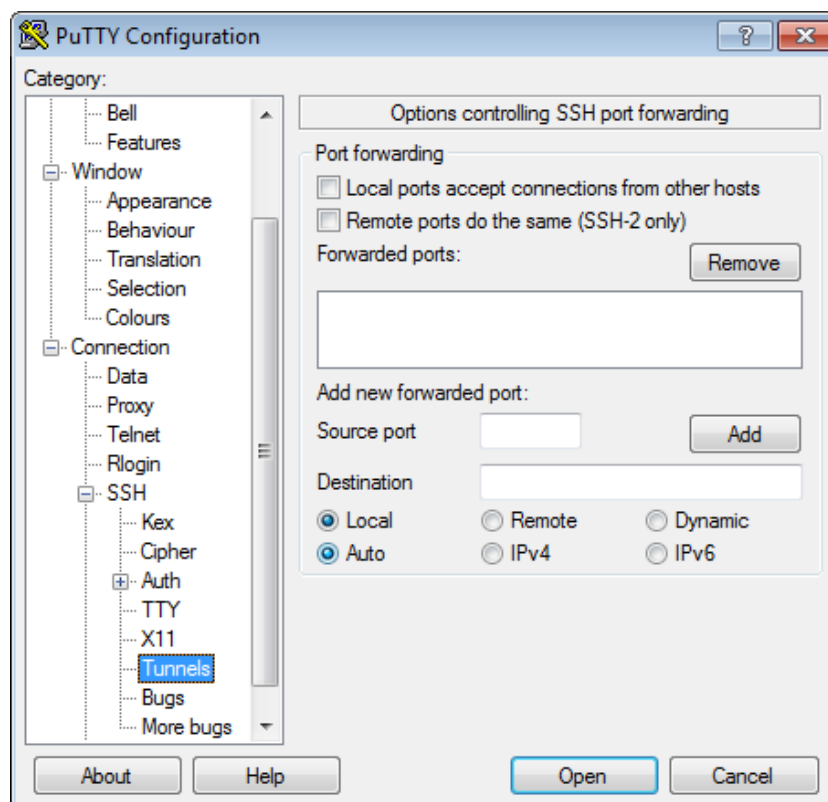


Figure 28 : PuTTY - Tunnels

- b. In the **Source port** field, enter the number of the port on which the SSH daemon listens for incoming connections on the remote computer, for example, 6070.
- c. In the **Destination** field, enter the `127.0.0.1:<port>`. `<port>` is the port number that has been defined in the configuration file.
Example: `127.0.0.1:6070`
- d. Select the **Remote** radio button.

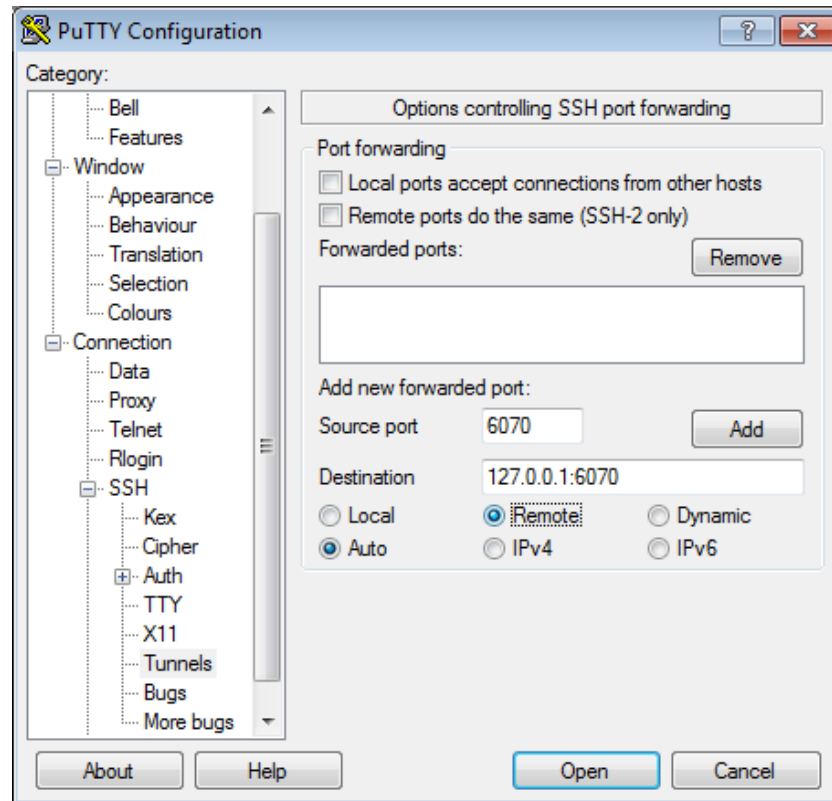


Figure 29 : PuTTY - Tunnels entered

- e. Click the **Add** button.
6. Click the **OK** button.
7. A command-line window opens.
Example:
Using username "user01".
user01@123.123.123.123's password:
8. Enter the user's password.
9. Perform the following substeps to verify whether the connection works without problems between `<port>` on the local computer and `<port>` on the remote computer. `<port>` is the port that is defined in the configuration file.
 - a. Enter the following `netcat` command.
`netcat <IP address> <port>`

- `<IP address>`

If you apply [Scenario 3: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN Without Blocking Firewall](#) (p. 32), `<IP address>` is the IP address of the local computer. If you apply [Scenario 4: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN With Blocking Firewall](#) (p. 33), use `127.0.0.1` instead.

- `<port>`

`<port>` is the number of the port that has been configured in step 5c).

Example:

- [Scenario 3: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN Without Blocking Firewall](#) (p. 32):

```
netcat 1.1.1.1 6070
```

- [Scenario 4: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN With Blocking Firewall](#) (p. 33):

```
netcat 127.0.0.1 6070
```

- b. Open the PIN pad daemon log file (default name: `ppd.log`).

If a new line according to the following example has been generated, this means that a connection between the netcat on the remote computer and the PIN pad daemon (port 6070 in the example) on the local computer has been established.

```
19.02.2018 16:09:27 | ppd_main_task | I: TCP connection from  
127.0.0.1:50003 accepted on port 6070
```

The connection between these two ports has been marked in red in the following figures.

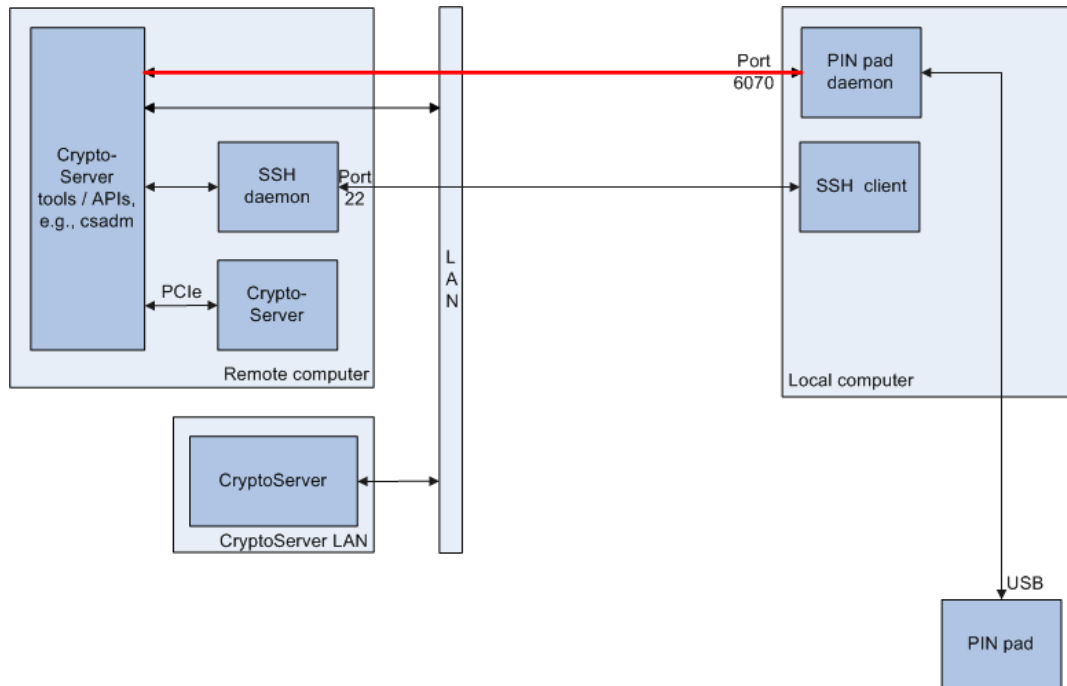


Figure 30 : Connection Verification

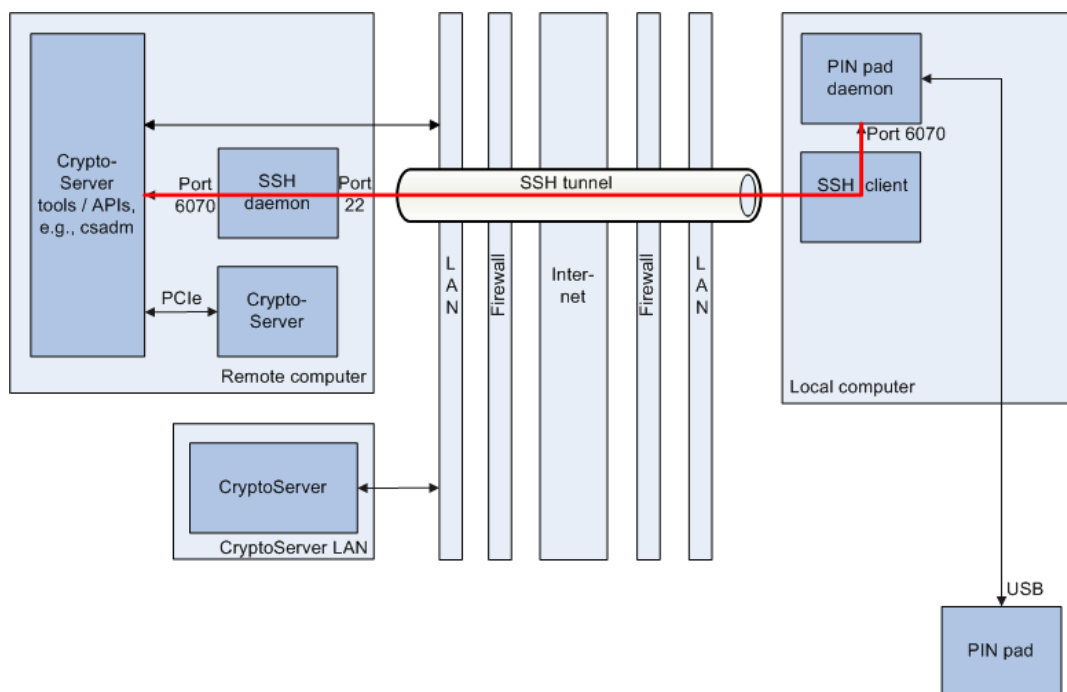


Figure 31 : Connection Verification

- c. Enter **CTRL + C** in the command-line to abort the `netcat` command.
10. Now you are logged in from the local computer to the remote computer and the connection between the local computer and the remote computer has been verified. In the command-line, you can perform any CryptoServer tool/API on the remote computer that needs authentication by a key stored on the smartcard in the PIN pad connected to the local computer.

As a first test, let `csadm` show you which keys are stored on the smartcard.

- a. Perform a command according to the following examples.

- [Scenario 3: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN Without Blocking Firewall \(p. 32\)](#):

```
csadm GetCardInfo=:cs2:cjo:USB0@1.1.1.1/mypwd
```

- [Scenario 4: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN With Blocking Firewall \(p. 33\)](#):

```
csadm GetCardInfo=:cs2:cjo:USB0@127.0.0.1/mypwd
```

Consider one special aspect of the `csadm GetCardInfo` command. This command only needs an installed `csadm` on the remote computer but it does not need any contact to a CryptoServer.

The display of the PIN pad shows the following.

```
Insert Smartcard
Press OK/Cancel
```

- b. Insert the smartcard into the PIN pad.

- c. Press the **OK** button of the PIN pad.

Example output in the command-line:

```
RSA-Key:    Utimaco IS GmbH / Init-Dev-1-Key
ECC-Key:    EcKey
RSA-Backup: KeyInfo02 #1
ECC-Backup: EcKey #1
MBK:
```

11. A second test retrieves the authentication status (permission mask) of the default administration user, ADMIN. His user authentication key has been provided on every smartcard delivered by Utimaco IS GmbH.

- a. Perform a command according to the following examples.

- [Scenario 3: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN Without Blocking Firewall \(p. 32\)](#):

```
csadm Dev=/dev/cs2.0 LogonSign=ADMIN,:cs2:cjo:USB0@1.1.1.1
GetAuthState
```

- **Scenario 4: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN With Blocking Firewall (p. 33)**4:

```
csadm Dev=/dev/cs2.0 LogonSign=ADMIN,:cs2:cjo:USB0@127.0.0.1
GetAuthState
```

The display of the PIN pad shows the following.

```
Insert Smartcard
Press OK/Cancel
```

- Insert the smartcard into the PIN pad.
- Press the **OK** button of the PIN pad.

Example output:

```
current AUTH state: 22000000
user: ADMIN
```

- If you want to terminate the connection to the remote computer, enter the `exit` command.

5.2.7 Connecting a Local PIN Pad on Linux to Remote CryptoServer Tools/APIs on Windows

This chapter describes how to connect a local Linux computer to a remote Windows computer so that the local PIN pad can be used to authenticate a command on the remote computer.



This chapter describes only **Scenario 3: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN Without Blocking Firewall (p. 32)**. To apply **Scenario 4: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN With Blocking Firewall (p. 33)**, you have to install an SSH daemon (for example, OpenSSH) on the remote computer. Installing and operating this SSH daemon is out of scope of this documentation.

- Ensure that an SSH server is running on the remote Windows computer.
- Perform the following command in a command-line on the local Linux computer to establish an SSH connection to the remote computer.

```
ssh <user>@<remote IP address>
```

Example:

```
ssh myname@123.123.123.123
```

3. Enter the user's password for the remote computer.
4. Perform the following substeps to verify whether the connection works without problems between the remote computer and port on the local computer that has been defined in the configuration file.

As already described in chapter [Requirements \(p. 156\)](#), a network client should be installed to be able to verify a connection to a certain port on the local computer. Use, for example, Telnet or PuTTY. PuTTY is described in the example described here.

- a. Enter the following command.

```
putty.exe -raw <IP address> <port>
```

- `<IP address>`
`<IP address>` is the IP address of the local computer.
- `<port>`
`<port>` is the number of the port that is configured in the configuration file.

Example:

```
putty.exe -raw 127.0.0.1 6070
```

- Open the PIN pad daemon log file (default name: `ppd.log`).
If a new line according to the following example has been generated, this means that a connection between the PuTTY on the remote computer and the PIN pad daemon (port 6070 in the example) on the local computer has been established.

```
19.02.2018 16:09:27 | ppd_main_task | I: TCP connection from  
127.0.0.1:50003 accepted on port 6070
```

The connection between these two ports has been marked in red in the following figures.



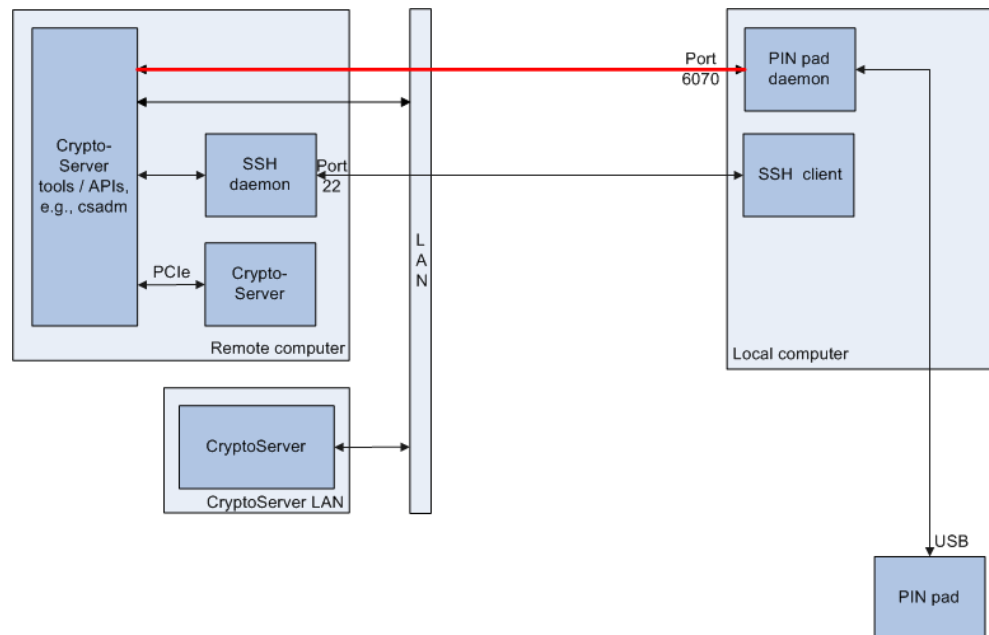


Figure 32 : Connection Verification

5. Now you are logged in from the local computer to the remote computer and the connection between the local computer and the remote computer has been verified. In the command-line, you can perform any CryptoServer tool/API on the remote computer that needs authentication by a key stored on the smartcard in the PIN pad connected to the local computer.

As a first test, let csadm show you which keys are stored on the smartcard.

- a. Perform a command according to the following examples.

```
csadm GetCardInfo=:cs2:cjo:USB0@2.2.2.2/mypwd
```

Consider one special aspect of the csadm `GetCardInfo` command. This command only needs an installed csadm on the remote computer but it does not need any contact to a CryptoServer.

The display of the PIN pad shows the following.

```
Insert Smartcard
Press OK/Cancel
```

- b. Insert the smartcard into the PIN pad.
 - c. Press the **OK** button of the PIN pad.
- Example output in the command-line:

```
RSA-Key: Utimaco IS GmbH / Init-Dev-1-Key  
ECC-Key: EcKey  
RSA-Backup: KeyInfo02 #1  
ECC-Backup: EcKey #1  
MBK:
```

6. A second test retrieves the authentication status (permission mask) of the default administration user, ADMIN. His user authentication key has been provided on every smartcard delivered by Utimaco IS GmbH.

- a. Perform a command according to the following examples.

```
csadm Dev=PCI:0 LogonSign=ADMIN,:cs2:cjo:USB0@2.2.2.2 GetAuthState
```

The display of the PIN pad shows the following.

```
Insert Smartcard
```

```
Press OK/Cancel
```

- b. Insert the smartcard into the PIN pad.

- c. Press the OK button of the PIN pad.

```
Example output:
```

```
current AUTH state: 22000000
```

```
user: ADMIN
```

7. If you want to terminate the connection to the remote computer, enter the `exit` command.

5.2.8 Connecting a Local PIN Pad on Linux to Remote CryptoServer Tools/APIs on Linux

1. This chapter describes how to connect a local Linux computer to a remote Linux computer so that the local PIN pad can be used to authenticate a command on the remote computer.

To apply [Scenario 3: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN Without Blocking Firewall](#) (p. 32), perform the following command in a command-line on the local computer to establish an SSH connection to the remote computer.

```
ssh <user>@<remote IP address>
```

To apply [Scenario 4: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN With Blocking Firewall](#) (p. 33), perform the following command on the local computer to establish an SSH connection to the remote computer.

```
ssh -R <remote port>:127.0.0.1:<local port> <user>@<remote IP address>
```

- `<remote port>`
The port on which the SSH daemon listens for incoming connections on the remote computer
- `<local port>`
The port on the local computer that is configured in the configuration file
- `<user>`
The user to be logged in at the remote computer
- `<remote IP address>`
IP address of the remote computer
Examples:
 - [Scenario 3: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN Without Blocking Firewall \(p. 32\)](#):
`ssh myname@123.123.123.123`
 - [Scenario 4: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN With Blocking Firewall \(p. 33\)](#):
`ssh -R 6070:127.0.0.1:6070 myname@123.123.123.123`

2. Enter the user's password for the remote computer.
3. Perform the following substeps to verify whether the connection works without problems between the local computer and port on the remote computer that has been defined in the configuration file.

- a. Enter the following netcat command.

```
netcat <IP address> <port>
```

- `<IP address>`
If you apply [Scenario 3: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN Without Blocking Firewall \(p. 32\)](#), `<IP address>` is the IP address of the local computer. If you apply, [Scenario 4: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN With Blocking Firewall \(p. 33\)](#), use `127.0.0.1` instead.
- `<port>`
Again, `<port>` is the number of the port on which the SSH daemon listens for incoming connections on the remote computer.
Example:

- [Scenario 3: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN Without Blocking Firewall \(p. 32\)](#)

- :

```
netcat 1.1.1.1 6070
```

- [Scenario 4: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN With Blocking Firewall \(p. 33\):](#)

```
netcat 127.0.0.1 6070
```

- Open the PIN pad daemon log file (default name: `ppd.log`).

If a new line according to the following example has been generated, this means that a connection between the netcat on the remote computer and the PIN pad daemon (port 6070 in the example) on the local computer has been established.

```
19.02.2018 16:09:27 | ppd_main_task | I: TCP connection from  
127.0.0.1:50003 accepted on port 6070
```

The connection between these two ports has been marked in red in the following figures.

