NGINX

1.23.1

Integration Guide

CryptoServer HSM

SecurityServer 4.45.3

utimaco®

Imprint

Copyright 2026	Utimaco IS GmbH Krefelder Straße 220 52070 Aachen Germany
Phone	AMERICAS +1-844-UTIMACO (+1 844-884-6226) EMEA +49 800-627-3081 APAC +81 800-919-1301
Internet	https://support.hsm.utimaco.com/
e-mail	support@utimaco.com
Document Version	1.0.0
Date	2026-07-21
Status	PUBLISHED
Document No.	IG-2026-0077

Table of Contents

1 Introduction

This guide is part of the information and support provided by Utimaco. Additional documents produced to support your Utimaco SecurityServer product can be found in the document directory of the Utimaco SecurityServer product bundle. All Utimaco SecurityServer product documentation is available on Utimaco's web site at *https://utimaco.com/*.

1.1 About This Guide

This guide provides an integration explaining how to integrate an Utimaco CryptoServer Hardware Security Module (HSM) with NGINX. Utimaco HSM is used to secure the private keys for SSL certificate and offload cryptographic operations on the HSM.

1.2 Target Audience for This Guide

This guide is intended for administrators of NGINX and of Utimaco HSMs.

1.3 Document Conventions

The following conventions are used in this guide:

Convention	Use	Example
Bold	Items of the Graphical User Interface (GUI), e.g., menu options	Select **Details** and click on **Properties** button
`Monospaced`	Code that is given for explanation or as an example, file paths	`certreq.exe -new request.inf IISCertRequest.csr`
Italic	References and important terms	Operating system listed in *Tested Versions*

Table 1: Document conventions

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

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

Here you will find additional notes or supplementary information.

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

1.4 Abbreviations

The following abbreviations are used in this guide:

Abbreviation	Meaning
API	Application Programming Interface
CA	Certificate Authority
CD	Compact Disc
CSADM	CryptoServer Command-line Administration Tool
GUI	Graphical User Interface
HSM	Hardware Security Module
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure

Abbreviation	Meaning
IMAP	Internet Message Access Protocol
IP	Internet Protocol
LAN	Local Area Network
PCIe	PCI Express Interface
PCRE	Perl Compatible Regular Expressions
PIN	Personal Identification number
PKCS#11	Public-Key Cryptography Standard #11
POP3	Post Office Protocol version 3
RSA	Rivest-Shamir-Adleman
SMTP	Simple Mail Transfer Protocol
SO	Security Officer
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

Abbreviation	Meaning
URL	Uniform Resource Locator

Table 2: List of abbreviations

2 Overview

2.1 NGINX

NGINX is an open-source software for web serving, reverse proxying, caching, load balancing, media streaming, and more. It started out as a web server designed for maximum performance and stability. In addition to its HTTP server capabilities, NGINX can also function as a proxy server for email (IMAP, POP3, and SMTP) and a reverse proxy and load balancer for HTTP, TCP, and UDP servers.

Libp11 provides an engine_pkcs11 that tries to fit the PKCS#11 API within the engine API of OpenSSL. That is, it provides a gateway between PKCS#11 modules and the OpenSSL engine API.

2.2 Utimaco CryptoServer HSM

CryptoServer is a hardware security module developed by Utimaco IS GmbH. CryptoServer is a physically protected specialized computer unit designed to perform sensitive cryptographic tasks and to securely manage, as well as store, cryptographic keys and data. It can be used as a universal, independent security component for heterogeneous computer systems.

3 Integration Requirements and Prerequisites

Ensure the system environment you will be using meets the following hardware and software requirements.

This guide assumes that the user has already installed and configured the required software.

3.1 Tested Versions

The integrations that have been successfully tested with the Utimaco HSM with NGINX.

Operating System	NGINX	OpenSSL	Utimaco Security Server Version	Utimaco HSM
RHEL8	Nginx 1.23.1	OpenSSL 1.1.1q	SecurityServer 4.45.3 p11tool2 from product package Utimaco SecurityServer	CryptoServer CSe-Series/Se- Series

Table 3: List of tested versions

SecurityServer 4.45.5 is not supported with NGINX.