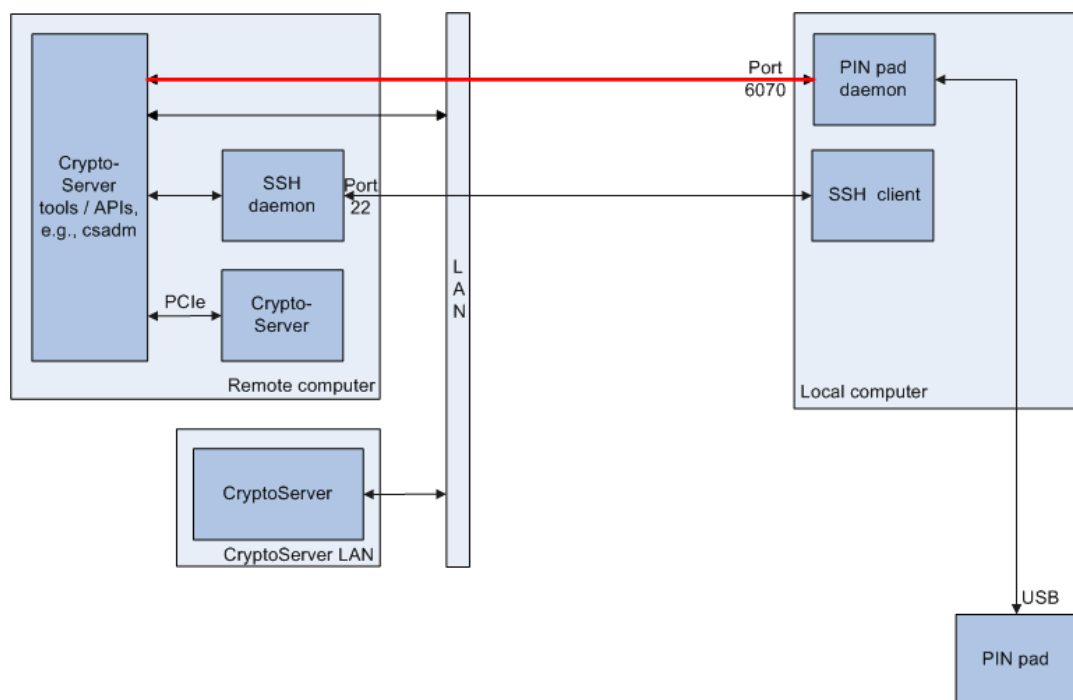


Figure 33 : Connection Verification

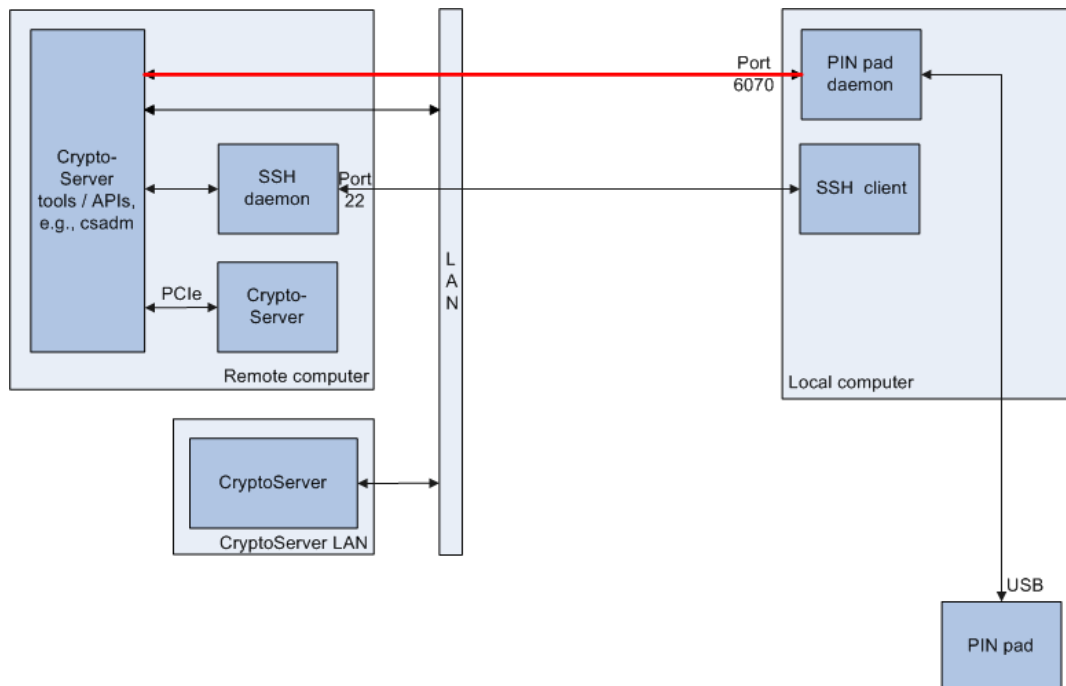


Figure 34 : Connection Verification

- c. Enter **CTRL + C** in the command-line to abort the netcat command.
4. Now you are logged in from the local computer to the remote computer and the connection between the local computer and the remote computer has been verified. In the command-line, you can perform any CryptoServer tool/API on the remote computer that needs authentication by a key stored on the smartcard in the PIN pad connected to the local computer. As a first test, let csadm show you which keys are stored on the smartcard.
 - a. Perform a command according to the following example.
 - **Scenario 3: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN Without Blocking Firewall (p. 32):**

```
csadm GetCardInfo=:cs2:cjo:USB0@1.1.1.1/mypwd
```
 - **Scenario 4: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN With Blocking Firewall (p. 33):**

```
csadm GetCardInfo=:cs2:cjo:USB0@127.0.0.1/mypwd
```

Consider one special aspect of the csadm `GetCardInfo` command. This command only needs an installed csadm on the remote computer but it does not need any contact to a CryptoServer.

The display of the PIN pad shows the following.

```
Insert Smartcard
```

```
Press OK/Cancel
```

- b. Insert the smartcard into the PIN pad.
- c. Press the **OK** button of the PIN pad.

Example output in the command-line:

```
RSA-Key:    Utimaco IS GmbH / Init-Dev-1-Key
ECC-Key:    EcKey
RSA-Backup: KeyInfo02 #1
ECC-Backup: EcKey #1
MBK:
```

5. A second test retrieves the authentication status (permission mask) of the default administration user, ADMIN. His user authentication key has been provided on every smartcard delivered by Utimaco IS GmbH.

- a. Perform a command according to the following example.

- [Scenario 3: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN Without Blocking Firewall \(p. 32\)](#):

```
csadm Dev=/dev/cs2.0 LogonSign=ADMIN,:cs2:cjo:USB0@1.1.1.1
GetAuthState
```

- [Scenario 4: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN With Blocking Firewall \(p. 33\)](#):

```
csadm Dev=/dev/cs2.0 LogonSign=ADMIN,:cs2:cjo:USB0@127.0.0.1
GetAuthState
```

The display of the PIN pad shows the following.

```
Insert Smartcard
```

```
Press OK/Cancel
```

- b. Insert the smartcard into the PIN pad.
- c. Press the **OK** button of the PIN pad.

Example output:

```
current AUTH state: 22000000
user: ADMIN
```

6. If you want to terminate the connection to the remote computer, enter the `exit` command.

5.2.9 Syntax for Authenticating a Command of a CryptoServer Tool/API

This chapter describes the syntax for how to authenticate a command that is performed on a remote computer by a PIN pad that is connected to a local computer. The complete syntax is as follows:

```
<smartcard specifier>[@<port>]@<IP address>[/<password>][#<smartcard PIN>]
```

Complete example:

```
:cs2:cjo:USB0@6070@127.0.0.1/mypwd#123456
```

- **<smartcard specifier>**

Smartcard specifiers, for example, :cs2:cjo:USB0, specify the smartcard type, the PIN pad type and the interface (USB or COM) the PIN pad is connected to. For details, see *Storage and Specification of RSA and ECDSA Keys for Authentication* in the [CryptoServer - csadm Manual \(p. 284\)](#).

- **<port>** (optional)

If a port other than the default port 6070 has been defined in the configuration file, this port must be used here. It is the number of the port that has to be open at the PIN pad daemon on the local computer. In figure *PIN pad access to a remote CryptoServer without a blocking firewall* in [Scenario 3: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN \(p. 32\)](#), this port is port 6070. In [Scenario 4: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN With Blocking Firewall \(p. 33\)](#), this port has to be open as well at the SSH daemon's side in direction to the CryptoServer tool/API. In figure *PIN pad access to a remote CryptoServer with a blocking firewall* in [Scenario 4: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN With Blocking Firewall \(p. 33\)](#), this port is port 6070.

- **<IP address>**

If you apply [Scenario 3: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN Without Blocking Firewall \(p. 32\)](#), use the IP address of the local computer. If you apply [Scenario 4: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN With Blocking Firewall \(p. 33\)](#), use 127.0.0.1 or localhost instead.

- **<password>** (optional)

This is the password for authenticating the exchange of the session key for secure messaging. This password has been defined in the password file. For details about the password file and the password, see [Configuring the PIN Pad Daemon \(PPD\) \(p. 158\)](#).



The password you enter here is shown in plaintext in the command-line. There is no option to hide it as it can be done for example in the csadm `LogonSign` or the csadm `LogonPass` command.



The password must not be confused with the smartcard PIN. The password is always entered in the command-line (preceded by a /) whereas the smartcard PIN is typically entered at the PIN pad but it can be entered in the command-line (preceded by a #) as an alternative. The password is used for authenticating the exchange of the session key for secure messaging, and the PIN is used for authenticating the command of a CryptoServer tool/API, for example, csadm.

- `<smartcard PIN>` (optional)

If you do not want to enter the PIN at the PIN pad, specify it here.



The PIN you enter here is shown in plaintext in your command-line. There is no option to hide it as it can be done for example for a password in the csadm `LogonSign` or the csadm `LogonPass` command.

Use `<smartcard PIN>` only, if the CryptoServer tool/API expects a PIN. Otherwise, an error message is shown.

Example for a command producing an error:

```
csadm GetCardInfo=:cs2:cjo:USB0@127.0.0.1#123456
```

The following steps show two examples.

1. As a first example, the csadm GetCardInfo command is used.
 - a. Perform a csadm command according to the following examples:
 - [Scenario 3: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN Without Blocking Firewall \(p. 32\)](#):

```
csadm GetCardInfo=:cs2:cjo:USB0@1.1.1.1/mypwd
```
 - [Scenario 4: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN With Blocking Firewall \(p. 33\)](#):

```
csadm GetCardInfo=:cs2:cjo:USB0@127.0.0.1/mypwd
```

Consider one special aspect of the csadm `GetCardInfo` command. This command only needs an installed csadm on the remote computer but it does not need any contact to a CryptoServer.

The display of the PIN pad shows the following.

```
Insert Smartcard
Press OK/Cancel
```

b. Insert the smartcard into the PIN pad.

c. Press the **OK** button of the PIN pad.

Example output in the command-line:

```
RSA-Key:      Utimaco IS GmbH / Init-Dev-1-Key
ECC-Key:      EcKey
RSA-Backup:   KeyInfo02 #1
ECC-Backup:   EcKey #1
MBK:
```

2. The second example retrieves the authentication status (permission mask) of the default administration user, ADMIN. His user authentication key has been provided on every smartcard delivered by Utimaco IS GmbH.

a. Perform a csadm `LogonSign` command according to the following examples:

- [Scenario 3: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN Without Blocking Firewall \(p. 32\):](#)

```
csadm Dev=PCI:0 LogonSign=ADMIN,:cs2:cjo:USB0@1.1.1.1
GetAuthState
```

- [Scenario 4: Remote CryptoServer Tools/API, Remote CryptoServer PCIe or LAN With Blocking Firewall \(p. 33\):](#)

```
csadm Dev=PCI:0 LogonSign=ADMIN,:cs2:cjo:USB0@127.0.0.1
GetAuthState
```

The display of the PIN pad shows the following.

```
Insert Smartcard
Press OK/Cancel
```

- Insert the smartcard into the PIN pad.
- Press the **OK** button of the PIN pad.

Example output in the command-line:

```
current AUTH state: 22000000
user: ADMIN
```

5.3 Load Balancing Mechanism

Currently, the load balancing mechanism, see [Load Balancing \(p. 71\)](#), is available only for the PKCS#11 API and for the C++ interface of the CXI API. The failover mechanism is provided for the PKCS#11 API and for both C++ and Java interfaces of the CXI API.

Before you start configuring and using the load balancing and failover features make sure that the following requirements are fulfilled for the CryptoServer devices which should belong to the same cluster.

5.3.1 Mandatory Preconditions

- Make sure that all CryptoServer are accessible for the client computer to the same network.
- You have loaded exactly the same firmware modules (versions and functionality in case of self-developed firmware modules) into all devices.
- You have loaded the same MBK into all CryptoServer in the cluster.
For details, see [Master Backup Key Management \(p. 98\)](#). In case you prefer using the command-line administration tool csadm, see the *CryptoServer – csadm Manual*.
- You have synchronized the user databases of all CryptoServer in the cluster, i. e. the same user account(s) and user credentials (password, keyfile, attributes) required for authentication/Secure Messaging are available on every CryptoServer LAN in the cluster.
For details information on how to create users, backup and restore a user database with CAT see 5.7, „User Management“. In case you prefer using the command-line administration tool csadm, see the *CryptoServer – csadm Manual*.



The usage of keys stored on smartcards for user authentication on CryptoServer devices organized in a cluster is not supported.



Consider to synchronize user accounts between all CryptoServer in a cluster after creation of new users, deletion of existing user accounts as well as any changes on existing user accounts on a CryptoServer in the cluster.

5.3.2 Optional Preconditions

- The time in all CryptoServer belonging to the CryptoServer cluster is synchronized by using NTP.
- The audit log configuration file is synchronized between all devices in the cluster.

5.3.3 Configuration Settings for the Use with CXI

- For detailed information on how to configure failover and load balancing for the C++ interface of the CXI API, read the HTML documentation provided on the SecurityServer product CD in `...\Documentation\Crypto_APIs\CXI`.
- For detailed information on how to configure failover for the Java interface of the CXI API, read the HTML documentation provided on the SecurityServer product CD in `...\Documentation\Crypto_APIs\CXI_Java`.

5.3.4 Configuration Settings for the Use with PKCS#11

Before you can use load balancing and failover in your PKCS#11 environment some settings in the configuration file `cs_pkcs11_R3.cfg` are required. For details, see the *PKCS#11 R3 Developer Guide* provided on the SecurityServer product CD in the `...\Documentation\Crypto_APIs\PKCS11_R3\` directory.

5.4 Master Backup Key Rollover

This section describes how to replace an existing Master Backup Key (MBK) by a new one.

Device system administrators can generate, import and use a new MBK to protect external key storage as well as for creating secure backup copies of secret data and private keys stored on the device from unauthorized access.

We call the process of replacing an existing MBK "Master Backup Key Rollover". While many of the commands needed for this process are performed with `csadm`, some mandatory steps must be performed by using other tools, like `p11tool2`, `P11CAT` or `cngtool`.

In the following descriptions, we assume that the PKCS#11 API or CSP/CNG API and the CryptoServer are correctly configured and operational.

For details about how to configure CryptoServer in PKCS#11, see [CryptoServer PKCS#11 - P11CAT - Manual \(p. 284\)](#), and about how to configure CryptoServer CSP/CNG in [CryptoServer - CryptoServer CSP and CryptoServer CNG Key Storage Provider 1.x and 2.x \(p. 284\)](#) provided in the SecurityServer product bundle (\Documentation\Administration Guides).



If you have generated backup files using the following commands/actions before the MBK rollover, these backup files are protected by the old MBK. These backup files can be restored after the MBK rollover.

- `p11tool2 BackupInternalKeys`
- P11CAT > **Backup/Restore > Backup/Restore Keys > Backup Internal Keys**
- `p11tool2 BackupExternalKeys`
- P11CAT > **Backup/Restore > Backup/Restore Keys > Backup External Keys**
- `p11tool2 BackupConfig`
- P11CAT > **Backup/Restore > Backup/Restore Config > Backup Slot Configuration Object**
- `cxitool BackupKey` (Even if this command has used a Tenant Backup Key (TBK) instead of an MBK.)
- `cngtool BackupKey`

Make sure that you know which MBK(s) had been used to generate these backup files (`csadm MBKListKeys ; MBK slot 3`) and that these MBKs are still available. Otherwise, these backup files are inaccessible after an MBK rollover. It is not possible to retrieve the MBK by which a backup file has been generated from this backup file.



A backup file generated by the following `csadm` commands is encrypted by an MBK but it is not covered by the MBK rollover. After an MBK rollover, this backup file cannot be restored and is inaccessible as long as the corresponding MBK is not imported into the MBK slot 3 on the CryptoServer.

- `csadm BackupDatabase`
- `csadm BackupUser`

A backup file generated by the following command is protected by other means than an MBK. An MBK rollover is irrelevant for this backup file. This backup file can be restored after an MBK rollover.

- csadm BackupKey



An HSM authentication key (`authkey.db` file) can neither be backed up nor restored. It is accessible after an MBK rollover.



MBK rollover for PKCS#11 is only supported for CryptoServer V4.30 and later and MBK rollover for CNG is only supported for CryptoServer V4.40 and later.



For security reasons, do not use the CryptoServer Simulator for the purpose of MBK rollover

5.4.1 MBK Rollover Preparations

Perform the following actions to prepare the MBK rollover. The execution order is irrelevant.

- Ensure that the device you are going to use for the MBK rollover procedure is not currently used by any other application.
- Ensure that the SecurityServer product CD delivered by Utimaco is installed on the computer you are going to use for the administration of the device. This guarantees that the required administration tools (csadm, CAT, p11tool2, cxitool or cngtool) are available on the computer.
- If smartcards are used to load or save MBK key shares from/to smartcards or if smartcards are used for command execution:
Ensure that the USB PIN pad delivered by Utimaco is connected to a USB port of the computer csadm or CAT is installed on.
- Have the smartcards with all necessary key shares of the old MBK at hand.

- Keep the number of smartcards *n* required for storing all shares of the new MBK at hand, together with a second set of *n* smartcards for the backup of the new MBK. The smartcard holders also should be present.
- Verify whether you use an internal or external keystore.
 - For a PKCS#11 keystore, the following applies:
If the `KeysExternal` parameter in the `cs_pkcs11_r3.cfg` configuration file has been set to true, an external keystore is used. As a consequence, the `KeyStorageType` parameter and the `KeyStorageConfig` parameter in the same file must have been set as well. Verify this by following the instructions in *Editing the cs_pkcs11_R3.cfg Configuration File* in *CryptoServer – PKCS#11 P11CAT Manual*.
If the `KeysExternal` parameter is set to false, an internal keystore is used.
 - For a CNG keystore, the following applies:
If the `KeysExternal` parameter in the `cs_cng.cfg` configuration file has been set to true, an external keystore is used. As a consequence, the `KeyStore` parameter in the same file must have been set as well. For details, see *Configuration File Options* in *CryptoServer - CryptoServer CSP and CNG Key Storage Provider*. If the `KeysExternal` parameter is set to false, an internal keystore is used.
- A user with at least permission 2 in the user group 6 (02000000) should be available for authenticating the generation of the new MBK. Alternatively, if authentication according to the two-person-rule is used, at least two users, each with permission 1 in the user group 6 (01000000) should be available.
- A user with key manager permissions should be available. For a PKCS#11 keystore, the following applies: The key manager permission mask is configured by the `CKA_CFG_AUTH_KEYM_MASK` parameter. Verify this by following the instructions in *GetGlobalConfig* or *GetSlotConfig* in *CryptoServer – PKCS#11 p11tool2 Reference Manual*.
- A user with at least permission 2 in the user groups 6 and 7 (22000000) should be available for authenticating the backup procedure of the active MBK. Alternatively, if authentication according to the two-person-rule is used, at least two users, each with permission 1 in the user groups 6 and 7 (11000000) should be available.
- If you want to perform an MBK rollover for the CSP/CNG API, we highly recommend to back up the keys by one of the following methods:
- Copying/pasting the external keystore files
They are in a directory (`C:\ProgramData\Utimaco\CNG\keys` (default)) that is defined in the CNG config file (`C:\ProgramData\Utimaco\CNG\cs_cng.cfg` (default)). For details,

see *CSP/CNG Configuration File* in *CryptoServer - CryptoServer CSP and CNG Key Storage Provider*.

- Performing the `cngtool BackupKey`. For details, see *BackupKey* in *CryptoServer - CryptoServer CSP and CNG Key Storage Provider*.
- If you are using a device cluster, the same MBK should be available on all devices in the cluster. If you are using a device cluster, ensure that the same active MBK has been imported on all device devices used to generate and store cryptographic keys into the (one and the same) external key storage.
- If you are using a device cluster, make sure that you have added all devices in the cluster into the CAT's list of configured Devices.

5.4.2 MBK Rollover of an Internal Keystore



This section only applies to an internal PKCS#11 keystore.

1. Follow the warnings in section [Master Backup Key Rollover \(p. 192\)](#).
2. Follow the instructions in section [MBK Rollover Preparations \(p. 194\)](#).
3. Back up all cryptographic keys of a PKCS#11 slot in the internal keystore by following either the instructions in *BackupInternalKeys* in [CryptoServer - PKCS#11 p11tool2 - Reference Manual \(p. 284\)](#) or in *Creating a Backup of all Keys in a Slot* in [CryptoServer PKCS#11 - P11CAT Manual \(p. 284\)](#).

Example input:

```
p11tool2 Slot=0 Login=KeyMgr,123456
BackupInternalKeys=internal_keys_p11slot0.bak
```

Example output:

```
2 internal key(s) backed up
```

4. Repeat step 3 for all other PKCS#11 slots.

- Verify which MBK is stored on the device.

Using csadm: Perform the `csadm MBKListKeys` command.

Using CAT: **Manage MBK > Info > MBKs stored in CryptoServer**, see *Retrieving MBK Information* in the [CryptoServer - CAT Manual \(p. 284\)](#).

A current (old) AES MBK is stored in MBK slot 3. MBK slots numbers must not be confused with PKCS#11 slot numbers.

Example output for `csadm MBKListKeys`:

slot	name	len	algo	type	k	generation	date	key	check	value
3	MbkAes30	32	AES	XOR	2	2019/05/09	08:57:24	D0779FB705960D8A:6228784FDFB0DF27		



Ensure that the value shown in the algo column is AES. Otherwise, do not continue the MBK rollover.

- Verify that key shares of this old MBK are stored, for example, on the smartcards.

Using csadm: Perform the `csadm MBKCardInfo` command.

Using CAT: **Manage MBK > Info MBKs stored in CryptoServer > Card Info**, see *Retrieving MBK Information* in the [CryptoServer - CAT Manual \(p. 284\)](#).

Example output for `csadm MBKCardInfo`:

REC	TYPE	ALG	LEN	NAME	TIME	GEN	K	ID	HASH
15	XOR	AES	256	MbkAes30	09.05.2019	08:57:24	0	0	D0779FB705960D8A

In the example, an XOR type is shown. That means that only two key shares are needed.

Example output for the second smartcard:

REC	TYPE	ALG	LEN	NAME	TIME	GEN	K	ID	HASH
15	XOR	AES	256	MbkAes30	09.05.2019	08:57:24	0	0	6228784FDFB0DF27

The key shares of one MBK are usually stored in the same record number on the smartcards (15 in the example). 15 is the default value for AES MBKs.

Use these smartcards to import the MBK in the next step.

7. Import the old MBK to a MBK slot number greater than 3 into the device, for example by using the MBK shares on the smartcards.

Using csadm: Perform the `csadm MBKImportKey` command

Using CAT: Follow the instructions in *Importing an MBK* in the [CryptoServer - CAT Manual](#) (p. 284).

Example for `csadm MBKImportKey`:

```
csadm Dev= PCI:0 LogonPass=God,123456 Key=:cs2:cjo:USB0,15 MBKImportKey=7
```

Follow the instructions on the PIN pad display.

8. Verify that the old MBK now is available in two MBK slots on the device.

Using csadm: Perform the `csadm MBKListKeys` command.

Using CAT: **Manage MBK > Info > MBKs stored in CryptoServer**, see *Retrieving MBK Information* in the [CryptoServer - CAT Manual](#) (p. 284).

Example output for `csadm MBKListKeys`:

slot	name	len	algo	type	k	generation	date	key	check	value
3	MbkAes30	32	AES	XOR	2	2019/05/09	08:57:24	D0779FB705960D8A:	6228784FDFB0DF27	
7	MbkAes30	32	AES	XOR	2	2019/05/09	08:57:24	D0779FB705960D8A:	6228784FDFB0DF27	

9. If no new MBK is available yet, generate a new MBK for the device.

Using csadm: Perform the `csadm MBKGenerateKey` command. Follow the instructions on the PIN pad display to authenticate the `csadm MBKGenerateKey` command, and then to generate the MBK.

Using CAT: Follow the instructions in *Generating MBK Information* in the [CryptoServer - CAT Manual](#) (p. 284).

For the following example you have to keep two smartcards at hand for the MBK generation as well as the smartcards of both users who have to authenticate the command.

Example for `csadm MBKGenerateKey`:

```
csadm Dev=PCI:0 LogonSign=Admin1,:cs2:cjo:USB0
LogonSign=Admin3,:cs2:cjo:USB0
```

```
Key=:cs2:cjo:USB0,1,:cs2:cjo:USB0,2,:cs2:cjo:USB0,3
MBKGenerateKey=AES,32,2,2,MbkAes29
```

10. If not already done yet, change the PIN of all smartcards on which a share of the MBK is stored.

Using csadm: Perform the `csadm MBKPINChange` command.

Using CAT: Follow the instructions in *Changing the PIN for an MBK Smartcard* in the [CryptoServer - CAT Manual \(p. 284\)](#).

Example for `csadm MBKPIN Change` :

The two generated MBK shares are stored on two smartcards, so the following command must be repeated for both smartcards.

```
csadm MBKPINChange=:cs2:cjo:USB0
```

11. Follow the instructions on the PIN pad display.
12. Import the new MBK into the device. It is important to import it to MBK slot 3.

Using csadm: Perform the `csadm MBKImportKey` command.

Using CAT: Follow the instructions in *Importing an MBK* in the [CryptoServer - CAT Manual \(p. 284\)](#).

Example for `csadm MBKImportKey` :

```
csadm Dev=PCI:0 LogonSign=Admin1,:cs2:cjo:USB0
LogonSign=Admin3,:cs2:cjo:USB0 Key=:cs2:cjo:USB0,14,:cs2:cjo:USB0,14
MBKImportKey=3
```

13. Execute the `csadm MBKListKeys` command to ensure that the MBK has been successfully imported into the device.

Using csadm: Perform the `csadm MBKListKeys` command.

Using CAT: **Manage MBK > Info > MBKs stored in CryptoServer**, see *Retrieving MBK Information* in the [CryptoServer - CAT Manual \(p. 284\)](#).

Example for `csadm MBKListKeys` :

```
csadm Dev=PCI:0 LogonSign=ADMIN,:cs2:cjo:USB0 MBKListKeys
Example output:
slot name      len algo type   k  generation date      key check value
-----
3   MbkAes29 32  AES  XOR    2  2019/05/09 09:04:11
2722BDA0D7D6E821:8F21825F743C3A49
7   MbkAes30 32  AES  XOR    2  2019/05/09 08:57:24
D0779FB705960D8A:6228784FDFB0DF27
```



The backup file(s) created in step 3 and 4 can only be used to restore the internal PKCS#11 keystore if the old MBK is still present on the device (MBK slot 7 in the example). Do not overwrite the MBK in this MBK slot by importing an MBK into this slot unless you have a backup for this MBK, for example, MBK shares on smartcards.

14. You can verify the result of the MBK rollover by performing the following substeps.
- Delete a cryptographic key in the internal keystore that you do not need anymore for this key.

Using p11tool2: Perform the `p11tool2 DeleteObject` command, see *DeleteObject* in the [CryptoServer - PKCS#11 p11tool2 - Reference Manual](#) (p. 284).

Using PKCS#11 CAT: Follow the instructions in *Deleting Objects from the CryptoServer* in the [CryptoServer PKCS#11 - P11CAT Manual](#) (p. 284).

Example for `p11tool2 DeleteObject`:

```
p11tool2 Slot=0 LoginUser=123456 Label="RSA Public Key" Id="P12"
DeleteObject
```

Example output:

```
1 Objects deleted
```

- Note down the slot number of the PKCS#11 slot the cryptographic key has been deleted from.
- Restore the cryptographic keys of the corresponding slot in the internal PKCS#11 keystore. Use the PKCS#11 slot number and the backup file that have been specified in step 3 and 4.

Using p11tool2: Perform the `p11tool2 RestoreInternalKeys` command, see *RestoreInternalKeys* in the [CryptoServer - PKCS#11 p11tool2 - Reference Manual](#) (p. 284).

Using PKCS#11 CAT: Follow the instructions in *Restoring a Backup of All Keys in a Slot* in [CryptoServer PKCS#11 - P11CAT Manual](#) (p. 284).

Example for `p11tool2 RestoreInternalKeys`:


```
p11tool2 Slot=0 Login=KeyMgr,123456  
RestoreInternalKeys=internal_keys_p11slot0.bak
```

Example output:

```
2 internal keys restored to internal key store
```

There is no need to specify the MBK slot number of the old MBK (7 in the example) in the command but this MBK slot number is not shown in the output either. The device automatically finds the correct MBK slot number.

- d. Verify whether the cryptographic key that you have deleted in step 14a) has been restored.

Using p11tool2: Follow the instructions in *ListObjects* in the [CryptoServer - PKCS#11 p11tool2 - Reference Manual](#) (p. 284).

Using PKCS#11 CAT: Follow the first steps in *Generating a Key* (only up to including the step *Click Object Management*) in [CryptoServer PKCS#11 - P11CAT Manual](#) (p. 284) for the corresponding PKCS#11 slot..



The MBK rollover has been completed successfully.

5.4.3 MBK Rollover of an External Keystore

Proceed as follows to perform an MBK rollover.

1. Follow the warnings in section [Master Backup Key Rollover](#) (p. 192).
2. Follow the instructions in section [MBK Rollover Preparations](#) (p. 194).
3. For a CNG keystore, the following applies: If no cryptographic key is available in the CNG keystore yet, perform a command according to the following example to create one. Later this key is backed up, deleted and restored to verify the MBK rollover.

```
cngtool name=Testkey1 createkey=RSA,512
```

Output:

```

-----
Provider : Utimaco CryptoServer Key Storage Provider
Device : PCI:0
Group : CNG
Mode : External Key Storage
-----

```

4. Verify which MBK is stored on the device. A current (old) AES MBK is stored in MBK slot 3. MBK slots numbers must not be confused with PKCS#11 slot numbers.

Using csadm: Perform the `csadm MBKListKeys` command.

Using CAT: **Manage MBK > Info > MBKs stored in CryptoServer**, see *Retrieving MBK Information* in the CryptoServer - CAT Manual.

Example output for `csadm MBKListKeys`:

```

lot name len algo type k generation date key check value
-----
3 MbKaes30 32 AES XOR 2 2019/05/09 08:57:24
D0779FB705960D8A:6228784FDFB0DF27

```

5. Verify that key shares of this old MBK are stored, for example, on the smartcards.

Using csadm: Perform the `csadm MBKCardInfo` command.

Using CAT: **Manage MBK > Info MBKs stored in CryptoServer > Card Info**, see *Retrieving MBK Information* in the CryptoServer - CAT Manual.

Example output for `csadm MBKCardInfo`:

```

REC TYPE ALG LEN NAME TIMEGEN K ID HASH
-----
15 XOR AES 256 MbKaes30 09.05.2019 08:57:24 0 0 D0779FB705960D8A

```

In the example, an XOR type is shown. That means that only two key shares are needed.

Example output for the second smartcard:

```

REC TYPE ALG LEN NAME TIMEGEN K ID HASH
-----
15 XOR AES 256 MbKaes30 09.05.2019 08:57:24 0 0 6228784FDFB0DF27

```

The key shares of one MBK are usually stored in the same record number on the smartcards (15 in the example). 15 is the default value for AES MBKs. Use these smartcards to import the MBK in the next step.

6. Import the old MBK to a MBK slot number greater than 3 into the device, for example by using the MBK shares on the smartcards.

Using csadm: Perform the `csadm MBKImportKey` command

Using CAT: Follow the instructions in *Importing an MBK* in the *CryptoServer - CAT Manual*.

Example for `csadm MBKImportKey`:

```
csadm Dev= PCI:0 LogonPass=God,123456 Key=:cs2:cjo:USB0,15 MBKImportKey=7
```

Follow the instructions on the PIN pad display.

7. Verify that the old MBK now is available in two MBK slots on the device.

Using csadm: Perform the `csadm MBKListKeys` command.

Using CAT: **Manage MBK > Info > MBKs stored in CryptoServer**, see *Retrieving MBK Information* in the *CryptoServer - CAT Manual*.

Example output for `csadm MBKListKeys`:

```
slot name len algo type k generation date key check value
-----
-----
3 MbkAes30 32 AES XOR 2 2019/05/09 08:57:24
D0779FB705960D8A:6228784FDFB0DF27
7 MbkAes30 32 AES XOR 2 2019/05/09 08:57:24
D0779FB705960D8A:6228784FDFB0DF27
```

8. If no new MBK is available yet, generate a new MBK for the device.

Using csadm: Perform the `csadm MBKGenerateKey` command. Follow the instructions on the PIN pad display to authenticate the `csadm MBKGenerateKey` command, and then to generate the MBK.

Using CAT: Follow the instructions in *Generating MBK Information* in the *CryptoServer - CAT Manual*.

For the following example you have to keep two smartcards at hand for the MBK generation as well as the smartcards of both users who have to authenticate the command.

Example for `csadm MBKGenerateKey`:

```
csadm Dev=PCI:0 LogonSign=Admin1,:cs2:cjo:USB0
```

```
LogonSign=Admin3,:cs2:cjo:USB0
Key=:cs2:cjo:USB0,1,:cs2:cjo:USB0,2,:cs2:cjo:USB0,3
MBKGenerateKey=AES,32,2,2,MbkAes29
```

9. If not already done yet, change the PIN of all smartcards whereon a share of the MBK is stored.

Using csadm: Perform the `csadm MBKPINChange` command.

Using CAT: Follow the instructions in *Changing the PIN for an MBK Smartcard* in the CryptoServer - CAT Manual.

Example for `csadm MBKPIN Change`:

The two generated MBK shares are stored on two smartcards, so the following command must be repeated for both smartcards.

```
csadm MBKPINChange=:cs2:cjo:USB0
```

10. Follow the instructions on the PIN pad display.

11. Import the new MBK into the device. It is important to import it to MBK slot 3.

Using csadm: Perform the `csadm MBKImportKey` command.

Using CAT: Follow the instructions in *Importing an MBK* in the CryptoServer - CAT Manual.

Example for csadm `MBKImportKey`:

```
csadm Dev=PCI:0 LogonSign=Admin1,:cs2:cjo:USB0
LogonSign=Admin3,:cs2:cjo:USB0 Key=:cs2:cjo:USB0,14,:cs2:cjo:USB0,14
MBKImportKey=3
```

12. Execute the `csadm MBKListKeys` command to ensure that the MBK has been successfully imported into the device.

Using csadm: Perform the `csadm MBKListKeys` command.

Using CAT: **Manage MBK > Info > MBKs stored in CryptoServer**, see *Retrieving MBK Information* in the CryptoServer - CAT Manual.

Example for csadm `MBKListKeys`:

```
csadm Dev=PCI:0 LogonSign=ADMIN,:cs2:cjo:USB0 MBKListKeys
```

Example output:

```
slot name len algo type k generation date key check value
-----
```

```
3 MbkAes29 32 AES XOR 2 2019/05/09 09:04:11
2722BDA0D7D6E821:8F21825F743C3A49
7 MbkAes30 32 AES XOR 2 2019/05/09 08:57:24
D0779FB705960D8A:6228784FDFB0DF27
```

13. For a PKCS#11 keystore, continue with step 14. For a CNG keystore, continue with step 17.

14. Only for an External PKCS#11 Keystore

Perform the `p11tool2 RecryptExternalKeys` command. This command first backs up all cryptographic keys of the specified PKCS#11 slot of the external PKCS#11 keystore and stores them in the specified backup file. Then it encrypts the cryptographic keys in this PKCS#11 slot of the external PKCS#11 keystore with the new MBKs, see *RecryptExternalKeys* in the [CryptoServer - PKCS#11 p11tool2 - Reference Manual](#). (p. 284) This step cannot be performed by using P11CAT.

Example:

```
p11tool2 Slot=0 Login=USR_0000,123456 RecryptExternalKeys=p11.pks.bak
```

Example output:

```
2 external key(s) backed up
2 external key(s) recrypted
```

15. Repeat step 14 for all other PKCS#11 slots of the external PKCS#11 keystore containing cryptographic keys.

The created backup file(s) created can only be used to restore the external PKCS#11 keystore if the old MBK is still present on the device (MBK slot 7 in the example). Do not overwrite the MBK in this MBK slot by importing an MBK into this slot unless you have a backup for this MBK, for example, MBK shares on smartcards.

16. You can verify the result of the MBK rollover by performing the following substeps.
 - a. Delete a cryptographic key in the internal keystore that you do not need anymore for this key.
 Using p11tool2: Perform the `p11tool2 DeleteObject` command, see *DeleteObject* in the *CryptoServer - PKCS#11 p11tool2 - Reference Manual*.
 Using PKCS#11 CAT: Follow the instructions in *Deleting Objects from the*

CryptoServer in the *CryptoServer PKCS#11 - P11CAT Manual*.

Example for `p11tool2 DeleteObject`:

```
p11tool2 Slot=0 LoginUser=123456 Label="RSA Public Key" Id="P12"  
DeleteObject
```

Example output:

```
1 Objects deleted
```

- b. Note down the slot number of the PKCS#11 slot the cryptographic key has been deleted from.
- c. Restore the cryptographic keys of the corresponding slot in the internal PKCS#11 keystore. Use the PKCS#11 slot number and the backup file that have been specified in step 3 and 4.

Using `p11tool2`: Perform the `p11tool2 RestoreInternalKeys` command, see *RestoreInternalKeys* in the *CryptoServer - PKCS#11 p11tool2 - Reference Manual*.

Using PKCS#11 CAT: Follow the instructions in *Restoring a Backup of All Keys in a Slot* in *CryptoServer PKCS#11 - P11CAT Manual*.

Example for `p11tool2 RestoreInternalKeys`:

```
p11tool2 Slot=0 Login=KeyMgr,123456  
RestoreExternalKeys=external_keys_p11slot0.bak
```

Example output:

```
2 external keys restored to external key store
```

There is no need to specify the MBK slot number of the old MBK (7 in the example) in the `csadm` command but this MBK slot number is not shown in the output either. The *CryptoServer* automatically finds the correct MBK slot number.

- d. Verify whether the cryptographic key that you have deleted in step 16a) has been restored.

Using `p11tool2`: Follow the instructions in *ListObjects* in the *CryptoServer - PKCS#11 p11tool2 - Reference Manual*.

Using PKCS#11 CAT: Follow the first steps in *Generating a Key* (only up to including

the step *Click Object Management*) in *CryptoServer PKCS#11 - P11CAT Manual* for the corresponding PKCS#11 slot..

17. Only for an External CNG Keystore

Back up the cryptographic key you have created in step 3. This key is backed up using the new MBK in MBK slot 3.

```
cngtool Name=Testkey1 backupkey
Output:
-----

Provider : Utimaco CryptoServer Key Storage Provider
Device   : PCI:0
Group    : CNG
Mode     : External Key Storage

-----

I: Successfully backed up key to: Testkey1_0.kbk
```

The backup file created in this step can only be used to restore the external CNG keystore if the new MBK in MBK slot 3 is still present on the CryptoServer. Do not overwrite the MBK in this MBK slot by importing an MBK into this slot unless you have a backup for this MBK, for example, MBK shares on smartcards.

18. Repeat the last step for all cryptographic keys in the keystore. Each key must be backed up individually.
19. Perform the `cngtool RecryptExternalKeys` command. This command encrypts the cryptographic keys in the external CNG keystore with the new MBK in MBK slot 3. This command works only with the Utimaco key storage provider. Only those keys are encrypted that the current user has Windows access rights for, see *RecryptExternalKeys* in [CryptoServer - CryptoServer CSP and CryptoServer CNG Key Storage Provider 1.x and 2.x \(p. 284\)](#) for details.

Example:

```
Example:
cngtool RecryptExternalKeys
Example output:
-----

Provider : Utimaco CryptoServer Key Storage Provider
Device   : PCI:0
Group    : CNG
Mode     : External Key Storage
```

```
-----
I: Recrypted 1 key(s), 0 failed.
```

20. Now, the MBK rollover for the external CNG keystore is finished.

You can verify the result of the MBK rollover by performing the following substeps.

- a. Delete the cryptographic key you have created in step 3.

Example:

```
cngtool name=Testkey1 DeleteKey
```

Example output:

```
-----

Provider : Utimaco CryptoServer Key Storage Provider
Device   : PCI:0
Group    : CNG
Mode     : External Key Storage

-----
```

```
I: 1 keys deleted
```

- b. Restore the cryptographic key you have created in step 3.

Example:

```
cngtool restorekey=Testkey1_0.kbk
```

Example output:

```
-----

Provider      : Utimaco CryptoServer Key Storage Provider
Device        : 3001@127.0.0.1
Group         : CNG
Mode          : External Key Storage

-----
```

```
I: Successfully restored key from: Testkey1_0.kbk
```

- c. Verify whether the cryptographic key that you have deleted has been restored.

Example:


```
cngtool listkeys
```

Example output:

```
-----  
Provider : Utimaco CryptoServer Key Storage Provider  
Device : PCI:0  
Group : CNG  
Mode : External Key Storage  
-----
```

```
Index AlgId Size Group Name Spec  
-----  
-----  
1 RSA 512 CNG Testkey1 0
```

Now, the verification of the MBK rollover result for an external CNG keystore is finished.



The MBK rollover of an external keystore has been completed successfully.

5.5 Configurable Role-Based Access

Role-based access provides the possibility to increase the necessary authentication level for selected device functions, and thus, to even further restrict the access to the system to users with customized roles/permissions.

The CryptoServer maximal possible authentication status that can be reached for selected functions in the different user groups has been increased to 15 (0xF).

The increased permission level enabling specific users to execute specific functions can be individually configured by the customer for the device external functions with help of the signed configuration file `cmds.scf`.

In order to do so, the configuration file `cmds_sample.cfg` has to contain a section `[Permissions]` specifying the increased permissions required for specific functions in the defined firmware modules. The syntax is as follows:

```
[Permissions]
```

```

<FC> = <SFC1>:<permissions>,<SFC2>:<permissions>,...,<SFCn>:<permissions>
or for better readability
[Permissions]
<FC> = <SFC1>:<permissions>,\
    <SFC2>:<permissions>,\
    ..., \
    <SFCn>:<permissions>

```

FC is the function code (ID) of the firmware module which is exactly three hex digits with leading 0x, e.g., 0x012.

SFC is the decimal sub-function code and permissions is a hexadecimal number (maximum 8 bytes without leading 0x, e.g., FF00F000) denoting the permission in each of the eight user groups. The SFCs do not have to be ordered numerically. The permissions defined in the signed configuration file can only be used to extend the permission required, by default, for the execution of the corresponding firmware module function. For security reasons it is not possible to suspend the required default permissions for the specified functions for external interfaces.

The signed configuration file `cmds.scf` is evaluated during startup, while all firmware modules register with their FC and SFCs at the CMDS module. In case there is an entry for a given firmware module, identified by its FC, found in the `cmds.scf` file, the required permissions for each specified SFC are set according to that definition. The list of customized permissions that are enforced after registration, is included in the audit log only if the log event "Startup messages" is activated, as by default. An entry in the Audit Log contains the following information:

```
<date> <time> <FC:0xXXX> <SFC:XX> Configured Permission=<permission>
```

For example

```
16.12.2015 13:20:45 FC:0x068 SFC:17 Configured Permission=6F000000
```

During command execution, the currently reached authentication level for the device function, identified by its FC and SFC, is compared to the required permission in the configuration file. If they are equal or no permission restrictions list exists for this module (FC), the command is executed normally performing the default (unchanged) permissions check for the command.

If the required customized permissions are not reached or in case the check of the default permissions fails, the error message B0830001 permission denied is returned.

See *Creating and Using the Signed Configuration File `cmds.s`* in the *CryptoServer - csadm Manual* cf for step-by-step instructions on how to create/edit, sign, load and activate your signed configuration file `cmds.scf`.

5.6 Interface Hardening by Disabling Selected Functions

The device software offers the possibility to additionally secure the device interfaces by disabling selected device functions. These functions have to be defined in the signed configuration file `cmds.scf`.

For that purpose, the configuration file `cmds_sample.cfg` has to contain the dedicated section `[DisableSFC]` specifying which functions should be disabled. The syntax is as follows:

```
[DisableSFC]
<FC> = <SFC1>,<SFC2>,...,<SFCn>
or for better readability
<FC> = <SFC1>,\
      <SFC2>,\
      ..., \
      <SFCn>
```

FC is the function code (ID) of the firmware module, which is exactly three hex digits with leading 0x, e.g., "0x012".

SFC is the specific decimal sub-function code (ID), specifying a function within a firmware module, which is disabled. The SFCs do not have to be ordered numerically.

The disabled firmware module functions are listed during device startup in the Boot Log in specific entries with the syntax

```
CMDS/DSOM: <FC> - disabled <SFC1> <SFC2> ... <SFCn>.
```

For example, `CMDS/DSOM: 0x083 - disabled 0 1 17` means that in the firmware module CMDS (with function code FC = 0x83) the functions Echo (with SFC = 0), Reverse Echo (with SFC = 1) and Set Maximum Authentication Failures (with SFC = 17) are disabled, and cannot be used by external applications.

If you try to access a disabled device function the following error message is returned by the device:

```
Error B0830061
```

This function is not available in this HSM configuration.

5.7 Configuring HMAC Password Requirements

Custom requirements for HMAC password users can be set in the `[PasswordRequirements]` section of the `cmds.scf` file. All of the parameters described below are optional.

Parameter	Description
MinLength	Minimum length of the password. The value must be a positive integer. If the value is smaller than the default minimum password length (global limit), it will be ignored and the default minimum password length will be used instead. For the default minimum password length, see the CMDS firmware module interface documentation (<code>mdl_CMDS.pdf</code>).
RequireLowerCase	The password requires at least one lower-case letter (a - z). Supported values: <code>true</code> and <code>false</code> . Default value: <code>false</code>
RequireUpperCase	The password requires at least one upper-case letter (A - Z). Supported values: <code>true</code> and <code>false</code> . Default value: <code>false</code>
RequireNumber	The password requires at least one number (0-9). Supported values: <code>true</code> and <code>false</code> . Default value: <code>false</code>
RequireSpecialChars	The password requires at least one special character (<code>`~!@#\$%^&*()-_+= [{ }];: ' ", < . > / ?</code> or a blank). Supported values: <code>true</code> and <code>false</code> . Default value: <code>false</code>
MinNumRestrictions	The value of <code>MinNumRestrictions</code> must be a positive integer or <code>0</code> . The value determines how many of the above parameters that are active (i.e., set to <code>true</code>) have to be met by the password. If the value is <code>0</code> or higher than the amount of the <code>true</code> values, all active parameters have to be met. Default value: <code>0</code>
UsernameAllowed	If set to <code>false</code> , the password must not be the same as the user name. It can contain the user name, though. Supported values: <code>true</code> and <code>false</code> . Default value: <code>true</code>

Example

The following example shows the `[PasswordRequirements]` section of the `cmds.scf` configuration file with configured custom HMAC password requirements.

```
[PasswordRequirements]
MinLength = 10
RequireLowerCase = true
RequireUpperCase = true
RequireNumber = false
RequireSpecialChars = true
MinNumRestrictions = 2
UsernameAllowed = false
```

Examples of `MinNumRestrictions` Configurations

Configuration file	Interpretation
<code>RequireLowerCase = false</code> <code>RequireUpperCase = false</code> <code>RequireNumber = false</code> <code>RequireSpecialChars = false</code> <code>MinNumRestrictions = 0</code>	No restrictions
<code>RequireLowerCase = true</code> <code>RequireUpperCase = false</code> <code>RequireNumber = false</code> <code>RequireSpecialChars = true</code> <code>MinNumRestrictions = 1</code>	Requires at least one lower-case letter (a-z) or one special character.
<code>RequireLowerCase = false</code> <code>RequireUpperCase = false</code> <code>RequireNumber = true</code> <code>RequireSpecialChars = true</code> <code>MinNumRestrictions = 2</code>	Requires at least one number and one special character.