3.2 Software Requirements

Software	Software Requirements
Nginx	Nginx 1.23.1
OpenSSL	OpenSSL 1.1.1q

Software	Software Requirements
Libp11 (Linux)	0.4.12
HSM Interface	SecurityServer PKCS#11 Provider

Table 4: List of software requirements

3.3 Hardware Requirements

Hardware	Hardware Requirements
Utimaco LAN HSM	CryptoServer CSe-Series/Se-Series LAN with firmware SecurityServer 4.45.3
Hardware	Hardware Requirements
Utimaco PCI-e HSM	CryptoServer CSe-Series/Se-Series PCI-e with firmware SecurityServer 4.45.3

Table 5: List of hardware requirements

Set up an account on the Utimaco support portal and request download access at the following URL: *https://support.hsm.utimaco.com/*.

3.4 3.4 Prerequisites

Before you begin, please ensure that:

- The CryptoServer is set up and configured. Refer to the CryptoServer documentation to set up the HSM.
- The CryptoServer Default Admin has been replaced with a new admin user.
- The MBK has been created and stored onto each HSM. Refer to the CryptoServer documentation to set up the MBK.

- The operating system is listed in Tested Versions.
- The SecurityServer is listed in Tested Version.
- The PKCS#11 library is set up and configured as per the environment. Refer the CryptoServer documentations to set up and configure the PKCS#11 library for CryptoServer.
- Ypu familiarize yourself with the Nginx and OpenSSL documents and setup process.
- There is a user with admin privileges as it is required for installing few packages on to the NGINX server.

4 Installing and Configuring Utimaco SecurityServer Software

4.1 Download and Install Utimaco Software

If you have not already done so, please create and request an Utimaco Support Portal Account. This will allow you to download the software components needed for this installation.

1. Copy the downloaded software at the appropriate location on the NGINX Server. Create `utimaco` folder under `/opt` directory and further create 2 directories `/opt/utimaco/bin` and `/opt/utimaco/lib`.

›_ Console

```
# mkdir -p  /opt/utimaco/bin

# mkdir /opt/utimaco/lib
```

2. Copy pkcs11 library file `libcs_pkcs11_R3.so` from Utimaco CryptoServer software to the `/opt/utimaco/lib` directory and make the file executable.

›_ Console

```
# cp ~/path_to_application_folder/lib/ libcs_pkcs11_R3.so

/opt/utimaco/lib
```

3. Copy the `csadm` and `p11tool2` files from Utimaco CryptoServer software to `/opt/utimaco/bin` directory and make both the files executable.

›_ Console

```
# cd ~/path_to_application_folder

# cp csadm p11tool2 /opt/utimaco/bin

# chmod  +x /opt/utimaco/bin/csadm /opt/utimaco/bin/p11tool2
```

4.2 CryptoServer PKCS#11 Configuration

1. Create the directory `/etc/utimaco`. Locate the Utimaco PKCS#11 configuration file in your SecurityServer directory, `Linux/x86-64/Crypto_APIS/PKCS11_R3/sample`. Copy the Utimaco PKCS#11 configuration file `cs_pkcs11_R3.cfg` into `/etc/utimaco` directory.

›_ Console

```
# mkdir /etc/utimaco

# cd <install directory>/Software/Linux/x86-

64/Crypto_APIs/PKCS11_R3/sample # cp cs_pkcs11_R3.cfg /etc/utimaco

# cd /etc/utimaco
```

2. Edit the `cs_pkcs11_R3.cfg` file and make the appropriate changes to the file.

cs_pkcs11_R3.cfg

```
[Global]

# For unix:

Logpath = /tmp

# Loglevel (0 = NONE; 1 = ERROR; 2 = WARNING; 3 = INFO; 4 = TRACE)  Logging = 1

Keepalive = true

MultiInitReturnsCKR_OK = true

# Set the Device to connect with

[CryptoServer] # Device specifier  Device = <HSM_IP>
```

For more information regarding the commands and command parameters please check the Utimaco CryptoServer documentation. The device may be a CryptoServer (PCIe or LAN) device. The device line will follow one of these patterns, based on the HSM form-factor:

Device = 288@<HSM IP address> Hardware (LAN) HSM

OR

Device = /dev/cs2.0 Hardware (PCIe) HSM

To make your testing easier, it would be good to enable the PKCS#11 log file. That can be enabled by editing the **Logging Loglevel**. Set the **LogPath** and **Logging Loglevel** to 1. For testing you may want to increase it to [1].

The added **LogPath** points to a writable directory, not to a file.

If you encounter problems, check the log file named `cs_pkcs11_R3.log` in the **LogPath** defined directory. When you are done testing, you should change Logging to 1 or 2. This will limit the logging to only critical and important messages.

4.3 Create SO User and Initialize a Slot

You should initialize a slot with a custom label using p11tool2. First, using p11tool2 create the SO or Security Officer and then using p11tool2 command initialize the Slot that you want to use, and the slot user as shown below.

›_ Console

```
# ./p11tool2 slot=<slot_no> Label=<token_label> Login=ADMIN,ADMIN.key
InitToken=<SO_PIN>

# ./p11tool2 slot=0 LoginSO=<SO_PIN> InitPin=<CryptoUser_PIN>
```

5 Integrating Nginx with Utimaco HSM

5.1 Installing OpenSSL

1. (Optional) It is recommended to update the system with latest security patch.

›_ Console

```
# yum update -y
```

If you are using existing or pre-installed openssl then skip step 2,3, 4 and 5.

2. Install dependent packages for openssl.

›_ Console

```
# yum install make gcc perl pcre-devel zlib-devel
```

3. Download the latest version of openssl on Linux machine from *https://www.openssl.org*.

›_ Console

```
# wget https://www.openssl.org/source/openssl-1.1.1q.tar.gz
```

4. Extract the downloaded file.

›_ Console

```
# tar xvf openssl-1.1.1q.tar.gz
```

5. Go to `openssl` directory and run the following commands to build and install openssl.

›_ Console

```
# cd openssl-1.1.1q

# ./config --prefix=/usr/local/openssl

# make

# make test

# make install

# export LD_LIBRARY_PATH=/usr/local/openssl/lib:$LD_LIBRARY_PATH

# export PATH=/usr/local/openssl/bin:$PATH

# openssl version -a
```

```
[root@Nginx ~]# openssl version -a
OpenSSL 1.1.1q  5 Jul 2022
built on: Thu Sep  1 06:50:40 2022 UTC
platform: linux-x86_64
options:  bn(64,64) rc4(16x,int) des(int) idea(int) blowfish(ptr)
compiler: gcc -fPIC -pthread -m64 -Wa,--noexecstack -Wall -O3 -DOPENSSL_USE_NODELETE -DL_ENDIAN -DOPENSSL_PIC -DOPENSSL_CPUID_
OBJ -DOPENSSL_IA32_SSE2 -DOPENSSL_BN_ASM_MONT -DOPENSSL_BN_ASM_MONT5 -DOPENSSL_BN_ASM_GF2m -DSHA1_ASM -DSHA256_ASM -DSHA512_AS
M -DKECCAK1600_ASM -DRC4_ASM -DMD5_ASM -DAESNI_ASM -DVPAES_ASM -DGHASH_ASM -DECP_NISTZ256_ASM -DX25519_ASM -DPOLY1305_ASM -DND
EBUG
OPENSSLDIR: "/usr/local/openssl/ssl"
ENGINESDIR: "/usr/local/openssl/lib/engines-1.1"
Seeding source: os-specific
[root@Nginx ~]#
```

Figure 1 : OpenSSL version output

After rebooting the server, export the library path every time.