Configuration file	Interpretation
<pre>RequireLowerCase = true RequireUpperCase = true RequireNumber = true RequireSpecialChars = true MinNumRestrictions = 4</pre>	<p>Requires at least</p> <ul style="list-style-type: none"> ▪ one lower-case letter (a-z), ▪ one upper-case letter (A-Z), ▪ one number and ▪ one special character.
<pre>RequireLowerCase = true RequireUpperCase = true RequireNumber = true RequireSpecialChars = true MinNumRestrictions = -1</pre>	<p>Invalid value of <code>MinNumRestrictions</code> because <code>MinNumRestrictions < 0</code>. <code>MinNumRestrictions</code> is interpreted as the number of <code>true</code> values (4), i.e., as if the <code>[PasswordRequirements]</code> section of the <code>cmds.scf</code> file contains the following:</p> <pre>RequireLowerCase = true RequireUpperCase = true RequireNumber = true RequireSpecialChars = true MinNumRestrictions = 4 // the # of trues</pre>
<pre>RequireLowerCase = true RequireUpperCase = false RequireNumber = false RequireSpecialChars = false MinNumRestrictions = 0</pre>	<p>Invalid value of <code>MinNumRestrictions</code> because <code>MinNumRestrictions < 1</code> and there is at least one <code>true</code> value. <code>MinNumRestrictions</code> is interpreted as the number of <code>true</code> values (1), i.e., as if the <code>[PasswordRequirements]</code> section of the <code>cmds.scf</code> file contains the following:</p> <pre>RequireLowerCase = true RequireUpperCase = false RequireNumber = false RequireSpecialChars = false MinNumRestrictions = 1 // the # of trues</pre>
<pre>RequireLowerCase = true RequireUpperCase = false RequireNumber = true RequireSpecialChars = false MinNumRestrictions = 3</pre>	<p>Invalid value of <code>MinNumRestrictions</code> because <code>MinNumRestrictions > the number of true values (here: 2)</code>. <code>MinNumRestrictions</code> is interpreted as the number of <code>true</code> values (2), i.e., as if the <code>[PasswordRequirements]</code> section of the <code>cmds.scf</code> file contains the following:</p> <pre>RequireLowerCase = true RequireUpperCase = false RequireNumber = true RequireSpecialChars = false MinNumRestrictions = 2</pre>

Configuration file	Interpretation
<code>RequireLowerCase = true</code> <code>RequireUpperCase = true</code> <code>RequireNumber = true</code> <code>RequireSpecialChars = false</code> <code>MinNumRestrictions = 6</code>	Invalid value of <code>MinNumRestrictions</code> because <code>MinNumRestrictions</code> > the number of <code>true</code> values (here: 3) and the maximum value (4). <code>MinNumRestrictions</code> is interpreted as the number of <code>true</code> values (3), i.e., as if the <code>[PasswordRequirements]</code> section of the <code>cmds.scf</code> file contains the following: <code>RequireLowerCase = true</code> <code>RequireUpperCase = true</code> <code>RequireNumber = true</code> <code>RequireSpecialChars = false</code> <code>MinNumRestrictions = 3</code>

Once the configuration file is loaded successfully, the new HMAC password requirements apply.

Existing HMAC users in the database can still log in and operate the HSM with no restrictions. HMAC users with passwords that do not comply with the configured password requirements can be restored.

6 CryptoServer Simulator

The CryptoServer Simulator is a software program provided by Utimaco IS GmbH. It simulates the cryptographic and administrative functions of all series CryptoServer. It is available for Windows 7 or later and Linux operating systems, and is delivered within the SecurityServer product bundle, or can be downloaded from the Utimaco web page <https://www.utimaco.com/downloads> after successful registration.

The CryptoServer Simulator is an ideal means for new customers to evaluate and test the CryptoServer without purchasing any hardware. The simulator should only be used for these purposes and not for data protection in real production environments. If the CryptoServer Simulator has been purchased as part of the software development kit, you can use it as well to develop and debug your own firmware modules providing specific cryptographic functions and commands.

6.1 Physical Protection

The CryptoServer Simulator usually runs on a non-protected computer. That means that no physical protection is provided as it is done on a hardware CryptoServer by using a tamper-protecting foil etc.

6.2 Cryptographic Accelerator Chip

The CryptoServer Se-Series Gen2 may be delivered with a cryptographic accelerator chip providing the highest performance for RSA and/or ECC operations. The CryptoServer Simulator does not simulate any accelerator chip.

6.3 CryptoServer Series

The CryptoServer Simulator only simulates CryptoServer Se-Series Gen2. It simulates the following memory sizes:

Memory	Size
IRAM	1 Mbyte
SDRAM	32 Mbyte
Flash	64 Mbyte
NVRAM	1 Mbyte

6.4 Features Overview

The table lists the differences between the CryptoServer Simulator and a hardware CryptoServer, i.e., a PCIe card mounted in a computer or in a CryptoServer LAN.

Item	CryptoServer Simulator	Hardware CryptoServer	
		PCIe card in a computer	CryptoServer LAN
Deployment purpose	Evaluation and test of data protection; firmware development	Data protection	
Physical protection	No	Yes	
Sensor detection	Possible, manually set, no default	Yes (automatically set)	
Cryptographic accelerator chip	No	Possible	
CryptoServer series	CryptoServer Se-Series Gen2	<ul style="list-style-type: none"> ▪ CryptoServer Se-Series Gen2 ▪ CryptoServer CSe-Series 	
Multiple connections	Possible, static, no default	No	Yes
Device address	3001@127.0.0.1	Example: Windows: PCI:0 (mounted in the host computer where CAT is installed) Linux: /dev/cs2.03	IP address of CryptoServer LAN
Firmware Module Files	Windows: .dll files Linux: .so files	.out files	
csadm commands			
ListFirmware	Windows: SDK Linux: SIM	C64 or C64+	
ListFiles	Windows: SDK Linux: SIM	C64 or C64+	
ModuleInfo Module type	Windows: SDK-DLL Linux: SDK-SO	CS2-COFF	
ModuleInfo Target CPU type	Windows: SDK Linux: SIM	C64+ or C64x	

6.5 Deliverables

6.5.1 The Master Key

A Master Key has already been generated and then stored on the CryptoServer.

As none of the options for administering this key are present, even Utimaco cannot know the key value. You can decide whether you want to keep this Master Key as it is, or completely reset the CryptoServer to generate a new Master Key.

6.5.2 ADMIN, the Default Administrator

The default administrator ADMIN has already been set up on the CryptoServer. He has the permission 2 in the user groups 7 and 6 (22000000).

The authentication with electronic signature procedure has already been defined as the authentication procedure for the default administrator ADMIN. The RSA algorithm is used as the cryptographic algorithm here. The public part of the RSA key has already been loaded into the CryptoServer's user database as authentication data.

You will find the private part of the RSA key, which is needed for authentication, in the product bundle. For the CryptoServer Simulator, it is the `ADMIN_SIM.key` keyfile in one of the following directories:

- `\Software\Windows\Administration\key`
- `\Software\Linux\Administration\key`

The `ADMIN_SIM.key` file is available in the (default) `C:\Program Files\Utimaco\SecurityServer\Administration` directory of the CryptoServer installation.

You do not need a password to open the file.

6.5.3 Serial Numbers and Descriptors

Every CryptoServer has a number of serial numbers that identify component parts. Some of these serial numbers will be displayed on the CryptoServer. This status information and additional descriptors are displayed when the CryptoServer's status is queried or called. All serial numbers and descriptors have already been loaded and stored on the device when the CryptoServer is supplied.

6.6 Starting the Simulator with `bl_sim5.exe`

If you start the CryptoServer Simulator by executing the `bl_sim5.exe` file explicitly, you can use the following parameters.

- no parameter
The simulator is started in BOOTLOADER mode with one connection.
- `-h`
The simulator is started with multiple connections, see 2.22.9, "Multiple Connections". If you want to debug the simulator, omit this option.
- `-o`
The SMOS operating system is started. The simulator is in the OPERATIONAL mode. If this option is omitted, the simulator is in the BOOTLOADER mode. In this case, no firmware modules are loaded and no operating system is started. You can load firmware modules (`mtc` files) manually using the `csadm LoadFile` command. You can start the operating system using the `csadm StartOS` command, see the *CryptoServer - csadm Manual*.
- `-d` (Windows only)
If necessary, use the `-d <devices directory>` option to specify the location of the devices directory. This directory contains, for example, the `ALARM.curr` file to simulate a sensor alarm manually, see [Sensor Detection \(p. 225\)](#).
Example: `C:\ProgramData\Utimaco\Simulator\devices`

If the `-d` parameter is not specified, the `devices` directory is expected to reside in `..\devices`.



As a best practice, use a combination of the `-h` option and the `-o` option. On Windows, this can be done by using the `cs_sim.bat` file.

6.7 Multiple Connections

If a CryptoServer PCIe card has been mounted in a computer, it offers only one connection to the network. Only one command sent to the CryptoServer can be processed at a time.

However, in a CryptoServer LAN, there is a small server between the CryptoServer PCIe card and the network. Multiple commands sent to the CryptoServer can be processed at a time.

The CryptoServer Simulator offers both, a single connection and multiple connections depending on the way the simulator has been started. If you start the CryptoServer Simulator by executing the `bl_sim5.exe` file explicitly, use the `-h` option to activate multiple connections, see [Using the CryptoServer Simulator \(p. 220\)](#). This parameter starts a small

server providing multiple connections. If you do not use the `-h` option, you start a simulator for a single connection.

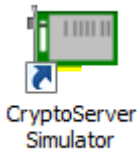
Windows: If you use the `cs_sim.bat` file, the CryptoServer Simulator is started automatically with the `-h` option.

Linux: The `cs_sim.sh` file includes starting the CryptoServer Simulator with the `-h` option.

6.8 Windows: Using the CryptoServer Simulator

The installation of the simulator is described in [Installing the Host Software and CryptoServer Simulator \(p. 131\)](#).

1. Start the CryptoServer Simulator in one of the following ways:
 - a. In Windows click **Start > All Programs > Utimaco > CryptoServer > CryptoServer Simulator**
 - b. Double-click the desktop icon for the CryptoServer Simulator.
The desktop icon is available if you kept the default setting for desktop links during the installation of the host software



- c. Open a command-line in the product CD installation directory for the CryptoServer Simulator (`C:\Program Files\Utimaco\SecurityServer\Simulator\bin`), enter `cs_sim.bat` and press **ENTER**.
Alternatively, type `bl_sim5.exe -h -o` and press **ENTER**.



The `-h` option starts a small server enabling multiple connections (see 2.22.9, "Multiple Connections"). The `-o` option starts the SMOS operating system.

If necessary, use the `-d <devices directory>` option to specify the location of the `devices` directory. This directory contains, for example, the `ALARM.curr` file

to simulate a sensor alarm. For details on the parameters, see [Starting the Simulator with bl_sim5.exe](#) (p. 218).

2. If a **Windows Security Alert** windows opens, click **Allow access**.
This starts the **Utimaco CryptoServer SDK5 – Simulator** command-line application. Do not close this application until you have finished using the simulator.
3. Double-click the **CryptoServer Administration** desktop icon to start the CAT. The CAT main window opens.
4. In the toolbar, click **Devices**.
This opens the **CryptoServer Devices** dialog box.
5. Enter the address `3001@127.0.0.1` in the **New Device** text box.
6. Click **Add to List**.
7. Click **Test** to check whether a connection can be established. The test result is displayed in a separate window.
8. Click **OK** to close this window.
9. Close the **CryptoServer Devices** dialog box .

The successfully established connection is now confirmed in the info field of the CAT main window. Furthermore, information about the date and time as well as the current status of the CryptoServer Simulator is displayed.

6.8.1 Starting Multiple CryptoServer Simulator Instances on Windows

As from SecurityServer 4.01, the simultaneous start and use of multiple CryptoServer Simulator instances is possible. This might be very useful, for example, if you want to configure and simulate your own CryptoServer load balancing cluster for evaluation and test purposes.

1. Start the CryptoServer Simulator, see [Using the CryptoServer Simulator](#) (p. 220).
2. Configure it according to your individual requirements, for example, create specific users, see *Creating a User* in the *CryptoServer - CAT Manual*.
3. Close the CryptoServer Simulator after you have finished the configuration.
4. Open a command-line window.



If you open the path mentioned in the next step in Windows Explorer, enter `cmd` in the address field and press **ENTER**, to open a command-line in the specified path.

5. Change to the CryptoServer product CD installation directory:

```
cd C:\Program Files\Utimaco\SecurityServer\Simulator\bin
```

6. Start, for example, three CryptoServer Simulator instances by typing `cs_multi.bat 3` and pressing **ENTER**.

Three identically configured CryptoServer Simulator instances are started. They have the following device addresses:

- First instance – `3001@localhost`
- Second instance – `3003@localhost`
- Third instance – `3005@localhost`

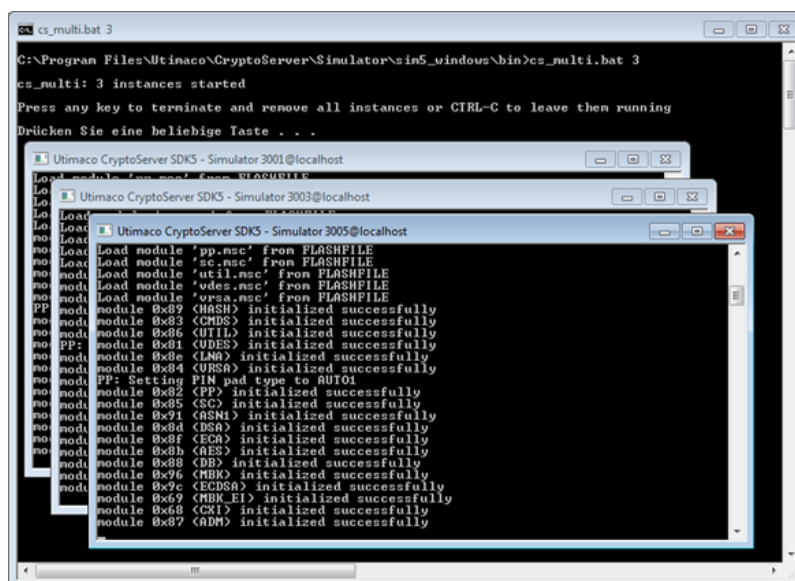


Figure 35 : Example for running three CryptoServer Simulator instances simultaneously (Windows)

To close all CryptoServer Simulator instances, change to the command-line window and press any key. When the instances are closed, all your changes are lost. The next time you start multiple CryptoServer Simulator instances they will have the same configuration as the one you have previously configured in step 2.

To only terminate the Simulator that is running in the currently active command-line window, for example, the instance with the device address 3003@localhost, press **CTRL+C**. The other CryptoServer Simulator instances, remain running and can be used.

6.9 Linux: Using the CryptoServer Simulator

1. Start the CryptoServer Simulator:

```
~/Utimaco/CryptoServer/Simulator/sim5_linux/bin/cs_sim.sh
```



The **-h** option in this sh file starts a small server enabling multiple connections, see [Multiple Connections](#) (p. 219). The **-o** option starts the SMOS operating system.

This starts the **Utimaco CryptoServer SDK5 – Simulator** command-line application. Do not close this application until you have finished using the simulator.

6.9.1 Starting Multiple CryptoServer Simulator Instances on Linux

As from SecurityServer 4.01 the simultaneous start and use of multiple CryptoServer Simulator instances is possible. This might be very useful, for example, if you want to configure and simulate your own CryptoServer load balancing cluster for evaluation and test purposes.

1. Start the CryptoServer Simulator, see [Starting the CryptoServer Simulator](#) (p. 223)
2. Configure it according to your individual requirements, for example, see *Creating a User* in the *CryptoServer - CAT Manual*.
3. Close the CryptoServer Simulator after you have finished the configuration.
4. Open a command-line window.
5. Change to the CryptoServer product CD installation directory: `<product CD>/Software/Linux/Simulator/sim5_linux/bin`
6. Start, for example, three CryptoServer Simulator instances by entering `cs_multi.sh 3` and press **ENTER**.

Three identically configured CryptoServer Simulator instances are started. They have the following device addresses:

- First instance – 3001@localhost
- Second instance – 3003@localhost
- Third instance – 3005@localhost

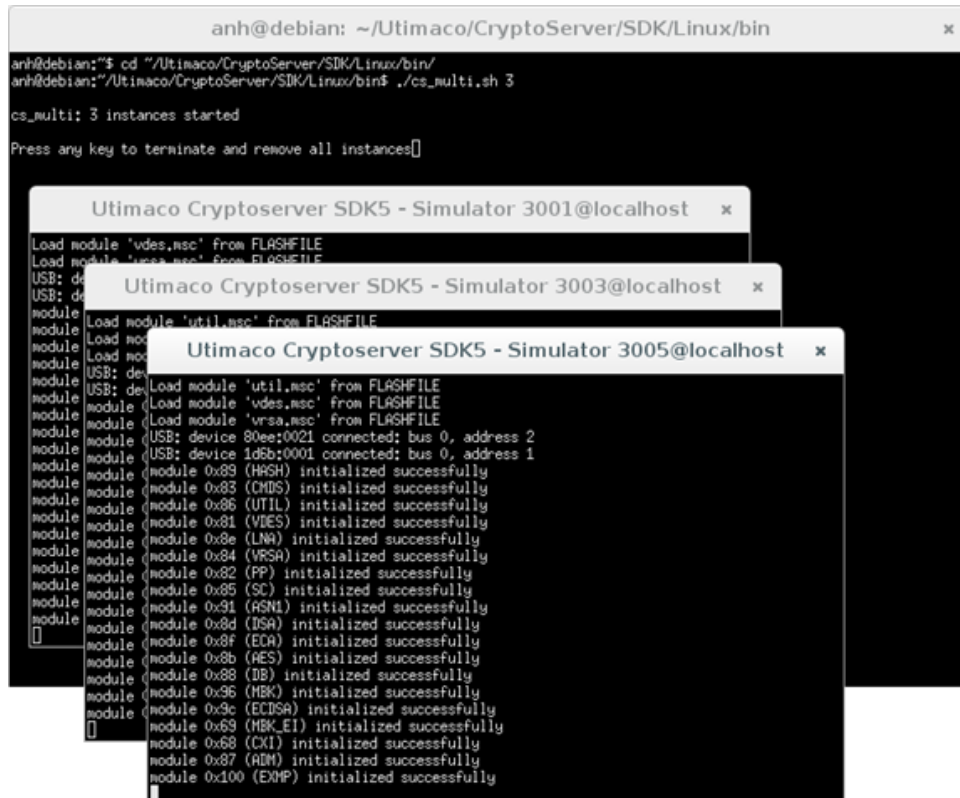


Figure 36 : Example for running three CryptoServer Simulator instances simultaneously (Linux)

To close all instances, press any key in the terminal. When the instances are closed, all your changes will be lost. Next time you start multiple CryptoServer Simulator instances they will have the same configuration as the one you have configured.

To only terminate the Simulator that is running in the currently active command shell, for example, the instance with the device address 3003@localhost, press **CTRL+C**. The other CryptoServer Simulator instances, that have been started, remain running and can be used.

6.10 Sensor Detection

Per default, the CryptoServer Simulator does not provide any support for the simulation of sensors, see [Sensors \(p. 17\)](#). But you can create a sensor alarm manually by editing the `ALARM.curr` file before starting the CryptoServer Simulator.

The default state of the `ALARM.curr` file is:

```
# SDK Alarm state:
# Bit 7 -- External Erase
# Bit 6 -- Power Low
# Bit 5 -- Power High
# Bit 4 -- Outer Foil
# Bit 3 -- Inner Foil
# Bit 2 -- Temp High
# Bit 1 -- Power Fail
# Bit 0 -- Temp Low

ALARM=00000000
```

If you want to set a "temperature too high" alarm, for example, change the value of the `ALARM` parameter to `00000100`. The alarm is indicated by a red alarm field in the status bar of the CAT main window. If you click the Show Status button in the CAT toolbar, the alarm is displayed in the information field as well:

```
=== Thursday, February 23, 2017 10:38:53 AM CET ===
=== CryptoServer GetState ===
mode      = Maintenance Mode
state     = INITIALIZED (0x000afd84)
temp      = 25.8 [C]
alarm     = ON
sens      = 02fd
           - Alarm has occurred
           - Temperature too high

bl_ver    = 5.01.1.0           (Model: Se-Series Gen2)
hw_ver    = 0.00.8.15
uid       = 4b61761d 25242a53 | Kav %$*S
adm1      = 5554494d 41434f20 43533030 30303030 | UTIMACO CS000000
adm2      = 53696d75 6c61746f 72205769 6e000000 | Simulator Win
adm3      = 496e6974 2d446576 2d312d4b 65790000 | Init-Dev-1-Key
```

You can combine two alarms at the same time, for example "temperature too high" alarm with a "power too low" alarm (`ALARM=01000100`).

You find the `ALARM.curr` file in: `C:\ProgramData\Utimaco\Simulator\devices`

If the `-d` option of the `bl_sim5.exe` file has been used, a different location of the devices directory might be used. For details on the `-d` option, see [Starting the Simulator \(p. 218\)](#).

The same alarms are used in the hardware CryptoServer as well but they are noticed and indicated automatically.

6.11 Device Address

If you use a single instance of a CryptoServer Simulator, its device address is `3001@127.0.0.1`. For handling multiple simulator instances, see [Starting Multiple CryptoServer Simulator Instances on Windows \(p. 221\)](#) and [Starting Multiple CryptoServer Simulator Instances on Linux \(p. 223\)](#).

The device addresses for hardware CryptoServers are different. It is e. g., `PCI:0` for a CryptoServer PCIe card mounted in the Windows host computer where CAT is installed. If this computer runs on Linux, `/dev/cs2.0` is an example. If you use a CryptoServer LAN, its IP address is its device address.



If you have developed your own firmware modules providing specific cryptographic functions and commands by using the CryptoServer Simulator, and you want to execute these firmware modules on a hardware CryptoServer, the device address is the only item that has to be changed.

6.12 Firmware Module Files

A firmware module file is a file compiled for the CryptoServer and ready for the interpretation of the CryptoServer operating system SMOS.

In case of a hardware CryptoServer, it is an `.out` file. It is a COFF file (common object file format).

In case of the CryptoServer SDK or the CryptoServer Simulator on Windows, it is a `.dll` file in the MS-DOS MZ executable file format, i.e. using MZ as the magical number at the beginning of the file.

In case of the CryptoServer Simulator on Linux, it is an `.so` file in the ELF format (executable and linking format).

`.out` files created for a hardware CryptoServer cannot be used on a CryptoServer Simulator, and `.dll` files and `.so` files created for the CryptoServer Simulator cannot be used on a hardware CryptoServer.