export

LD_LIBRARY_PATH=/usr/local/openssl/lib:$LD_LIBRARY_PATH # export PATH=/usr/local/openssl/bin:$PATH

5.2 Installing Libp11

1. Download the latest libp11 package from *Releases · OpenSC/libp11 · GitHub*.

›_ Console

```
# wget https://github.com/OpenSC/libp11/releases/download/libp110.4.12/
libp11-0.4.12.tar.gz
```

2. Extract the file.

›_ Console

```
# tar -xvf libp11-0.4.12.tar.gz
```

3. Go to `libp11` directory, build and install libp11using the following commands.

›_ Console

```
# cd libp11-0.4.12

# ./configure OPENSSL_CFLAGS="-I/usr/local/openssl/include/openssl"

OPENSSL_LIBS="-L/usr/local/openssl/lib -lcrypto" prefix="/usr/local/libp11/"

# make

# make install # export LD_LIBRARY_PATH=/usr/local/openssl/lib: /usr/local/
libp11/lib/:$LD_LIBRARY_PATH
```

```
[root@Nginx libp11-0.4.12]# ./configure OPENSSL_CFLAGS="-I/usr/local/openssl/include/openssl" OPENSSL_LIBS="-L/usr/local/opens
sl/lib -lcrypto" prefix="/usr/local/libp11/"
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking target system type... x86_64-pc-linux-gnu
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /usr/bin/mkdir -p
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking whether make supports nested variables... yes
checking whether make supports nested variables... (cached) yes
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking whether gcc understands -c and -o together... yes
checking whether make supports the include directive... yes (GNU style)
checking dependency style of gcc... gcc3
checking for pkg-config... /usr/bin/pkg-config
checking pkg-config is at least version 0.9.0... yes
checking how to run the C preprocessor... gcc -E
checking for grep that handles long lines and -e... /usr/bin/grep
checking for egrep... /usr/bin/grep -E
checking for ANSI C header files... yes
checking for sys/types.h... yes
```

Figure 2 : Output of configure command for libp11

```
config.status: creating examples/Makefile
config.status: creating tests/Makefile
config.status: creating src/config.h
config.status: src/config.h is unchanged
config.status: executing depfiles commands
config.status: executing libtool commands
configure: creating src/libp11.map

libp11 has been configured with the following options:

Version:                0.4.12
libp11 directory:       /usr/local/libp11/lib
Engine directory:       /usr/lib64/engines-1.1
Default PKCS11 module:
API doc support:        no

Host:                   x86_64-pc-linux-gnu
Compiler:               gcc
Preprocessor flags:
Compiler flags:         -g -O2 -pthread
Linker flags:
Libraries:              -lpthread -ldl

OPENSSL_CFLAGS:         -I/usr/local/openssl/include/openssl
OPENSSL_LIBS:           -L/usr/local/openssl/lib -lcrypto

[root@Nginx libp11-0.4.12]#
```

Figure 3 : Output of configure command for libp11 (continued)

If you are using existing or pre-installed openssl then change the value of OPENSSL_CFLAGS and OPENSSL_LIBS to their correct path.

Make a note of "Engine Directory" path while running the configure command as the pkcs11.so file is generated inside this directory after running "make install" command.

After rebooting the server, export the library path every time.

export LD_LIBRARY_PATH=/usr/local/openssl/lib:

/usr/local/libp11/lib/:$LD_LIBRARY_PATH and # export PATH=/usr/local/openssl/bin:$PATH

5.3 Configuring OpenSSL to Use Utimaco HSM

5.3.1 Setting up Utimaco CryptoServer Library in OpenSSL Configuration File

1. Open the file `/usr/local/openssl/ssl/openssl.cnf` and enter the following line in the first line of the file.

›_ Console

```
openssl_conf = openssl_init
```

2. Enter the following lines under last section of `openssl.cnf` file.

›_ Console

```
[openssl_init] engines=engine_section


[engine_section] pkcs11 = pkcs11_section


[pkcs11_section] engine_id = pkcs11

dynamic_path = /usr/local/libp11/lib/pkcs11.so MODULE_PATH = /opt/utimaco/lib/
libcs_pkcs11_R3.so init = 0
```

Dynamic path and Module path will get changed according to the user environment.

5.3.2 Verify PKCS#11 Engine

Run the command below to verify if the OpenSSL Engine is available or not.

›_ Console

```
# openssl engine pkcs11 -t
```

```
[root@Nginx ~]# openssl engine pkcs11 -t
(pkcs11) pkcs11 engine
     [ available ]
[root@Nginx ~]#
```

Figure 4 : Verification of pkcs11 engine

5.4 Install NGINX

NGINX requires the following dependent packages:

- PCRE.
- ZLIB.

5.4.1 Installing PCRE

1. Download the `pcre` installation file.

›_ Console

```
# wget ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/pcre8.44.tar.gz
```

2. Extract the file.

›_ Console

```
# tar -zxf pcre-8.44.tar.gz
```

3. Go to the directory where the file is extracted.

›_ Console

```
# cd pcre-8.44
```

4. Run the following command to compile and install.

›_ Console

```
# ./configure

# make

# make install
```

5.4.2 Installing ZLIB

1. Download the `zlib` installation file.

›_ Console

```
# wget http://zlib.net/zlib-1.2.11.tar.gz
```

2. Extract the file.

›_ Console

```
# tar -zxf zlib-1.2.11.tar.gz
```

3. Go to the directory where the file is extracted.

›_ Console

```
# cd zlib-1.2.11
```

4. Run the following command to compile and install.

›_ Console

```
# ./configure
# make
# make install
```

5.4.3 Installing NGINX

1. Download the `zlib` installation file.

›_ Console

```
# wget https://nginx.org/download/nginx-1.23.1.tar.gz
```

2. Extract the file.

›_ Console

```
# tar -zxf nginx-1.23.1.tar.gz
```

3. Go to the directory where the file is extracted.

›_ Console

```
# cd nginx-1.23.1
```

4. Run the following command to compile and install.

›_ Console

```
# ./configure --sbin-path=/usr/local/nginx/nginx --conf-

path=/usr/local/nginx/nginx.conf --pid-path=/usr/local/nginx/nginx.pid --

with-pcre=/home/pcre-8.44 --with-zlib=/home/zlib-1.2.11 --withhttp_ssl_module --
with-stream --with-cc-opt="-I

/usr/local/openssl/include" --with-ld-opt="-L /usr/local/openssl/lib -ldl

-Wl,-rpath,/usr/local/openssl/lib"

# make

# make install
```

5. Verify that the NGINX has been installed with the correct version of openssl.

›_ Console

```
# ./nginx -V
```

```
[root@Nginx nginx]# ./nginx -V
nginx version: nginx/1.23.1
built by gcc 8.3.1 20190507 (Red Hat 8.3.1-4) (GCC)
built with OpenSSL 1.1.1q  5 Jul 2022
TLS SNI support enabled
configure arguments: --sbin-path=/usr/local/nginx/nginx --conf-path=/usr/local/nginx/nginx.conf --pid-path=/usr/local/nginx/ng
inx.pid --with-pcre=/home/pcre-8.44 --with-zlib=/home/zlib-1.2.11 --with-http_ssl_module --with-stream --with-cc-opt='-I /usr/
local/openssl/include' --with-ld-opt='-L /usr/local/openssl/lib -ldl -Wl,-rpath,/usr/local/openssl/lib'
[root@Nginx nginx]#
```

Figure 5 : Nginx version

6. After the installation is finished, start NGINX.

›_ Console

```
# ./nginx
```

```
[root@Nginx nginx]# ./nginx
[root@Nginx nginx]#
```