6.13 csadm Commands

The following `csadm` commands differ slightly depending on whether they are executed on the CryptoServer Simulator or on a hardware Cryptoserver:

The `ListFirmware` command lists among others information about the CPU target type the firmware module has been compiled for. It is `C64` or `C64+` for a hardware CryptoServer, `SDK` for the CryptoServer Simulator on Windows and `SIM` for the CryptoServer on Linux.

The `ListFiles` command lists among others information about the CPU target type the firmware module has been compiled for. It is `C64` or `C64+` for a hardware CryptoServer, `SDK` for the CryptoServer Simulator on Windows and `SIM` for the CryptoServer on Linux.

The `ModuleInfo` command lists among others information about the module type the firmware module has been compiled for. It is `CS2-COFF` for a hardware CryptoServer, `SDK-DLL` for the CryptoServer Simulator on Windows and `SDK-SO` for the CryptoServer on Linux.

The `ModuleInfo` command lists among others information about the CPU target type the firmware module has been compiled for. It is `C64+` or `C64x` for a hardware CryptoServer, `SDK` for the CryptoServer Simulator on Windows and `SIM` for the CryptoServer on Linux.

The `ListUser` command always lists the `I[0]` attribute value for the ADMIN user on the CryptoServer Simulator, but never the `I[1]` attribute value, i.e., the ADMIN user always can perform commands he/she needs an authentication for. There is no mandatory requirement for the ADMIN user to change his/her credentials.

See the *CryptoServer – csadm Manual* for more details on `csadm` commands.

7 Monitoring and Maintenance

7.1 Power Supply Monitoring

The power supply of the CryptoServer is provided by the following sources:

- **PCIe interface**

This is the usual power supply if the CryptoServer PCIe card has been mounted in a computer and this computer has been switched on or the CryptoServer PCIe card has been mounted in a CSLAN and this CSLAN has been switched on.

- **Carrier battery**

The carrier battery on the CryptoServer PCIe card is the backup power supply for the case that the power supply via the PCIe interface is not available.

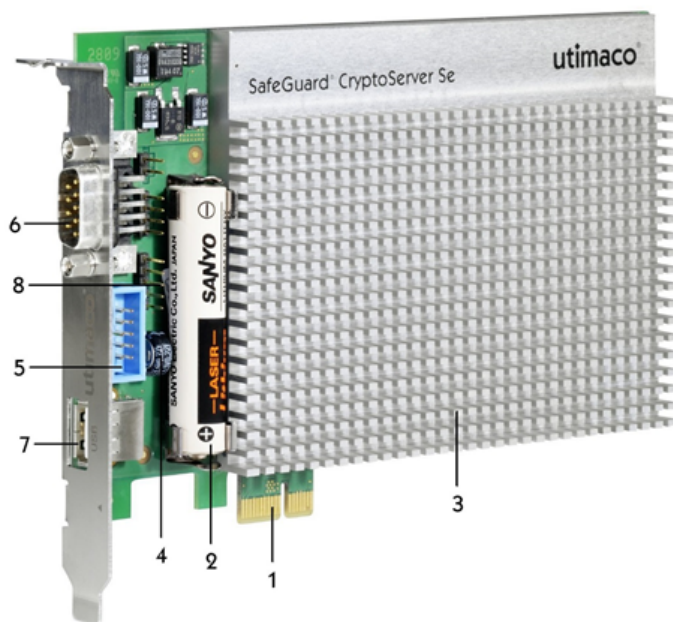


Figure 37 : The CryptoServer Se PCIe card

Item 2 in the figure indicates the carrier battery.

- **External battery**

The external battery is an optional power supply that can be provided in the CryptoServer LAN.

The external battery is the backup power supply for the case that the power supply via the PCIe interface is not available.



Figure 38 : The external battery of the CryptoServer LAN V4

▪ Capacitor

If neither the power supply via the PCIe port nor the carrier battery nor the external battery provide a sufficient power supply, it is provided by a capacitor on the CryptoServer PCIe card for a minimum of 5 minutes and a maximum of 30 minutes.

If even the capacitor cannot provide the required voltage anymore, an alarm is triggered to prevent keys or other sensitive data from being read-out by unauthorized means. The alarm state information is lost during power failure. A power failure occurs as well when the CryptoServer is started for the very first time, see [An Alarm and Its Consequences](#) (p. 38).

The level of the power supply determines the behavior of the server:

Voltage	Behavior of the CryptoServer
Below 2.55 V (carrier battery and external battery)	If the power supply via the PCIe port is not available anymore, the carrier battery voltage and the external battery voltage are below 2.55 V, and the capacitor does not provide any voltage anymore, an alarm is triggered, all keys and sensitive data on the CryptoServer are deleted, an entry is recorded in the CryptoServer's audit log file, and the CryptoServer is restarted. After the restart, the processor of the CryptoServer is suspended, and commands are not executed any longer. There is no lower threshold for the voltage provided via the PCIe port because it is assumed that there is a carrier battery or external battery available.

Voltage	Behavior of the CryptoServer
Carrier battery voltage and external battery voltage above 2.55 V and internal supply voltage below 7.9 V or power supply via the PCIe interface is present	Normal operation. Power is supplied to the sensors and the internal memory.
Above 7.9 V (internal supply voltage)	If the internal supply voltage is above 7.9 V, an alarm is triggered, all keys and sensitive data on the CryptoServer are deleted, an entry is recorded in the CryptoServer's audit log file, and the CryptoServer is restarted. After the restart the processor of the CryptoServer is suspended, and commands are not executed any longer. There is no upper threshold for the carrier battery voltage or the external battery voltage to trigger an alarm.

Table 48: Voltage ranges and alarms



If the carrier battery or the external battery has been replaced, the methods to retrieve the battery state (e.g., the csadm Dev=PCI:0 GetBattState command or by clicking the Show > Battery State menu in the CAT main window) do not show the current battery state. Wait at least one minute before retrieving the battery state.



The alarm triggered by the power supply monitoring sensors is a temporary alarm. Once the mains power returns to within the limits of the predefined threshold values, the reason for triggering this alarm is no longer present. CryptoServer can then be brought back into operation.

The table describes the replies to battery state requests that are initiated by e.g., the csadm Dev=PCI:0 GetBattState command or by clicking the Show > Battery State menu in the CAT main window. No automatic actions are initiated by the CryptoServer if the voltage falls below the described thresholds:

Voltage	Behavior of the CryptoServer
Below 2.6 V (external battery)	Requesting the battery state delivers a reply indicating among other things that the external battery is not available.

Voltage	Behavior of the CryptoServer
Below 2.75 V (carrier battery)	Requesting the battery state delivers a reply indicating among other things that the carrier battery voltage is too low.
Below 3.15 V (external battery)	Requesting the battery state delivers a reply indicating among other things that the external battery voltage is too low.

Table 49: Voltage ranges and state requests

7.2 Temperature Monitoring

The sensors monitor the CryptoServer's internal temperature whilst it is in operation. The table shows the permitted temperature ranges and also how the CryptoServer reacts if the temperature falls below or rises above the threshold values.

Temperature	Behaviour of CryptoServer
Below -12 °C (10.4 °F)	Alarm is triggered, all sensitive data on the CryptoServer is deleted, and the CryptoServer is restarted. After the restart, CryptoServer's processor is suspended, commands are not executed any longer.
-12 °C to -5 °C (10.4 °F to 23 °F)	The processor of the CryptoServer is suspended; a new entry is written into the audit log file; commands are not executed any longer. After being back to the normal operational temperature (-5 °C to 62 °C) the CryptoServer has to be restarted to be operational again.
-5 °C to 62 °C (23 °F to 143.6 °F)	Normal operation
62 °C to 66 °C (143.6 °F to 150.8 °F)	The processor of the CryptoServer is suspended; a new entry is written into the audit log file; commands are not executed any longer. After being back to the normal operational temperature (-5 °C to 62 °C) the CryptoServer has to be restarted to be operational again.
Above 66 °C (150.8 °F)	An alarm is triggered, all sensitive data on the CryptoServer is deleted and the security module is restarted. After the restart CryptoServer's processor is suspended, commands will not be executed any longer.

Table 50: Operational temperature of the CryptoServer



Once the CryptoServer's processor is suspended (i.e., its internal temperature exceeded or fell below the operational temperature range), it does not respond to any request. Any attempt to connect to the CryptoServer will usually result in a kind of timeout error from the device driver. Resetting the CryptoServer has no effect if its internal temperature still exceeds or is below

the operational temperature range. In case of CryptoServer's suspend mode caused by high temperature, we recommend switching off the power supply for some time in order to cool down the CryptoServer.

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.

An entry is recorded in the CryptoServer audit log file both when the device goes into sleep mode and when an alarm is triggered.

If the CryptoServer is in sleep mode, it can be reset to Operational Mode once its internal temperature returns to the permitted temperature range. To do this, you must perform a hardware reset on the CryptoServer. As no keys or other data are deleted when the CryptoServer goes into sleep mode, it returns to Operational Mode once the restart procedure is completed.

7.3 Logs

On the CryptoServer, all actions that involve security issues (for example, creating a new user) and events (especially the triggering of an alarm) are logged. There are two types of logs:

- the boot log and
- the audit log

7.3.1 Boot Log

Every time the CryptoServer starts up, i.e. after being switched on, and after a reset, every individual step and its results are recorded in the boot log. These include:

- The version number of the loaded operating system (SMOS)
- The version number of the internal sensor switch

- Information about whether a license file is present and which system settings have been made on the basis of this license file
- For every firmware module that was loaded and started by the operating system: a short description and unique ID to identify it, whether it could be initialized successfully or which errors occurred during the initialization process.

If any problems occur during the boot process, you can use the boot log to analyze the errors. The boot log is stored as a text file on the CryptoServer and is therefore available in a readable format after you import it.

7.3.2 Audit Log

In the audit log all events and actions involving security issues that occur or are performed whilst the CryptoServer is running are recorded. For example, these events are logged:

- The sensor triggers an alarm
- The permitted upper limit for the CryptoServer's internal temperature has been exceeded and the CryptoServer has therefore changed into sleep mode (power down mode)
- An alarm is reset
- The CryptoServer is completely deleted
- The date/time in the CryptoServer's real time clock (RTC) is reset
- Firmware modules or packets are loaded, deleted or replaced
- Users are created, changed or deleted or their "properties" are changed
- An audit log signature key (auditkey.db) is created.
- The audit log is deleted

For details on how to configure audit log entries, see *Configuring the Audit Log Files* in the *CryptoServer - CAT Manual*.

7.3.2.1 Format of the Audit Log Entries

```
DD.MM.YY hh:mm:ss [user] event data [returncode] <CR-LF>
```

Field	Meaning
DD.MM.YY	Date on which the event occurred in day.month.year format. The date is taken from the CryptoServer's internal real time clock.

Field	Meaning
hh:mm:ss	Time at which the event occurred in hour:minute:second format. The time is taken from the CryptoServer's internal real time clock.
[user]	Name(s) of the user(s) who performed the action. Multiple users are separated by commas. However, if an "external" event occurs, i.e. alarm or restart, the user ID is missing.
event	Description of the event in a readable format
data	Additional information about the event in a readable format
[returncode]	Return value of the function that has been performed. A return value of 0 means that the function has been performed successfully. A return code other than 0 shows the returned error number.
<CR-LF>	Carriage return - line feed

Table 51: Meaning of the audit log entry fields

The square brackets [...] displayed for user and returncode in the previous table do not mean that these fields are optional. In each case the relevant user name or return code is effectively enclosed in square brackets.

Audit Log Entries (p. 240) contains a list of all audit log entries and describes by which audit message classes they are activated.

7.3.2.2 Auditable Events

The CryptoServer offers extensive audit functionality. The following table gives an overview of the auditable events. For information on the configuration of auditable events, see *Configuring Auditable Events* in the *CryptoServer - csadm Manual*.



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

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

- A timestamp with date and time of the event (if the RTC is running),
- Username 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 are always logged:

Event	Audit log contains additional information about
New Alarm	Alarm description
The temperature exceeds the valid range	Measured temperature
<code>Clear</code> command performed	
<code>ResetAlarm</code> command performed	The special audit message class is used for this event
<code>SetAuditConfig</code> command performed	Old and new configuration values
FIPS restrictions applied mode	
FIPS mode	
FIPS mode entered	
FIPS mode left	
<code>Clear=DEFAULT</code> command performed	ERASE to factory settings
Erase executed	

Table 52: Events that are always logged into the audit log

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	Audit message class number	Audited by default?
Firmware and file management (e.g., commands to load or delete a file/firmware module, or load the Firmware Encryption Key)	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 the system clock of the CryptoServer	3	yes
CryptoServer startup	4	yes
Audit Log file management (e.g., deletion the audit log file, any changes in the audit configuration)	5	yes
Master Backup Key management (e.g., create or import a Master Backup Key)	6	yes

Auditable event	Audit message class number	Audited by default?
Key management (e.g., key management functions of the firmware module CXI)	7	no
Successful authentications/logins	8	no
Failed authentications/logins	9	yes
Backup and restore operations (e.g., backup database or restore database)	10	yes
Operating system events	11	yes
Action needed	12	yes
Audit message classes reserved for future use	13 – 23, 31	no
Customer-defined audit message classes	24 - 30	no
Audit message class always generating an audit log entry	-	no
Special audit message class	-	no

Table 53: Auditable events

7.3.2.3 Signed Audit Logs

The CryptoServer can generate signed audit log files on the computer csadm is running on.

The audit log signature key, see [Audit Log Signature Key \(p. 96\)](#), is used for this process. The generated files have extra long audit log file names, see *Audit Log File Names* in the [CryptoServer - csadm Manual \(p. 284\)](#). For details about how to generate and use this key, see *Generating and Verifying Signed Audit Log Files* in the [CryptoServer - csadm Manual \(p. 284\)](#).

7.3.2.4 Audit Log File Names

Depending on the CryptoServer version, the mode of the CryptoServer and the location the audit log files are stored, the schemas for building the audit log file names differ from each other. There are short, long and extra long audit log file names.

Short Audit Log File Names

Short audit log files names are numbered from audit_00.log up to and including audit_09.log. Short audit log file names are used in the following situations:

- CryptoServer version < V4.30 in Operational Mode or Maintenance Mode
- CryptoServer version ≥ V4.30, the CryptoServer is in Maintenance Mode and the system firmware modules (`*.sys` files) originate from a CryptoServer version < V4.30

These conditions are for example fulfilled for a CryptoServer with the following history:

- The CryptoServer has been initially delivered with a version < V4.30.
- The CryptoServer has been upgraded to V4.30.
- An External Erase has been performed, that means among other things, that the firmware modules including the ADM und SMOS firmware modules fall back on the SYS files of the CryptoServer version that has been initially delivered. The SYS files of a CryptoServer are always the SYS files of the initially delivered CryptoServer version (in the example < V4.30) even if the CryptoServer has been upgraded at a later date.

The system firmware modules are loaded onto the CryptoServer during an early phase of production, at the manufacturer's site. Even if the operating system or other firmware modules is later on deleted or replaced by newer versions, or if the CryptoServer is cleared by the customer or upgraded to a higher version, all system firmware modules remain unchanged.

In this situation, audit log files with long audit log file names cannot be read. This is as well the case if the CryptoServer is in the Bootloader Mode. Audit log files with short names are not signed and are stored in the FLASH memory of the CryptoServer.

Long Audit Log File Names

Long audit log files names are numbered from `audit000000.log` up to `auditFFFFFF.log`. These audit log files are not signed and are stored in the FLASH memory of the CryptoServer.

If the system firmware modules (`*.sys` files) originate from a CryptoServer version \geq V4.30 and if a CryptoServer version \geq V4.30 is present, long audit file names are used.

In the context of audit log files, the condition "CryptoServer version \geq 4.30" means as well that the conditions ADM version \geq 3.0.26.0 and $4.6.1.0 \leq$ SMOS version < 5.0 or SMOS version \geq 5.6.1.0 are fulfilled. If a CryptoServer < V4.30 is upgraded to CryptoServer \geq V4.30, all audit log files are renamed from short file names to long file names when the upgraded CryptoServer is booted for the first time. The same renaming process is performed, if a CryptoServer \geq V4.30 is in the Maintenance Mode with SYS files < V4.30 and then switches from Maintenance Mode to Operational Mode.

If an audit log file with a long audit log file name is full, the following line is appended to the file.

```
EOF <file number>
```

<file number> is the 6-digit hexadecimal number without a leading 0x that is part of the file name. If, for example, the audit000000.log file is full, the following line is appended.

```
EOF 000000
```

If rotation has been enabled, a new audit log file with a file number incremented by 1, for example, `audit000001.log`, is automatically created. The maximum value is `auditFFFFFF.log`. This means that the name of a full audit log file will never be reused to create a new audit log file, and that a maximum of 16,777,216 log files x 240,000 byte = 3.6 Tbyte can be stored in audit log files (not at one point in time). If the maximum number of allowed audit log files as specified by the `MaxFileCount` parameter when performing the `csadm SetAuditConfig` command has been exceeded by the creation of the new file, the oldest audit log file is automatically deleted.

If rotation has been disabled and all existing audit log files are full, the CryptoServer refuses executing any functions that are able to create an audit log entry. The only exceptions are functions for deleting audit log files. If an audit log file is deleted, this is logged as the first new audit log entry, see [Audit Log Entries \(p. 240\)](#).

Example 1

A CryptoServer < V4.30 in the Operational Mode with 10 audit log files from `audit_00.log` up to `audit_09.log` is upgraded to CryptoServer V4.30. The audit log files are renamed using long audit log file names when the CryptoServer V4.30 boots for the first time.

Short audit log file name	Long audit log file name	File number
<code>audit_00.log</code>	<code>audit000000.log</code>	0
<code>audit_01.log</code>	<code>audit000001.log</code>	1
<code>audit_00.log</code>	<code>audit000002.log</code>	2
...
<code>audit_09.log</code>	<code>audit000009.log</code>	9

Table 54: Change of audit log file names due to an upgrade to CryptoServer V4.30

Example 2

A CryptoServer V4.30 in the Operational Mode has 10 audit log files from `audit000000.log` up to `audit000009.log`. If a `audit000009.log` is full and rotation has been enabled, `audit000000.log` is deleted, EOF `000009` is appended to the `audit000009.log` file and the `audit00000A.log` file is created.

Example 3

A CryptoServer V4.30 in the Operational Mode has 10 audit log files from `audit000001.log` up to `audit00000A.log`. If all audit log files are deleted, a new file `audit00000B.log` is created.

Audit log file name	File number
<code>audit00000B.log</code>	B

Table 55: After the deletion of all audit log files



The file number in the tables above is equivalent to the <n> input parameter of the `csadm ClearAuditLogFiles` command, see *ClearAuditLogFiles* in the *CryptoServer - csadm Manual*.

Example 4

A CryptoServer has been upgraded from V4.21 to V4.30 and is in the Operational Mode. It has the audit log files `audit00000E.log` (full) and `audit00000F.log` (not full). This CryptoServer changes into Maintenance Mode. In this mode, further audit log messages are created so that the audit log files `audit_00.log` (full) and `audit_01.log` (not full) are created. When this CryptoServer boots the next time in Operational State, the following is performed.

`EOF 00000F` is appended to the `audit00000F.log` file.

- `audit_00.log` is renamed to `audit000010.log` and is terminated with `EOF 000010`.
- `audit_01.log` is renamed to `audit000011.log`.
- It is verified whether the current number of existing audit log files (4) is lower or equal to the allowed maximum number of audit log files (see the `csadm GetAuditConfig` command and the `csadm SetAuditConfig` command).

If this is the case, nothing happens.

If it is higher, nothing special happens at this moment either. Audit log messages are still written to the audit log file with the highest number. However, if this file is full and the files are non-rotating, any further processing is blocked until audit log files have been deleted (`csadm ClearAuditLog` command or `csadm ClearAuditLogFiles` command) and the number of audit log files is below its allowed maximum.

A CryptoServer should return to the Operational Mode as soon as possible. When the CryptoServer makes the transition from the Maintenance Mode to the Operational Mode, the `csadm GetAuditLog` command or the `csadm GetSignedAuditLog` command should be performed as soon as possible to store the audit log files permanently.

Extra Long Audit Log File Names

Extra long audit log file names are used for signed audit log files, which are created using the `csadm GetSignedAuditLog` command (see *GetSignedAuditLog* in the *CryptoServer - csadm Manual*). These files are stored on the computer csadm is running on. The names of these files are built according to the schema `<CryptoServer serial number>_<8-digit hexadecimal number>.log`.

The 8-digit hexadecimal number is derived from the 6-digit hexadecimal file number of the corresponding unsigned audit log file on the CryptoServer.

Example: From the unsigned audit log file `audit000000A.log` in the FLASH memory of the CryptoServer, the signed `CS123456_0000000A.log` file on the csadm computer is created. If you want to retrieve the serial number of your CryptoServer, perform a command according to the following example:

```
csadm Dev=PCI:0 GetState | grep adm1
```

Example output:

```
adm1 = 5554494d 41434f20 43533030 30303030 |UTIMACO CS123456|
```

The string before the last pipe symbol is the serial number. In this example, the serial number is `CS123456`.

Example using the CryptoServer Simulator:

```
csadm Dev=3001@127.0.0.1 GetState | grep adm1
```

Example output:

```
adm1 = 5554494d 41434f20 43533030 30303030 |UTIMACO SI003001|
```

The string before the last pipe symbol is the serial number. In this example, `SI003001` is the serial number of the CryptoServer Simulator because this simulator is listening on port 3001. SI stands for "simulator".

In case of the PaymentServer Simulator, the serial number is always `CS0000000`.

7.3.2.5 Audit Log Entries

The following table provides an overview of the used audit message classes. The following subchapters describe the audit log entries for these classes. In the CryptoServer Administration Tool (CAT) these classes are shown as events if you click **Manage > Audit Log Settings**.

<i>Audit message class name</i>	<i>Audit message class mask</i>	<i>Audit message class number</i>	<i>Description /Event name</i>	<i>Default</i>
OS_AUDIT_CLASS_FIRMWARE	0x00000001	1	Firmware management	Yes
OS_AUDIT_CLASS_USER	0x00000002	2	User management	Yes
OS_AUDIT_CLASS_TIME	0x00000004	3	Date/Time Management	Yes
OS_AUDIT_CLASS_STARTUP	0x00000008	4	Startup messages	Yes
OS_AUDIT_CLASS_AUDIT	0x00000010	5	Audit log management	Yes

<i>Audit message class name</i>	<i>Audit message class mask</i>	<i>Audit message class number</i>	<i>Description /Event name</i>	<i>Default</i>
OS_AUDIT_CLASS_MBK	0x00000020	6	MBK management	Yes
OS_AUDIT_CLASS_KEY	0x00000040	7	Key management	No
OS_AUDIT_CLASS_AUTH_SUCCESS	0x00000080	8	Successful login attempts	No
OS_AUDIT_CLASS_AUTH_FAILED	0x00000100	9	Failed login attempts	Yes
OS_AUDIT_CLASS_BACKUP_RESTORE	0x00000200	10	Backup/Restore	Yes
OS_AUDIT_CLASS_SYSTEM	0x00000400	11	Operating system events ^[1]	Yes
OS_AUDIT_CLASS_ACTION_NEEDED	0x00000800	12	Operations that require action	Yes
<Class reserved for future use>	-	13-23, 31	- 8	-
OS_AUDIT_CLASS_CUSTOM_MASK	-	24-30	- 8	-
OS_AUDIT_CLASS_ALWAYS ^[2]	0xFFFFFFFF	-	- 8	-
<Special audit message class>	-	-	- 8	-

Table 56: Audit message classes

[1] This audit message class is neither shown nor can it be assigned in the audit log configuration under **Manage > Audit log settings > Events** in the CryptoServer Administration Tool (CAT).

[2] If an audit message class name is set to OS_AUDIT_CLASS_ALWAYS, the audit log entry is written in any case, independently from the audit log configuration.

- The **first column** shows the name of the audit message class.
- The **second** and the **third columns** show the audit message class mask and number as used in the `csadm ... GetAuditConfig` command and in the `csadm ... SetAuditConfig` command.

Examples:

```
csadm dev=192.168.4.1 getauditconfig
Audit log configuration parameters:
Number of logfiles:          3
Rotate logfiles:            yes
Max filesize:               200000
Events:                     0x00000010 (Bits 5)
```

```
csadm Dev=192.168.4.1 LogonPass=sven,swordfish
SetAuditConfig=MaxFileCount=5,Events=1:2:3:4:5:6:8:9

csadm Dev=192.168.4.1 LogonSign=nils,:cs2:cjo:USB0
SetAuditConfig=RotateLogfiles=yes,Events=0x000001BF
```

In the examples, `0x00000010` and `0x000001BF` are the masks, and `5` and `1:2:3:4:5:6:8:9` are the corresponding numbers.

- The **fourth column** shows the audit message class as displayed in the CryptoServer administration tool (CAT) under **Manage > Audit log settings > Events**.



Figure 39 : Audit message classes (event names) in CAT

- The **fifth column** shows whether the audit message class is enabled by default.

The following table shows the function codes for the firmware modules:

Function code	Firmware module
0x000	SMOS
0x068	CXI
0x069	MBK, extended interface
0x083	CMDS
0x087	ADM
0x088	DB
0x096	MBK, normal interface
0x09A	NTP

Table 57: Function codes and firmware modules

Each firmware module specification contains more detailed information about its audit log entries.

The following table shows which function codes including the subfunction code and subfunction name belong to a dedicated audit message class. If a function code cannot be assigned to an audit message class, it is not shown in this table.

Audit message class name	Function code, subfunction code and subfunction information
OS_AUDIT_CLASS_FIRMWARE	FC:0x083 SFC:0x13 SetAdminMode15
	FC:0x083 SFC:0x14 SetStartupMode
	FC:0x087 SFC:0x03 LoadFile
	FC:0x087 SFC:0x04 DeleteFile
	FC:0x087 SFC:0x13 LoadFirmwareDecryptionKey
OS_AUDIT_CLASS_USER	FC:0x068 SFC:0x21 AddUser This is the CXI AddUser subfunction, not the CMDS AddUserExtended subfunction.
	FC:0x068 SFC:0x22 DeleteUser This is the CXI DeleteUser subfunction.
	FC:0x083 SFC:0x05 DeleteUser This is the CMDS DeleteUser subfunction.
	FC:0x083 SFC:0x06 ChangeUser

<i>Audit message class name</i>	<i>Function code, subfunction code and subfunction information</i>
	FC:0x083 SFC:0x0D RestoreUser This audit log entry is created if either OS_AUDIT_CLASS_BACKUP_RESTORE or OS_AUDIT_CLASS_USER is enabled.
	FC:0x083 SFC:0x0E AddUserExtended This is the CMDS AddUserExtended subfunction, not the CXI AddUser subfunction.
	FC:0x083 SFC:0x11 SetMaxAuthFailures Neither the function code nor the subfunction code is shown in the audit log entry.
OS_AUDIT_CLASS_TIME	FC:0x087 SFC:0x07 SetTime
	FC:0x09A SFC:0x02 ChangeActivationState
	FC:0x09A SFC:0x04 SetNTPSettings
OS_AUDIT_CLASS_STARTUP	FC:0x... SFC:0x... Checking configured authentication requirements in the audit log
	FC: 0x000 SCF: ... SMOS successfully started
OS_AUDIT_CLASS_AUDIT	FC:0x087 SFC:0x0B ClearAuditLog
	FC:0x087 SFC:0x1D GenerateAuditLogKey
	FC:0x087 SFC:0x20 ClearAuditLogFiles
OS_AUDIT_CLASS_MBK	FC:0x069 SFC:0x00 Import DES key from PIN pad Attention! This is the extended MBK interface with the function code 0x069, not the normal MBK interface with the function code 0x096. Neither the function code, nor the subfunction code is shown in the audit log entry.
	FC:0x069 SFC:0x01 Import DES key from smartcard16
	FC:0x069 SFC:0x02 Import AES key from smartcard16
	FC:0x069 SFC:0x03 Generate DES key and write to smartcard16
	FC:0x069 SFC:0x04 Generate AES key and write to smartcard16
	FC:0x069 SFC:0x08 Import MBK key from PIN pad and store on smartcard16
	FC:0x069 SFC:0x09 Generate DES key shares and store on smartcard16
	FC:0x069 SFC:0x0A Generate AES key shares on store on smartcard16
	FC:0x096 SFC:0x04 GenerateMBK
	FC:0x096 SFC:0x05 ImportMBK
OS_AUDIT_CLASS_KEY	FC:0x068 SFC:0x05 InitKeyGroup
	FC:0x068 SFC:0x08 BackupKey
	FC:0x068 SFC:0x09 RestoreKey
	FC:0x068 SFC:0x0B GenerateKey

<i>Audit message class name</i>	<i>Function code, subfunction code and subfunction information</i>
	FC:0x068 SFC:0x0C OpenKey OpenKey only creates audit log entries for SecurityServer/ CryptoServer SDK versions earlier than 4.30 and CXI firmware module versions earlier than 2.4.0.0.
	FC:0x068 SFC:0x0D DeleteKey
	FC:0x068 SFC:0x0F SetKeyProp
	FC:0x068 SFC:0x10 ExportKey
	FC:0x068 SFC:0x11 ImportKey
	FC:0x068 SFC:0x19 CreateObject
	FC:0x068 SFC:0x1A GenerateKeyPair
	FC:0x068 SFC:0x1B CopyObject
	FC:0x068 SFC:0x1C DeriveKey
	FC:0x068 SFC:0x1D WrapKey
	FC:0x068 SFC:0x1E UnwrapKey
	FC:0x068 SFC:0x2B SplitKey
OS_AUDIT_CLASS_AUTH_SUCCESS	-
OS_AUDIT_CLASS_AUTH_FAILED	-
OS_AUDIT_CLASS_BACKUP_RESTORE	FC:0x083 SFC:0x0C BackupUser
	FC:0x083 SFC:0x0D RestoreUser This audit log entry is created if either OS_AUDIT_CLASS_BACKUP_RESTORE or OS_AUDIT_CLASS_USER is enabled.
	FC:0x087 SFC:0x18 ExportDBEntry
	FC:0x087 SFC:0x19 ImportDBEntry
OS_AUDIT_CLASS_SYSTEM	FC: 0x000 SCF: ... CryptoServer shutdown
	FC: 0x000 SCF: ... SHA-512 known answer test failed
	FC: 0x000 SCF: ...Unpacking firmware module failed
	FC: 0x000 SCF: ... Loading firmware module failed
	FC: 0x000 SCF: ... Start function of firmware module failed
	FC: 0x000 SCF: ... Firmware module is defective
	FC: 0x000 SCF: ... DRBG: Known answer test failed
	FC: 0x000 SCF: ... DRBG: Initialization failed
	FC: 0x000 SCF: ... DRBG: Random number generation failed
	FC: 0x000 SCF: ... DRBG: Reseed failed
	FC: 0x000 SCF: ... TRNG: Hardware failed
	FC: 0x000 SCF: ... TRNG: Self-test failed
	FC: 0x000 SCF: ... TRNG: Online test failed

<i>Audit message class name</i>	<i>Function code, subfunction code and subfunction information</i>
	FC: 0x088 SFC: Corrupted database
OS_AUDIT_CLASS_ACTION_NEEDED	-
<Class reserved for future use>	-
OS_AUDIT_CLASS_CUSTOM_MASK	-
OS_AUDIT_CLASS_ALWAYS	FC: 0x000 SCF: ... FIPS mode entered or left
	FC: 0x000 SCF: ...Clear to factory settings
	FC: 0x000 SCF: ...Erase executed
	FC: 0x000 SCF: ... New alarm detected
	FC: 0x000 SCF: ...Temperature exceeds valid range
	FC:0x087 SFC:0x15 Clear
	FC:0x087 SFC:0x1A ConfigParamSet
<Special audit message class>	FC:0x087 SFC:0x14 ResetAlarm

Table 58: Function codes and subfunction codes of audit messages

The tables in the following chapters contain all possible audit log entries. They contain information about which of the following tools generates an audit log entry for which action (for example CAT: **Manage > Firmware Decryption Key**):

- csadm
- CryptoServer Administration Tool (CAT)
- PKCS#11 CryptoServer Administration Tool (p11CAT)
- p11tool2
- cxitool
- cngtool



API calls resulting in audit log entries are not listed.

7.3.2.5.1 OS_AUDIT_CLASS_FIRMWARE

The OS_AUDIT_CLASS_FIRMWARE audit message class is mainly used when firmware modules are loaded, deleted etc. The following tables describe the corresponding audit log entries and the actions leading to them.

<i>Audit log entry</i>	<i>Description</i>
SetAdminMode from <old> to <new> [error code]	Setting the administration mode Neither the function code (0x83) nor the subfunction code (0x13) is shown in the audit log entry. Initiation: csadm SetAdminMode
SetStartupMode <new> [error code]	Setting the startup mode Neither the function code (0x83) nor the subfunction code (0x14) is shown in the audit log entry. Initiation: csadm SetStartupMode
FC:0x087 SFC:0x03 Load File '<filename>' Part 1 [error code]	Loading the <filename> file. The part of the file is optional. Initiation: csadm LoadFile or csadm LoadPKG or CAT (Manage > Firmware > Update (installs only new firmware) or Manage > Firmware > New installation (deletes all files; installs new firmware package))
FC:0x087 SFC:0x03 Verification of MMC signature failed (<filename>) err = <error code>	The <filename> file (.mtc file) is a firmware module and the verification of its MMC signature has failed with the error <error code>. Initiation: csadm LoadFile
FC:0x087 SFC:0x04 Delete File '<filename>' [error code]	Deleting the <filename> file. Initiation: csadm DeleteFile or csadm LoadPKG with option ForceClear or CAT (Manage > Firmware > New installation (deletes all files; installs new firmware package))
FC:0x087 SFC:0x13 Load FW DecKey #R0 [error code]	Loading the firmware description key. Initiation: csadm LoadFWDecKey or CAT (Manage > Firmware Decryption Key)

Table 59: Audit log entries for OS_AUDIT_CLASS_FIRMWARE

7.3.2.5.2 OS_AUDIT_CLASS_USER

The OS_AUDIT_CLASS_USER audit message class is used when users are created, changed, restored and deleted. The following tables describe the corresponding audit log entries and the actions leading to them.

Audit log entry	Description
FC:0x068 SFC:0x21 user_add: <key group> <user name> (<Optional user name>, <authentication mechanism ID> <user's permission>) [error code]	Adding a user in CXI Initiation: P11CAT (e.g., Slot Management > Init Token > SO PIN, Confirm SO PIN > Init Token > Login)
FC:0x068 SFC:0x22 user_delete: <key group> <user name> [error code]	Deleting a user in CXI Initiation: P11CAT (Login/Logout > Login Generic > Login > Slot Management > Delete SO > Delete SO > Yes)
FC:0x083 SFC:0x05 Delete User '<user name>' [error code]	Deleting a user in CMDS Initiation: <code>csadm DeleteUser</code> or CAT (Manage > User > Delete User)
FC:0x083 SFC:0x06 Change User '<user name>' [error code]	Changing a user. Initiation: <code>csadm ChangeUser</code> CAT (Manage > User > Change Token/Password)
FC:0x083 SFC:0x0D Restore User '<user name>' (<Optional user name>, <authentication mechanism ID> <user's permission>) [error code]	Restoring a user. Initiation: <code>csadm RestoreUser</code> or CAT (Manage > Manage User > Restore Users) This audit log entry is created if either OS_AUDIT_CLASS_BACKUP_RESTORE or OS_AUDIT_CLASS_USER is enabled.
FC:0x083 SFC:0x0E Add User '<user name>' (<Optional user name>, <authentication mechanism ID> <user's permission>) [error code]	Adding a user in CMDS Initiation: <code>csadm AddUser</code> or CAT (Manage > User > Add User)
Set MaxAuthFailures from <old> to <new> [error code]	Setting the maximum number of authentication failures Neither the function code (FC: 0x083) nor the subfunction code (SFC: 0x11) is shown in the audit log entry. Initiation: <code>csadm SetMaxAuthFails</code>

Table 60: Audit log entries for OS_AUDIT_CLASS_USER

7.3.2.5.3 OS_AUDIT_CLASS_TIME

The OS_AUDIT_CLASS_TIME audit message class is used when the time is set, the limitations for time changes changed and the activation state for these changes is changed.

The following tables describe the corresponding audit log entries and the actions leading to them.

Audit log entry	Description
FC:0x087 SFC:0x07 Set Time from <date/time stamp> [error code]	Setting the time Initiation: csadm SetTime or CAT (Manage > Date/Time)
FC:0x09A SFC:0x02 NTP: New Settings MaxAdjustPerOperation=<value1> MaxAdjustPerDay=<value2> [error code]	Changing the activation state Initiation: CAT (Manage > NTP Settings > Enabled / Disabled)
FC:0x9A7 SFC:0x04 NTP: New Settings MaxAdjustPerOperation=<value1> MaxAdjustPerDay=<value2> [error code]	Setting the NTP settings Initiation: CAT (Manage > NTP Settings > Max. time to set per day and Manage > NTP Settings > Max. time to set per operation)

Table 61: Audit log entries for OS_AUDIT_CLASS_TIME

7.3.2.5.4 OS_AUDIT_CLASS_STARTUP

The OS_AUDIT_CLASS_STARTUP audit message class is used when the configured authentication requirements in the audit log are checked the power-on self-test has failed, the cryptographic algorithm test has failed, SMOS is started etc. The following tables describe the corresponding audit log entries and the actions leading to them.

Audit log entry	Description
FC:<function code> SFC: <subfunction code> Configured Permission=<value>	Checking the configure authentication requirements in the audit log Initiation: csadm - Edit the cmds.scf file, perform csadm SignConfig, csadm LoadFile=...\cmds.scf, csadm ListFiles, csadm Restart, csadm GetBootLog and csadm GetAuditLog For details, see <i>Using the Signed Configuration File cmds.scf</i> in the <i>u.trust Anchor - csadm Manual</i> and <i>Creating and Using the Signed Configuration File cmds.scf</i> in the <i>CryptoServer – csadm Manual</i> .

<i>Audit log entry</i>	<i>Description</i>
CMD5: unable to initialize module POST for Power-on Self-tests (err = <error code>)	Failed power-on self-test initialization. Initiation: The POST (power-on self-test) firmware module is not available and one of the following actions is performed: Windows command-line (shutdown -r -t 0), csadm Restart or csadm CSLReboot or CAT: Manage > Reboot CryptoServer
POST: Selftest of utility functions failed with error code: <error code>	Failed power-on self-test
POST: <Algorithm name> Cryptographic Test failed with error code: <error code>	Failed cryptographic algorithm test, i.e., for AES, ECDA, HASH, HMAC RSA, DSA or DES
POST: <Algorithm name> Cryptographic Test skipped	Skipped cryptographic algorithm test, i.e., for AES, ECDA, HASH, HMAC RSA, DSA or DES Initiation: The corresponding firmware module (AES, ECDSA etc.) is not available and one of the following actions is performed: Windows command-line (shutdown -r -t 0), csadm Restart or csadm CSLReboot . Or, CAT: Manage > Reboot CryptoServer
SMOS Ver. <version> successfully started	SMOS has been started successfully Initiation: csadm Restart or CAT: Manage > Reboot CryptoServer

Table 62: Audit log entries for OS_AUDIT_CLASS_STARTUP

7.3.2.5.5 OS_AUDIT_CLASS_AUDIT

The OS_AUDIT_CLASS_AUDIT audit message class is used when the audit log configuration is cleared or changed. The following tables describe the corresponding audit log entries and the actions leading to them.

Audit log entry	Description
FC:0x087 SFC:0x0B Clear Audit Files '<filename>' (<number>) [<error code>]	<p>Deleting audit log file(s)</p> <p><number> indicates the number of the (youngest) parts that are not deleted (omitted in error case).</p> <p>Initiation:</p> <p><code>csadm ClearAuditLog</code> or</p> <p>CAT: Manage > Audit Log > Clear Log > Yes.</p> <p>These actions can be initiated in the Operational Mode and in the Maintenance Mode.</p> <p>See below for another method to delete audit log files (SFC = 0x20).</p>
<p>Successful execution:</p> <p>FC:0x087 SFC:0x1D Generate Audit Log Key with mode <mode> and public key fingerprint <fingerprint></p> <p>Failed execution:</p> <p>FC:0x087 SFC:0x1D Generate Audit Log Key ERROR [<error code>]</p>	<p>Generating an audit log signature key</p> <p><mode> indicates the mode of the audit log signature key (see description of the <mode> parameter in section <i>GenerateAuditLogKey</i> in u.trust Anchor - csadm Manual for details) and <fingerprint> indicates the public key fingerprint.</p> <p>Initiation: <code>csadm GenerateAuditLogKey</code></p>
<p>Successful execution:</p> <p>FC:0x087 SFC:0x20 Clear Audit Files (audit<first>.log - audit<last>.log)</p> <p>Failed execution:</p> <p>FC:0x087 SFC:0x20 Clear Audit Files ERROR [<error code>]</p>	<p>Deleting audit log file(s)</p> <p><first> indicates the first audit log file that has been deleted, and <last> indicates the last file that has been deleted with both of them being a 6-digit hexadecimal number without a leading 0x.</p> <p>Initiation: <code>csadm ClearAuditLogFiles</code></p> <p>See above for another method to delete audit log files (SFC = 0x0B).</p>

Table 63: Audit log entries for OS_AUDIT_CLASS_AUDIT

7.3.2.5.6 OS_AUDIT_CLASS_MBK

The OS_AUDIT_CLASS_MBK audit message class is used when an MBK (master backup key) is generated, imported and stored. The following tables describe the corresponding audit log entries and the actions leading to them.

Audit log entry	Description
mbk_ei_import_pp: DES-16, slot <slot number> [error code]	Importing a DES key from the PIN pad. Neither the function code (FC:0x069) nor the subfunction code (SFC:0x00) is shown in this audit log entry.
mbk_ei_import_des_sc: DES- 16, '<key name>', slot <slot number> [error code]	Importing a DES key from the smartcard. Neither the function code (FC:0x069) nor the subfunction code (SFC:0x01) is shown in this audit log entry.


Audit log entry	Description
mbk_ei_import_aes_sc: AES- 32, '<key name>', slot <slot number> [error code]	Importing an AES key from the smartcard. Neither the function code (FC:0x069) nor the subfunction code (SFC:0x02) is shown in this audit log entry.
mbk_ei_generate_des_sc: DES-16, '<key name>', record <record number> [error code]	Generating a DES key and writing to the smartcard. Neither the function code (FC:0x069) nor the subfunction code (SFC:0x03) is shown in this audit log entry.
mbk_ei_generate_aes_sc: AES-32, '<key name>', record <record number> [error code]	Generating an AES key and writing to the smartcard. Neither the function code (FC:0x069) nor the subfunction code (SFC:0x04) is shown in this audit log entry.
mbk_ei_imp_pp_write_sc: <key info>, '<key name>', record <record number> [error code]	Importing an MBK key from the PIN pad and storing them on the smartcard. Neither the function code (FC:0x069) nor the subfunction code (SFC:0x08) is shown in this audit log entry.
mbk_ei_gen_des_key_shares: DES-16 '<key name>', <k>- out-of-<n>, record <record number> [error code]	Generating DES key shares and storing them on the smartcard. Neither the function code (FC:0x069) nor the subfunction code (SFC:0x09) is shown in this audit log entry
mbk_ei_gen_aes_key_shares: AES-32 '<key name>', <k>- out-of-<n>, record <record number> [error code]	Generating AES key shares and storing them on the smartcard. Neither the function code (FC:0x069) nor the subfunction code (SFC:0x0A) is shown in this audit log entry.
mbk_key_generate: <key info> '<key name>' <k>- out-of-<n> [error code]	Generating an MBK. Initiation: csadm MBKGenerateKey or CAT: Manage > Master Backup Key > Generate
mbk_key_import: <key info> '<key name>', <x> parts, slot <slot number> [error code]	Importing an MBK. Initiation: csadm MBKImportKey or CAT: Manage > Master Backup Key > Import

Table 64: Audit log entries for S_AUDIT_CLASS_MBK

7.3.2.5.7 AUDIT_CLASS_KEY

The OS_AUDIT_CLASS_KEY audit message class is used when keys are generated, backed up, restored, opened, deleted, exported, imported, derived, wrapped and unwrapped and additional processes are performed. The following tables describe the corresponding audit log entries and the actions leading to them.

Audit log entry	Description
FC:0x068 SFC:0x05 key_init: <key group> [error code]	Initiating a key group Initiation: By the P11CAT or p11tool2 program when reinitializing a PKCS#11 slot, thereby deleting all key material.

Audit log entry	Description
FC:0x068 SFC:0x08 key_backup: <key info>, <key group>: <key name>: <key specifier> [error code]	Backing up a key Initiation: Call by the appropriate method of the keyCAT, P11CAT, p11tool2 or cxitool program although backing up a key is not a PKCS#11 function.
FC:0x068 SFC:0x09 key_restore: <key info>, <key group>: <key name>: <key specifier> [error code]	Restoring a key Initiation: Call by the appropriate method of the keyCAT, P11CAT, p11tool2 or cxitool program although restoring a key is not a PKCS#11 function.
FC:0x068 SFC:0x0B key_generate: <key info>, <key group>: <key name>: <key specifier> [error code]	Generating a key. Initiation: Call by the appropriate method of the csadm, P11CAT, p11tool2, cxitool or cngtool program.
FC:0x068 SFC:0x0C key_open: <key group>: <key name>: <key specifier> [error code]	Opening a key. Initiation: Call by the appropriate method of the csadm, CAT or P11CAT program. <div data-bbox="676 943 1383 1200"> Opening a key only creates audit log entries for SecurityServer/CryptoServer SDK versions earlier than 4.30 and CXI firmware module versions earlier than 2.4.0.0</div>
FC:0x068 SFC:0x0D key_delete: <key info>, <key group>: <key name>: <key specifier> [error code]	Deleting a key. Initiation: Call by the appropriate method of the P11CAT, p11tool2, cxitool or cngtool program.
FC:0x068 SFC:0x0F key_prop_set: <key group>: <key name>: <key specifier> [error code]	Setting key properties. Initiation: Call by the appropriate method of the keyCAT, P11CAT program by changing key properties e.g., the label of a key.
FC:0x068 SFC:0x10 key_export: <key info>, <key group>: <key name>: <key specifier> [error code]	Exporting a key. Initiation: Call by the appropriate method of the keyCAT cngtool program.
FC:0x068 SFC:0x1 key_import: <key info>, <key group>: <key name>: <key specifier> [error code]	Importing a key. Initiation: Call by the appropriate method of the keyCAT cngtool program.

<i>Audit log entry</i>	<i>Description</i>
FC:0x068 SFC:0x19 obj_create: <key info>, <key group>: <key name>: <key specifier> [error code]	Creating an object. Initiation: Call by the appropriate method of the P11CAT or p11tool2 program.
FC:0x068 SFC:0x1A key_pair_gen: <key info>, <key group>: <public key name>: <public key specifier> / <private key name>: <private key specifier> [error code]	Generating a key pair. Initiation: Call by the appropriate method of the P11CAT or p11tool2.
FC:0x068 SFC:0x1B cxi_ext_obj_copy: <key info>, <key group>: <source key name>: <source key specifier> > <destination key group>: <destination key name>: <destination key specifier> [error code]	Copying an object. Initiation: Call by the appropriate method of the P11CAT or p11tool2 program.
FC:0x068 SFC:0x1C key_derive: <key info>, <key group>: <key name>: <key specifier> [error code]	Deriving a key. Initiation: Call by the appropriate method of the P11CAT or p11tool2 program.
FC:0x068 SFC:0x1D key_wrap: <key info>, <key group>: <key name>: <key specifier> [error code]	Wrapping a key. Initiation: Call by the appropriate method of the P11CAT or p11tool2 program.
FC:0x068 SFC:0x1E key_unwrap: <key info>, <key group>: <key name>: <key specifier> [error code]	Unwrapping a key. Initiation: Call by the appropriate method of the P11CAT or p11tool2 program.
FC:0x068 SFC:0x2B key_split: <key info>, <key group>: <key name>: <key specifier> [error code]	Splitting a key. Initiation: Appropriate call to the CXI firmware module.

Table 65: Audit log entries for OS_AUDIT_CLASS_KEY

7.3.2.5.8 OS_AUDIT_CLASS_AUTH_SUCCESS

The OS_AUDIT_CLASS_AUTH_SUCCESS audit message class is used when the authentication of a user succeeded. The following tables describe the corresponding audit log entries and the actions leading to them.

<i>Audit log entry</i>	<i>Description</i>
authentication (<authentication mechanism ID>) succeeded [error code]	Successful authentication. Initiation: csadm LogonPass or csadm LogonSign p11tool2 LoginUser cxitool LogonPass or cxitool LogonSign cngtool automatic login CAT: e.g. Login/Logout > Login Generic > User name > Login

Table 66: Audit log entries for OS_AUDIT_CLASS_AUTH_SUCCESS

7.3.2.5.9 OS_AUDIT_CLASS_AUTH_FAILED

The OS_AUDIT_CLASS_AUTH_FAILED audit message class is used when an authentication attempt fails. The following tables describe the corresponding audit log entries and the actions leading to them.

<i>Audit log entry</i>	<i>Description</i>
'<user>' authentication (<authentication mechanism ID>) failed, failure counter: <number> [error code]	Failed authentication without maximum value for failed authentication attempts set. Initiation: Call by the appropriate method of the csadm, CAT, P11CAT, p11tool2, cxitool or cngtool program.
'<user>' authentication (<authentication mechanism ID>) failed, <number> attempts left [error code]	Failed authentication with maximum value for failed authentication attempts set. Initiation: Call by the appropriate method of the csadm, CAT, P11CAT, p11tool2, cxitool or cngtool program.
'unknown' Authentication failed [error code]	Failed authentication due to a wrong user name. Initiation: Call by the appropriate method of the csadm, CAT, P11CAT, p11tool2, cxitool or cngtool program
'<user>' account locked, authentication refused [error code]	Too many failed authentication attempts. Initiation: Call by the appropriate method of the csadm, CAT, P11CAT, p11tool2, cxitool or cngtool program.

Table 67: Audit log entries for OS_AUDIT_CLASS_AUTH_FAILED

7.3.2.5.10 OS_AUDIT_CLASS_BACKUP_RESTORE




The OS_AUDIT_CLASS_BACKUP_RESTORE audit message class is used when a user is backed up or restored and a database entry is exported or imported. The following tables describe the corresponding audit log entries and the actions leading to them.






<i>Audit log entry</i>	<i>Description</i>
FC:0x083 SFC:0x0C Backup User '<user name> [error code]	Backing up a user. Initiation: csadm BackupUser or CAT: Manage > Manage User > Backup Users
FC:0x083 SFC:0x0D Restore User '<user name> (<Optional user name>, <authentication mechanism ID> <user's permission>) [error code]	Restoring a user. Initiation: csadm RestoreUser or CAT Manage > Manage User > Restore Users This audit log entry is created if either OS_AUDIT_CLASS_BACKUP_RESTORE or OS_AUDIT_CLASS_USER is enabled.
FC:0x087 SFC:0x18 Backup DB Entry (Database / Searchkey): '<DB name>' / '<user name>' [error code]	Exporting a database entry. Initiation: csadm BackupDatabase or CAT Manage > Backup/Restore > Copy databases from Source CryptoServer to Backup directory > Select database > Add > Execute)
FC:0x087 SFC:0x19 Restore DB Entry (Database / Searchkey): '<DB name>' / '<user name>' [error code]	Importing a database entry. Initiation: csadm RestoreDatabase or CAT Manage > Backup/Restore > Copy databases from Backup directory to Source CryptoServer > Select database > Add > Execute

Table 68: Audit log entries for OS_AUDIT_CLASS_BACKUP_RESTORE

7.3.2.5.11 OS_AUDIT_CLASS_SYSTEM

The OS_AUDIT_CLASS_SYSTEM audit message class is used when the power-on self-test succeeded, the u.trust Anchor cHSM is shut down and some operations failed. The following tables describe the corresponding audit log entries and the actions leading to them.

Audit log entry	Description
FC: 0x088 SFC: Database '<database name>' corrupted (integrity failure detected)	Corrupted database
CMDS: Power-on Self-tests performed successfully	Successful power-on self-test Initiation: Windows command (shutdown -r -t 0) csadm Reset (not recommended), csadm Restart or csadm CSLReboot CAT: Manage > Reboot CryptoServer
[] FC:0x000 SFC: CryptoServer shut down	CryptoServer shutdown. cHSM shutdown Initiation: csadm Reset (not recommended), csadm ResetToBL , csadm Restart, csadm CSLShutdown or csadm CSLReboot) CAT: Manage > Reboot CryptoServer  This applies for V4.5.2.0 ≤ SMOS version V5.0.0.0 or SMOS version ≥ V5.5.4.0.
[] FC:0x000 SFC: DRBG: known answer test failed [<error code>]	The known answer test of the deterministic random bit generator (DRBG) has failed. This test verifies whether a device produces the expected output as a reaction to a given input.  This applies for V4.5.2.0 ≤ SMOS version V5.0.0.0 or SMOS version ≥ V5.5.4.0
[] FC:0x000 SFC: DRBG: initialization of '<profile name>' failed [<error code>]	Failed DRBG initialization The profile name is either 'Real' or 'Pseudo'.  This applies for V4.5.2.0 ≤ SMOS version V5.0.0.0 or SMOS version ≥ V5.5.4.0

<i>Audit log entry</i>	<i>Description</i>
[] FC:0x000 SFC: DRBG: generation failed [<error code>]	<p>Failed deterministic random number generation</p> <hr/> <p> This applies for V4.5.2.0 ≤ SMOS version V5.0.0.0 or SMOS version ≥ V5.5.4.0</p> <hr/>
[] FC:0x000 SFC: DRBG: reseed failed [<error code>]	<p>Failed reseed of the deterministic random bit generator</p> <hr/> <p> This applies for V4.5.2.0 ≤ SMOS version V5.0.0.0 or SMOS version ≥ V5.5.4.0</p> <hr/>
[] FC:0x000 SFC: Module <function code> (<module name>) is corrupt [<error code>]	<p>The firmware module has been detected as defective due to a failed integrity check. The function code is identical to the module ID.</p> <hr/> <p> This applies for V4.5.2.0 ≤ SMOS version V5.0.0.0 or SMOS version ≥ V5.5.4.0</p> <hr/>
[] FC:0x000 SFC: TRNG: hardware failure [<error code>]	<p>TRNG (true random number generator) hardware failure</p> <hr/> <p> This applies for V4.5.2.0 ≤ SMOS version V5.0.0.0 or SMOS version ≥ V5.5.4.0</p> <hr/>
[] FC:0x000 SFC: TRNG: selftest failed [<error code>]	<p>Failed TRNG self-test</p> <hr/> <p> This applies for V4.5.2.0 ≤ SMOS version V5.0.0.0 or SMOS version ≥ V5.5.4.0</p> <hr/>

<i>Audit log entry</i>	<i>Description</i>
[] FC:0x000 SFC: TRNG: online test failed	<p>Poor statistical quality of a random number</p> <hr/> <p> This applies for V4.5.2.0 ≤ SMOS version V5.0.0.0 or SMOS version ≥ V5.5.4.0</p> <hr/>
FC:0x000 SFC: SHA-512 Known Answer Test failed	<p>Known Answer Test failed</p> <hr/> <p> This applies for V4.5.2.0 ≤ SMOS version V5.0.0.0 or SMOS version ≥ V5.5.4.0</p> <hr/>
[] FC:0x000 SFC: Unpack of module '<module name>' failed [error code]	<p>Unpacking a firmware module file, e.g., <code>adm.msc</code>, has failed</p> <hr/> <p> This applies for V4.5.2.0 ≤ SMOS version V5.0.0.0 or SMOS version ≥ V5.5.4.0</p> <hr/>
[] FC:0x000 SFC: Loading of module '<module name>' failed [error code]	<p>Loading a firmware module file, e.g., <code>adm.msc</code>, has failed</p> <hr/> <p> This applies for V4.5.2.0 ≤ SMOS version V5.0.0.0 or SMOS version ≥ V5.5.4.0</p> <hr/>
[] FC:0x000 SFC: Start function of module '<module name>' failed [error code]	<p>Start function of a firmware module file, e.g., <code>adm.msc</code>, has failed.</p> <hr/> <p> This applies for V4.5.2.0 ≤ SMOS version V5.0.0.0 or SMOS version ≥ V5.5.4.0</p> <hr/>

Table 69: Audit log entries for OS_AUDIT_CLASS_SYSTEM

7.3.2.5.12 Class reserved for future use

This audit message class is used for future extensions

7.3.2.5.13 OS_AUDIT_CLASS_CUSTOM_MASK

The OS_AUDIT_CLASS_CUSTOM_MASK audit message class is only used for extension implemented by the customer.

7.3.2.5.14 OS_AUDIT_CLASS_ALWAYS

If an audit message class name is set to OS_AUDIT_CLASS_ALWAYS, the audit log entry is written in any case.

The OS_AUDIT_CLASS_ALWAYS audit message class is used when the device is cleared, a certification mode (CC, FIPS) has to be handled, a new alarm is detected, the temperature exceeds its valid range etc. The following tables describe the corresponding audit log entries and the actions leading to them.

<i>Audit log entry</i>	<i>Description</i>
FC:0x087 SFC:0x15 Clear [error code]	Clearing the device Initiation: <code>csadm Clear</code> or CAT Manage > Clear
FC:0x087 SFC:0x1A change audit config param id: <ID> (<ID specifier>) from: <old> to: <new> [error code]	Changing audit log parameter. Initiation: <code>csadm SetAuditConfig</code> or CAT: Manage > Audit Log > Change configuration > OK
FIPS restrictions applied mode = <ON/OFF>, FIPS error state = <ON/OFF> [error code]	FIPS restrictions applied mode
FIPS mode = <ON/OFF>, FIPS error state = <ON/OFF> [error code]	FIPS mode. Initiation: Call by the appropriate method of the <code>csadm</code> or CAT.
[] FC:0x000 SFC: CryptoServer has entered FIPS mode	The CryptoServer has entered the FIPS mode. Initiation: Call by the appropriate method of the <code>csadm</code> or CAT.
[] FC:0x000 SFC: CryptoServer has left FIPS mode	The CryptoServer has left the FIPS mode. Initiation: Call by the appropriate method of the <code>csadm</code> or CAT program.

Audit log entry	Description
[] FC:0x000 SFC: CryptoServer ERASE to factory setting	CryptoServer ERASE to factory settings. Initiation: Call by the appropriate method of the csadm (including <code>Clear=DEFAULT</code>) or CAT (including Manage > Clear to Factory Settings). Consider that all sensitive data is removed during this process. If you use the CryptoServer Simulator, use the <code>ALARM.curr</code> file for performing the External Erase, see Sensor Detection (p. 225) for details.
[] FC:0x000 SFC: New ALARM detected: <value> (<description>)	New alarm. Initiation: <ul style="list-style-type: none"> See, An Alarm and Its Consequences (p. 38) first, then Sensor Detection (p. 225) for a list of possible alarms. This chapter also describes how to create an alarm if you use a CryptoServer Simulator. In this case, you must perform a <code>csadm ... Restart</code> command or CAT > Manage > Reboot CryptoServer before the alarm is active. Consider that all sensitive data is removed during this process.
[] FC:0x000 SFC: Temperature exceeds valid range (<value>°C): Shutdown !	The temperature exceeds 62 °C or falls below -5°C, see 2.5.11, "Temperature Monitoring" for details.
[] FC:0x000 SFC: CryptoServer ERASE executed	ERASE executed. Initiation: See, An Alarm and Its Consequences (p. 38) first, then follow the instructions in <i>Performing an External Erase</i> in the <i>CryptoServer - CAT Manual</i> . Consider that all sensitive data is removed during this process.

Table 70: Audit log entries for OS_AUDIT_CLASS_ALWAYS

7.3.2.5.15 Special Audit Message Class

A special audit message class is used for an audit log entry due to an alarm that is reset. This class ensures that an audit log entry is always created, independently from the configuration of the audit log. Only if a reset alarm cannot be logged due to a full and non-rotating audit log, as an exception resetting this alarm is executed without logging.

Audit log entry	Description
FC:0x087 SFC:0x14 Reset Alarm [error code]	Resetting an alarm. Initiation: <code>csadm ResetAlarm</code> or CAT Manage > Reset alarm

Table 71: Audit log entry for the special audit message class

7.3.2.5.16 OS_AUDIT_CLASS_ACTION_NEEDED

The OS_AUDIT_CLASS_ACTION_NEEDED audit message class is used to alert the user, that host must be updated. The following tables describe the corresponding audit log entries and the actions leading to them.


<i>Audit log entry</i>	<i>Description</i>
user <username> authenticated with old HMAC mech and host must be updated	<p>A user that has HMAC authentication and logs in using old host software (without PBKDF) and new firmware (with PBKDF) is successfully logged in and a new entry is added to the audit log.</p> <div> We encourage the user to upgrade the host software and use a more secure mechanism of HMAC authentication.</div>

Table 72: Audit log entries for OS_AUDIT_CLASS_ACTION_NEEDED

8 Troubleshooting

8.1 Gathering Diagnostic Information

If a problem occurs whilst the CryptoServer is running, you can call a range of status information that may help you sort out the problem.

To view the most important status information, you can use CAT or csadm.

8.1.1 Gathering Diagnostic Information with CAT

If a problem occurs while the CryptoServer is running, you can call a range of status information that may help you sort out the problem.

To view the most important status information, perform the following steps:

1. Start CAT.
2. Click the **Show** menu.
3. Select **Diagnostics**.
 - The **Save Diagnostics** dialog box opens. The following diagnostic information is displayed:
 - The current date and time on the host computer when the diagnostics query was sent to the CryptoServer.
 - The CAT version
 - The address of the CryptoServer or the IP address of the CryptoServer LAN
 - The CryptoServer status
 - The boot log
 - Driver information (GetInfo)
 - The battery state of the carrier battery and the external battery
 - All files currently present on the CryptoServer
 - All active firmware modules on the CryptoServer
 - All users set up on the CryptoServer
 - Date and time on the CryptoServer
 - The alarm log (only displayed if bootloader version ≤ 2.5 is loaded on the CryptoServer)

- Information about the Master Backup Key that is saved on the CryptoServer
4. Click **Save**.
 - A dialogue box opens to determine the file location.
 5. Select the location, e.g., on your computer, for saving the file and enter an appropriate file name.
 6. Click **Save** to save the `.txt` file.
 7. Click **Cancel**.
 - The **Save Diagnostics** dialog box closes.
 8. Click the **Show** menu.
 9. Select **Audit Log**.
 - The **CryptoServer Audit Log** dialog box opens.
 - You can filter the display containing the log entries by users and by commands.
 10. Click **Save Log**.
 - A dialogue box opens to determine the file location.
 11. Select the location, e.g., on your computer, for saving the audit log and enter an appropriate file name.
 12. Click **Save** to save the `.log` file.
 13. Click **Close** to close the **CryptoServer Audit Log** dialog box



The diagnostic information has been retrieved and saved successfully and can be sent to Utimaco for further analysis.



To change the audit log configuration for showing more detailed logging information than provided by default, you must log in to the CryptoServer with at least authentication status 22000000, click the **Manage** menu and select **Audit Log Settings**.

8.1.2 Gathering Diagnostic Information with csadm

Perform the following csadm commands in a command prompt/shell and save the output in a text file to send it to the manufacturer Utimaco for problem analysis:

```
csadm [Dev=<device>] GetState
csadm [Dev=<device>] GetBootLog
csadm [Dev=<device>] GetAuditLog
csadm [Dev=<device>] GetInfo
csadm [Dev=<device>] GetBattState
csadm [Dev=<device>] ListFiles
csadm [Dev=<device>] ListFirmware
csadm [Dev=<device>] ListUser
csadm [Dev=<device>] GetTime
csadm [Dev=<device>] <Authentication> MBKListKeys
```

For csadm versions < 2.5.3, the command

```
csadm [Dev=<device>] <Authentication> MBKListKeys
```

does not need any authentication. In this case, replace this command by the following one:

```
csadm [Dev=<device>] MBKListKeys
```

8.1.3 Gathering Diagnostic Information with CryptoServer LAN

If you have a CryptoServer LAN, you can use the syslog and csxlan.log logfiles to help you with problem analysis. You can find these logfiles on the CryptoServer LAN in the `/var/log/` directory.

- All events relating to the system are recorded in the syslog.
- All events relating to the central control process are recorded in the log.

8.1.4 Gathering Diagnostic Information during the Boot Process

During the startup of SMOS and other firmware modules the CryptoServer writes log messages to its log interface:

To output the log messages of the CryptoServer CSe and the CryptoServer Se-Series Gen2 a special USB-To-Serial Adapter (Prolific PL2303) has to be connected to one of the two USB

ports available. The CryptoServer CSe and the CryptoServer Se-Series Gen2 do not have a serial port.

To watch the log messages on a PC a terminal program (for example, csadm tool) has to be connected to the USB-To-Serial Adapter (115000 baud, 8 data bits, 1 stop bit, no parity).



For every error or warning that occurs during the startup of SMOS, an appropriate message is output. Additionally, the log messages contain a list of all firmware modules that have been found by SMOS, and the information whether these modules could have been started successfully or not.



If the boot process is stopped due to a fatal error, connecting a terminal to the log interface may be the only way to get information about the problem.

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

8.2 Problem Analysis

8.2.1 Alarms

An alarm state is caused by certain extraordinary physical circumstances.

Two different kinds of alarms are possible:

- Temporary alarms. These alarms can be reset.
- Permanent alarms. These alarms cannot be reset.

Permanent alarms occur in case of a damage of the inner or outer tamper-protecting foil, whereas usually all other possible alarms are temporary.

Abbreviation	Description
Temp_low	Temperature too low (see Temperature Monitoring (p. 231))
Temp_high	Temperature too high (see Temperature Monitoring (p. 231))

Abbreviation	Description
	Voltage too high (CryptoServer-internal battery)
Pow_low	Voltage too low (CryptoServer-internal battery)
In_foil	Inner tamper-protecting foil damaged. This alarm reason is only relevant for CryptoServer CSe-Series.
Out_foil	Outer tamper-protecting foil damaged. This alarm reason is only relevant for CryptoServer CSe-Series.
ext_Erase	External deleting / clearing done
inval_MK	Invalid (corrupted) Master Key (reason for this is usually an empty battery, see below)
Power failed	Sensor controller without power
Sensory Controller failed	No reaction from sensor controller

Table 73: Possible alarm reasons

See [An Alarm and Its Consequences \(p. 38\)](#) for further details.

8.2.2 CryptoServer LAN does not Boot

Under some circumstances it is possible that the CryptoServer LAN does not boot and the mode Offline is displayed in the display.

Possible cause 1: Error in the file system

Due to an error in the file system, the current boot partition cannot be booted. One possible reason for this is that the flash memory has failed or that the number of write cycles has been exceeded.

Possible solution 1: Repair file system

1. Connect a keyboard and a screen to the CryptoServer LAN.
2. If there really is an error in the file system, you may see a prompt, in CSLANOS versions 3.0.0 to 3.0.4 of the CryptoServer LAN, asking whether you want the file system to be repaired. Answer **yes** to this prompt.
3. Alternatively, reboot your CryptoServer LAN and select a different boot partition.

Possible cause 2: Power Supply Unit is defect

As of CSLANOS version 4.1.0 the CryptoServer LAN has two power supplies. Hardware on the CryptoServer LAN, or one or both of the power supply units, are defective.

Possible solution 2: Exchange Power Supply Unit

If only one power supply unit is defective, contact the manufacturer Utimaco to order a new one and follow the instructions in *Removing/Swapping a Power Supply Module* in the *CryptoServer LAN V5 - Operating Manual*.

If both power supply units are defective, send the CryptoServer LAN back to the manufacturer Utimaco. Before you do so, please use our online portal to open a new RMA case <https://support.hsm.utimaco.com/support/rma/new>.

Possible cause 3: No connection to PCIe Card

The CryptoServer LAN cannot communicate with the mounted CryptoServer PCIe card.

Possible solution 3: Review Error Message

Connect a keyboard and a screen to the rear side of the CryptoServer LAN and check whether an error message has been generated.

8.2.3 Firmware Package not loading correctly

Possible causes 1: Wrong Firmware Package

- You have a CryptoServer Se-Series Gen2 and have attempted to load a firmware package for the CSe-Series. The system displays an error message in this case.
- You have a CSe-Series CryptoServer and have attempted to load a firmware package for the CryptoServer Se-Series Gen2. The system displays an error message in this case.

Possible solution 1: Load correct Firmware Package

Load the appropriate firmware package into your CryptoServer. The firmware packages are provided on the SecurityServer/CryptoServer SDK product CD in `Firmware\SecurityServer-<Series>\` .

- If you have a CryptoServer Se-Series Gen2, load the SecurityServer package `SecurityServer-Se2-Series-<version>.mpkg` .
- If you have a CryptoServer CSe-Series, load the SecurityServer package `SecurityServer-CSe-Series-<version>.mpkg` .

Possible causes 2: Invalid Module Loaded

You have attempted to load a firmware package into the CryptoServer and receive the error message Invalid Package. This error message can only occur if you attempt to load a

firmware package of your own.

Possible reasons for the error message:

- You have entered an incorrect file name for the firmware package.
- The package format version is incorrect.
- The number of files specified does not match the actual number of files in the *.mpkg file.
- The entered file size does not match the actual file size.
- The .mpkg file contains no firmware module.

The following file extensions are important here:

.mmc = Module Manufacturer Container

.mtc = Module Transport Container

Possible cause 3: Self-Programmed Firmware Modules signed with no Alternative Module Signature Key loaded

You have attempted to load a firmware package containing one or more self-programmed firmware modules into the CryptoServer. The public part of the Alternative Module Signature Key has not been imported yet into the CryptoServer. Therefore, the verification of the module signature fails.

Possible solution 3: Import Alternative Module Signature Key

1. Import the public part of the Alternative Module Signature Key into the CryptoServer.

Example:

```
csadm [Dev=<device>] LogonSign=ADMIN,:cs2:cjo:USB0  
LoadAltMdlSigKey=MyMdlSig.key
```

2. Load the firmware package or only the self-signed firmware module into the CryptoServer.

Example:

```
csadm [Dev=<device>] LogonSign=ADMIN,:cs2:cjo:USB0  
LoadPkg=cs-1.2.2.0.mpkg,NoClear+NoDelete
```

or

```
csadm [Dev=<device>] LogonSign=ADMIN,:cs2:cjo:USB0 LoadFile=myMDL.mtc
```

3. Restart the CryptoServer for the new firmware to become effective.

Example:

```
csadm [Dev=<device>] Restart
```

Possible cause 4: Self-Programmed Firmware Modules signed with a wrong Alternative Module Signature Key

You have attempted to load a firmware package containing one or more self-programmed firmware modules into the CryptoServer that has/have been signed with a different Alternative Module Signature Key as the one that has been imported into the CryptoServer. Therefore, the verification of the module signature fails.

Possible solution 4: Sign Self-Programmed Firmware Modules with correct Alternative Module Signature Key

1. Sign the self-programmed firmware module with the same Alternative Module Signature Key as the one imported in the CryptoServer.

Example:

```
csadm Model=c86 MMCSignKey=c:\keys\MyMdlSig.key MakeMTC=c:\firmware\myMDL.out
```

2. Optionally, create your own firmware package containing Utimaco's standard firmware and your own firmware module.

Example:

```
csadm Pack=E:\cs\fw\fw_pkg\CS-Se2-1.2.3.4
```

The firmware package CS-Se2-1.2.3.4.mpkg is created.

3. Load your own firmware module/package into the CryptoServer.

Example:

```
csadm [Dev=<device>] LogonSign=ADMIN,:cs2:cjo:USB0 LoadPkg=CS-Se2-1.2.3.4.mpkg,NoClear+NoDelete
```

or

```
csadm [Dev=<device>] LogonSign=ADMIN,:cs2:cjo:USB0 LoadFile=myMDL.mtc
```

4. Restart the CryptoServer for the new firmware to become effective.

Example:

```
csadm [Dev=<device>] Restart
```

Possible cause 5: Self-Programmed Firmware Modules signed with correct Alternative Module Signature Key still fails

You have attempted to load a firmware package containing one or more self-programmed firmware modules into the CryptoServer. However, the verification of the module signature fails even though you have loaded the Alternative Module Signature Key used to sign your self-programmed firmware module.

Possible solution 5: Generate a new Alternative Module Signature Key and re-sign Self-Programmed Firmware Modules

Proceed as follows:

1. Generate a new Alternative Module Signature Key.

Example:

```
csadm KeyType=RSA GenKey=E:\cskeys\altMDLsig.key,2048,altMDLsigKey
```

2. Load its public part into the CryptoServer.

Example:

```
csadm [Dev=<device>] LogonSign=ADMIN,:cs2:cjo:USB0 LoadAltMdlSigKey= E:\cskeys\altMDLsig.key
```

3. Sign the self-programmed firmware module with the new Alternative Module Signature Key.

Example:

```
csadm Model=c86 MMCSignKey=E:\cskeys\altMDLsig.key MakeMTC=:\firmware\myMDL.out
```

4. Optionally, create your own firmware package containing Utimaco's standard firmware and your own2 firmware module.

Example:

```
csadm Pack=E:\cs\fw\fw_pkg\CS-Se2-1.2.3.4
```

5. Load your own firmware package into the CryptoServer.

Example:

```
csadm [Dev=<device>] LogonSign=ADMIN,:cs2:cjo:USB0 LoadPkg=CS-Se2-1.2.3.4.mpkg,ForceClear
```

or

```
csadm [Dev=<device>] LogonSign=ADMIN,:cs2:cjo:USB0 LoadFile=myMDL.mtc
```

6. Restart the CryptoServer for the new firmware to become effective.

Example:

```
csadm [Dev=<device>] Restart
```

8.2.4 Problem Analysis for CSP/CNG Provider 1.x and 2.x

You may find useful information to help you if there are problems with CSP and CNG in the `cs2cng.log` logfile.

8.2.4.1 Logfile for CSP/CNG Provider 1.x

Up to SecurityServer/CryptoServer SDK 4.10, the CSP/CNG Provider 1.x has been provided.

By default, errors and warnings are recorded in the logfile.

1. Click **Start > All Programs > Utimaco > CryptoServer > CSP Configuration**.
→ The **CryptoServer CSP Configuration, Version <1.x.x>** dialog box opens.
2. Click **Settings**.
3. Click **View Log** button if you want to view the log entries, or click **>>** if you want to change the directory to which the logfile is saved.
4. Click a different log level to enable it, and click the **Apply** button if you want to change the default setting.
5. Click **OK**.
→ The **CryptoServer CSP Configuration, Version <1.x.x>** dialog box closes.

8.2.4.2 Logfile for CSP/CNG Provider 2.x

As from SecurityServer/CryptoServer SDK 4.10 the CSP/CNG Provider 2.x is provided.

The CSP/CNG Provider 2.x is configured with the `cs_cng.cfg` configuration file. The storage location for the configuration file is defined in the system environment variable `CS_CNG_CFG`, which is by default `C:\ProgramData\Utimaco\CNG`.

The logging along with other important settings is configured in the `cs_cng.cfg` configuration file. You can individually set the location for storing the logfile (`Logpath`) and the log level (`Logging`). By default, the log level setting is `Logging = 0`, i.e. no logging information is written. You can find the `cs2cng.log` logfile, by default, in the `C:\ProgramData\Utimaco\CNG\log` directory.

For detailed information about CSP/CNG 1.x and 2.x configuration, see *CryptoServer CSP and CryptoServer Key Storage Provider 1.x and 2.x Manual for System Administrators* provided on the delivered product CD in the `\Documentation\Administration Guides` directory.

8.2.5 Problem Analysis for EKM

You may find useful information to help you if there are problems with Extensible Key Management (EKM) in the EKM logfile. By default, the Utimaco EKM API generates log entries for log level 3.

You will find an example configuration file `cssqlekm.cfg` in the following directory on your SQL Server: `C:\ProgramData\Utimaco\EK M`

1. Use a text editor to open the configuration file.
The top part of the `cssqlekm.cfg` configuration file looks like this on your SQL Server:


```
This is a sample configuration file
# path to logfile
LogFile = C:/ProgramData/Utimaco/EKM/cssqlekm.log

# loglevel
LogLevel = 3
```

In the following table you will find an overview of the different log levels and their individual meanings.

0 = NONE	No log entries are generated.
1 = ERROR	Errors are recorded in the log file.
2 = WARNING	Warnings and errors are recorded in the log file. Contains log level 1.
3 = INFO	Information, warnings and errors are recorded in the log file. Contains log level 1 and 2.
4 = TRACE	All available information is recorded in the log file. Contains log level 1, 2 and 3.

If you want to enable a different log level, you must enter the appropriate number for the required log level value for LogLevel.



We recommend that you only enable logging from the Utimaco EKM API with log level 4 (TRACE) for the purpose of problem analysis and then to use the default setting (LogLevel = 3) again.

2. Save the `cssqlekm.cfg` and close your text editor.
3. Restart your SQL-Server.

8.2.6 Problem Analysis for Java-based GUI Applications

If you start a Java-based application (for example, CryptoServer Administration Tool (CAT) or P11CAT) on a Linux computer and an error according to the examples

```
java.lang.UnsatisfiedLinkError: /tmp/jcsapi6247226668792798062.so: /tmp/
jcsapi6247226668792798062.so: failed to map segment from shared object:
Operation not permitted
```

or

```
Caused by: java.lang.UnsatisfiedLinkError: /tmp/
jcsapi7418373955164564681.so: /tmp/jcsapi7418373955164564681.so: Fehler
beim Mappen des Shared Objects
```

occurs, the `XDG_RUNTIME_DIR` environment variable must be created and set to, for example, `/var/run` or `/run`. Then restart the Java-based application.

8.2.7 Problem Analysis for PKCS#11

As from SecurityServer/CryptoServer SDK 3.2 product CD the PKCS#11 R2 (as from SecurityServer/CryptoServer SDK 4.40.0 product CD the PKCS#11 R3) implementation is supplied.

You may find useful information to help you solving any problems with PKCS#11 if you enable logging for PKCS#11. The system records information, errors and warnings in the `cs_pkcs11_R3.log` logfile. By default, no log entries are generated by the Utimaco PKCS#11 R3 API. This must be enabled in the `cs_pkcs11_R3.cfg` configuration file.



We recommend enabling logging for the Utimaco PKCS#11 R3 API for the purpose of problem analysis. After it is enabled, log entries will be generated from then on. This can result in considerable volumes of data.



The Utimaco PKCS#11 R3 API only generates log entries if PKCS#11 R3 is configured correctly.

On a computer with a Windows operating system, you will find the `cs_pkcs11_R3.cfg` configuration file after the installation of the CryptoServer host software here, by default:

```
C:\ProgramData\Utimaco\PKCS11_R3\
```

1. Use a text editor to open the `cfg` file.
2. To enable logging for Utimaco's PKCS#11 R3 interface, set the log level you require in the Global section by configuring the `Logpath` and `Logging` entry as shown in the following example:

```
[Global]
# Path to the logfile (name of logfile is attached by the API)
# For unix:
#Logpath = /tmp
# For windows:
Logpath = C:/tmp

# Loglevel (0 = NONE; 1 = ERROR; 2 = WARNING; 3 = INFO; 4 = TRACE)
Logging = 3
# Maximum size of the logfile in bytes (file is rotated with a backup file
if full)
Logsize = 10mb
```

The following table shows an overview of the different log levels and their meanings.

Name	Level	Description
NONE	0	No logging output will be produced (default)
ERROR	1	Log errors of the CryptoServer PKCS#11 library and CryptoServer firmware modules
WARNING	2	Log errors and warnings of the CryptoServer PKCS#11 library and CryptoServer modules
INFO	3	Log errors and warnings of the CryptoServer PKCS#11 library and CryptoServer firmware modules. Additionally, information of the CryptoServer PKCS#11 library will be logged.
TRACE	4	Log errors, warnings and information of the CryptoServer PKCS#11 library and CryptoServer firmware modules. Additionally, trace output like function calls will be logged.

Table 74: Logging levels

3. Save the cfg configuration file and close your text editor.



Delete the logfile as soon as it is no longer needed.

8.2.8 Problems with the Network



If the reason for a problem with the Network is not immediately apparent, you can view the `syslog` and `csxlan.log` log files, which may contain information that helps you resolve the problem, see M011-0008-en Gathering Diagnostic Information with CryptoServer LAN.

Possible cause 1: Internet Protocols not supported

The CryptoServer LAN cannot be addressed over the network.

The CryptoServer LAN with operating system CSLANOS version 4.2.0 and later support the Internet Protocols IPv4 and IPv6.

Possible solution 1: Correct network settings on the LAN device

1. Use the menu options on the CryptoServer LAN to assign an IP address for the device.
2. Use the menu options on the CryptoServer LAN to assign the IP address of the default gateway.
3. Check whether you have connected the network cable to the network port connection (eth0 or eth1) for which you have specified an IP address.
4. Use the menu options on the CryptoServer LAN to enable the SSH daemon, if you want to configure the CryptoServer LAN via an SSH connection.
5. If you cannot address the CryptoServer LAN over the network after these actions, you can use the menu options on the CryptoServer LAN to send a PING to your Admin PC or from your Admin PC to your CryptoServer LAN.

For detailed descriptions of the CryptoServer LAN menu navigation, see *CryptoServer LAN V5 - Operating Manual*.

Possible cause 2: Standard port 288 cannot be reached

The standard port 288 cannot be reached. The possible reasons for this include problems with the routing, the firewall or the address conversion in IT networks (NAT).

Possible solution 2: Release port 288

Check the firewall rules. The default protocol is TCP.

Possible cause 3: Maximum number of permitted connections has been reached

The maximum number of permitted connections to the CryptoServer LAN (MaxConnections) has been reached. For the CryptoServer LAN with CryptoServer version $\geq 3.2.0$, we have set MaxConnections, to 256. For CryptoServer LAN version $\geq 4.5.4$, we have set MaxConnections, to 4100.

Possible solution 3: Increase number of permitted connections

You must increase the maximum number of permitted connections (MaxConnections) in the `csxlan.conf` file. The following below is based on making changes to the `csxlan.conf` file via SSH-access with WinSCP for Windows.



The default system configuration of CSLANOS version 4.5.x and higher prohibits remote login for the root user via SSH connection.

To enable SSH-login for the user root, you should edit the configuration file for the SSH daemon `/etc/ssh/sshd_config` to change the default setting `PermitRootLogin no` to `PermitRootLogin yes`. Afterwards, the SSH daemon has to be reloaded for the setting to become effective (`/etc/init.d/sshd reload`).

Data required for SSH access:

Computer name or IP address	Name of the CryptoServer LAN/IP address of the CryptoServer LAN
Port number	22
User name	root
Password	utimaco

1. Start your SCP client (e.g. WinSCP) and open the `/etc` directory.
In this directory, you find the `csxlan.conf` (`/etc/csxlan.conf`).
2. Right-click on the file and select **Edit** from the context menu.
→ The `csxlan.conf` file opens.
3. For the CryptoServer LAN with CryptoServer version $\geq 3.2.0$ and CryptoServer LAN version $< 4.5.4$, increase the value set for the `MaxConnections=256` entry to the value you require. The value that you set here for `MaxConnections` should not be greater than 1000, as this can cause performance problems under some circumstances.
4. Save and close the `csxlan.conf` file.
5. Reboot your CryptoServer LAN so that the changed `csxlan.conf` file is used.

8.2.9 The CryptoServer hangs-up during the Boot Phase

If the CryptoServer hangs-up during the boot phase, it cannot be accessed anymore over the command interface.

Possible causes:

- One of the basic firmware modules has been deleted.
- Buggy or incompatible software has been loaded.

If the CryptoServer hangs-up 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 basic firmware modules or loaded buggy or incompatible software. In such a situation, the backup copies of the basic firmware modules (i.e., all system firmware modules `*.sys`) can be started to get the CryptoServer in Operational Mode again.

Possible Solution:

The backup copies of the basic firmware modules (i.e., all system firmware modules `*.sys`) can be started to get the CryptoServer in Operational Mode again.

Perform the following steps:

1. Start the CryptoServer.
2. Run the `ResetToBL` command.
3. Run the `RecoverOS` command.
After that, only the system firmware modules are running, which is sufficient to administrate the CryptoServer.
4. The bad/missing firmware can now be replaced/loaded using the normal administration command `LoadFile`.



The CryptoServer can be put again in normal Operational Mode (running all firmware modules `*.msc`) with a `Restart` command.



See also section *Commands for Administration in CryptoServer - csadm Manual* for command details.

8.2.10 The CryptoServer is in Maintenance Mode

Normally, the CryptoServer only switches to Maintenance Mode after an alarm and remains in that mode until the alarm has been reset by a CryptoServer user with system administration permissions. However, there can also be other causes for the Maintenance Mode.

Possible cause: Firmware Modules not started correctly

When the CryptoServer was booted or restarted (reset), one of the essential firmware modules could not be started. In this case, it is not possible to operate the CryptoServer correctly and it changes to Maintenance Mode.

Possible solution 1: Restart the CryptoServer

Restart the CryptoServer:

- Using CAT: Click the **Manage** menu and select **Reboot CryptoServer**.
- Using csadm: In a command prompt/shell type `csadm Dev=<device> Restart` and press **ENTER**.

Possible solution 2: Reinstall Firmware Packages

If the CryptoServer is not in Operational Mode after a restart, follow these steps:

1. View the boot log file to determine what was started when the CryptoServer was switched on, and whether the essential firmware modules were initialized or not.
 - Using CAT: Click the **Show** menu and then click **Boot Log**. The boot log entries appear in the CAT information field.
 - Using csadm: In a command prompt/shell type `csadm Dev=<device> GetBootLog` and press **ENTER**.
The version of CryptoServer's operating system SMOS (Security Module Operating System) is shown and the module state is displayed as started.
The essential firmware modules should be listed in the Bootlog with their module ID (in hexadecimal format), module name and the comment `initialized successfully`.
If one of the firmware modules was not initialized successfully (e.g., due to

dependencies to other firmware modules that could not be fulfilled), or is missing, the CryptoServer cannot be operated correctly.

2. Reinstall the firmware package from the product CD and ensure that all sensitive data (user and key databases) including the Master Backup Key is deleted in the CryptoServer.



Make sure to have prepared recent backups of all sensitive data and the MBK stored on the CryptoServer. Otherwise, they will be lost after loading the SecurityServer firmware package as mentioned above and you will not be able to import your sensitive data into the CryptoServer again. See the *CryptoServer – CAT Manual* and the *CryptoServer - csadm Manual* for detailed information about how to back up the CryptoServer databases storing the sensitive data.

- Using CAT: For detailed instructions see *Installing/Updating the Firmware* in the *CryptoServer - CAT Manual*.
- Using csadm: Execute the following csadm command: `csadm Dev=<device> <Authentication> LoadPkg=<package>,ForceClear`
For details about the syntax of the `csadm LoadPkg` command see the csadm in-tool help or *LoadPkg* in the *CryptoServer - csadm Manual*.

3. Import the database backup into your CryptoServer by using CAT, see *Restoring Databases* in the *CryptoServer - CAT Manual*, or by using csadm, see *RestoreDatabase* in the *CryptoServer - csadm Manual*.

8.3 Providing Support Information for a CryptoServer PCIe

If you have a CryptoServer PCIe card and want to contact Utimaco's support, get the following information ready for us:

- Your customer or company name
- Give an exact description of the problem.
- Can the problem be reproduced?
- The version number of the product CD or of the SecurityServer package used.
The version number of the product CD is identical to the version number of the SecurityServer package that is supplied on the product CD.

- The Diagnostic Information, saved as a `.txt` file, see [Gathering Diagnostic Information](#) (p. 263).

8.4 Providing Support Information for a CryptoServer LAN

If you have a CryptoServer LAN and want to contact Utimaco's support, get the following information ready for us:

- The CryptoServer LAN's serial number.
 - You can find the CryptoServer LAN's serial number via the menu options on the front panel of the CryptoServer LAN.
CSLAN admin. > CSLAN Info > Show Version
 - Alternatively, you can find the serial number on the right-hand side of the CryptoServer LAN.
 - If you can only access the CryptoServer LAN remotely, execute the `csadm CSLGetSerial` command on your administration computer.
`csadm [Dev=<device>] CSLGetSerial`
For further details about this `csadm` command, read the corresponding chapter in the *CryptoServer - csadm Manual*.
- The CSLANOS and `dsp_admin` version number.
 - You can find out the `dsp_admin` version number via the menu options on the front panel the CryptoServer LAN.
CSLAN admin. > CSLAN Info > Show Version
 - If you can only access the CryptoServer LAN remotely, execute the `csadm CSLGetVersion` command on your administration computer.
`csadm [Dev=<device>] CSLGetVersion`
For further details about this `csadm` command, read the corresponding chapter in the *CryptoServer - csadm Manual*.
- Give an exact description of the problem.
- Can the problem be reproduced?
- The Diagnostic Information, saved as a `.txt` file.
To save the Diagnostic Information as a `.txt` file on your computer, see [Gathering Diagnostic Information](#) (p. 263) Additionally, provide the logfiles, see [Gathering Diagnostic Information with CryptoServer LAN](#) (p. 265).



The default system configuration of CSLANOS version 4.5.x and higher prohibits remote login for the root user via SSH connection.

To enable SSH-login for the user root, you should edit the configuration file for the SSH daemon `/etc/ssh/sshd_config` to change the default setting `PermitRootLogin no` to `PermitRootLogin yes`. Afterwards, the SSH daemon has to be reloaded for the setting to become effective (`/etc/init.d/sshd reload`).

9 Contact Address for Support Queries

If an error occurs while operating the CryptoServer, read [Troubleshooting \(p. 263\)](#) to solve it.

If you have any further questions on CryptoServer, feel free to contact us.

You can reach us from Monday to Friday, 09.00 a.m. to 05.00 p.m., Central European Time (CET).

Utimaco IS GmbH
Germanusstr. 4
52080 Aachen
Germany

RMA Query

If you need to send the device back to Utimaco IS GmbH, please open a new RMA case (Return Merchandise Authorization). We request that you use the following web address. RMA cases cannot be opened by email or phone.

<https://support.hsm.utimaco.com/support/rma/new>

Other Support Queries

- Mail (preferred contact method)
support@utimaco.com³
Attach the diagnostic information to your email.
- Web portal
<https://support.hsm.utimaco.com/support/cases/new/>
The diagnostic information will be requested in our response if necessary.
- By phone
AMERICAS +1-844-UTIMACO (+1 844-884-6226)
EMEA +49 800-627-3081
APAC +81 800-919-1301
The diagnostic information will be requested in our response if necessary.

³ <mailto:support@utimaco.com>

10 References

Reference	Title/Company	Doc. -No.
[CSADMIN]	CryptoServer – csadm Manual/ Utimaco IS GmbH.	2009-0003
[CSCSe-OM]	CryptoServer PCIe CSe-Series Operating Manual/ Utimaco IS GmbH.	M013-0002-en
[CSSe2-OM]	CryptoServer PCIe Se-Series Gen2 Operating Manual/Utimaco IS GmbH.	M015-0001-en
[CSCSP-CNG]	CryptoServer - CryptoServer CSP and CNG Key Storage Provider/Utimaco IS GmbH.	2008-0002
[CSFIPS-AdmGuide]	CryptoServer – Administrator's Guide for CryptoServer Se-Series Gen2/CSe in FIPS Mode/ Utimaco IS GmbH	2011-0002
[CSLAN5]	CryptoServer LAN V5 – Administration Manual/ Utimaco IS GmbH.	2018-0010
[CS_CXIttool]	CryptoServer – cxitool Manual / Utimaco IS GmbH	2018-0003
[CS_PKCS11CAT]	CryptoServer – PKCS#11 P11CAT Manual/ Utimaco IS GmbH.	M013-0001-en
[CS_PKCS11DEV]	PKCS#11 R3 Developer Guide/Utimaco IS GmbH.	2012-0007
[CS_PKCS11T2]	CryptoServer –PKCS#11 p11tool2 Reference Manual/Utimaco IS GmbH.	2012-0014
[CSTrSh]	CryptoServer Troubleshooting/Utimaco IS GmbH.	M011-0008-en
[SM2]	SM2 Digital Signature Algorithm, draft-shen-sm2- ecdsa-02, S.Chen / X.Lee, Chinese Academy of Science, February 2014	