Figure 6 : Nginx service start

7. Open http://<nginx_server_ip> in any browser and verify that you can see nginx welcome page.

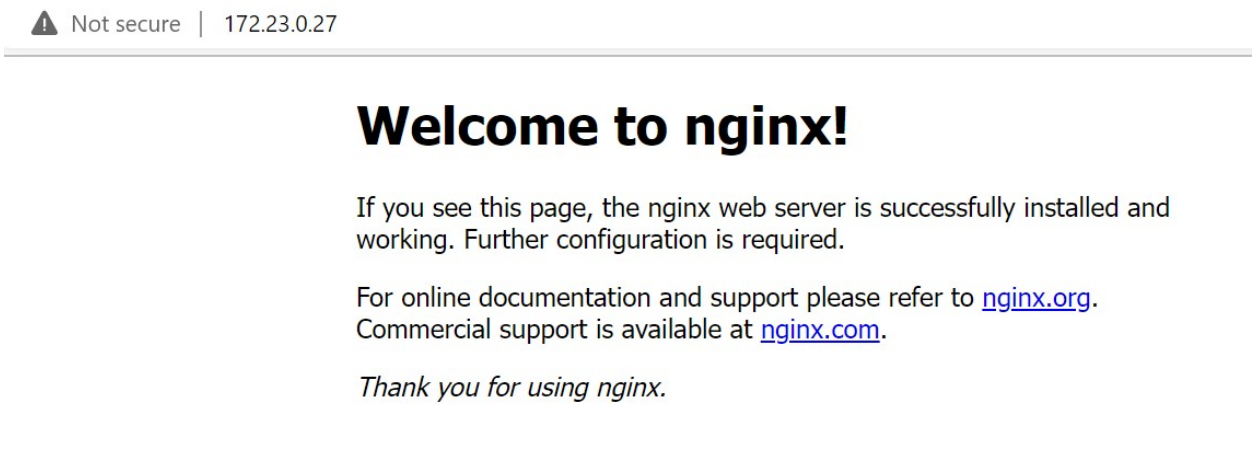

Figure 7 : Nginx Welcome Page with Http

5.5 Generating Keys and Certificate for SSL

1. Generate the RSA key using p11tool2.

›_ Console

```
# p11tool2 slot=4 LoginUser=123456

PubKeyAttr=CKA_LABEL="RSAKey",CKA_ID=0x45

PrvKeyAttr=CKA_LABEL="RSAKey",CKA_ID=0x45  GenerateKeyPair=RSA
```

2. Verify that the keys are generated onto the HSM using following command.

›_ Console

```
# p11tool2 LoginUser=<cryptouser_password> ListObjects
```

Example:

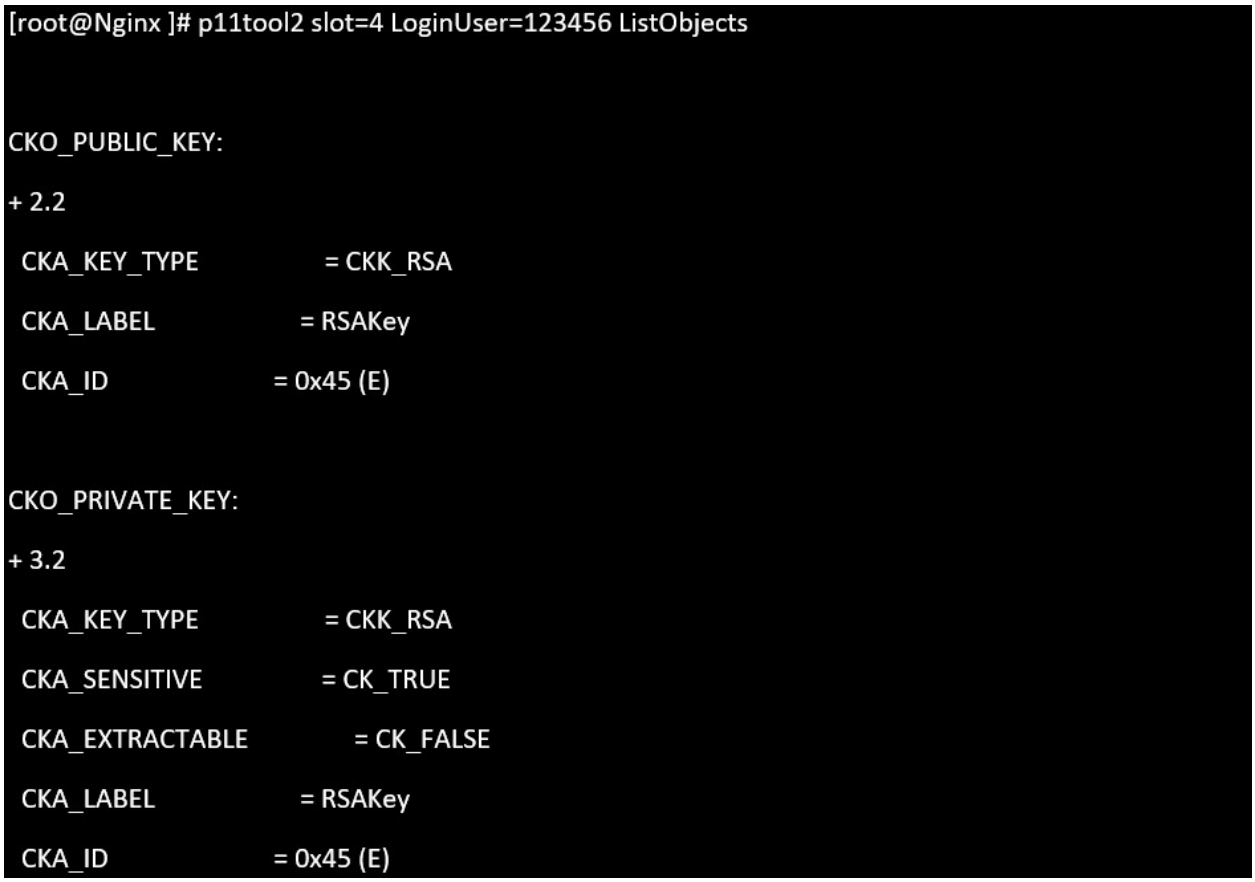

Figure 8 : List keys

3. Generate a certificate request.

›_ Console

```
# openssl req -engine pkcs11 -new -key "pkcs11:token=Nginx;object=RSAKey"
-keyform engine -out RSACSR.csr
```

Here, Nginx is the token label and RSAKey is the key on the HSM. Provide Cryptouser PIN when prompted. Also provide other required information for certificate.

```
[root@Nginx nginx]# openssl req -engine pkcs11 -new -key "pkcs11:token=Nginx;object=RSAKey" -keyform engine -out RSACSR.csr
engine "pkcs11" set.
Enter PKCS#11 token PIN for Nginx:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:CA
Locality Name (eg, city) []:Campbell
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Utimaco
Organizational Unit Name (eg, section) []:Utimaco
Common Name (e.g. server FQDN or YOUR name) []:Nginx
Email Address []:support@utimaco.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
[root@Nginx nginx]#
```

Figure 9 : Certificate request output

```
[root@Nginx nginx]# cat RSACSR.csr
-----BEGIN CERTIFICATE REQUEST-----
MIICyzCCAbMCAQAwgYUxCzAJBgNVBAYTAlVTMQswCQYDVQQIDAJDQTERMA8GA1UE
BwwIQ2FtcGJlbGwxEDAOBgNVBAoMB1V0aW1hY28xEDAOBgNVBAsMB1V0aW1hY28x
DjAMBgNVBAMMBU5naW54MSIwIAYJKoZIhvcNAQkBFhNzdXBwb3J0QHV0aW1hY28u
Y29tMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAhk2Kr3pxpBAPid9j
xZJ036/LwpAZXfZt+uR4QPH3oTgwLi4qZQQpWbm4dYFjZPWSN75RgU/Oscz7CQjK
+xM21CN2uED6Vtkl1vwRLvS5tDehQTIC3BSl4/2UMYLptHdZL6gcekzb9zsLN8nc
elmdWvdd8cnqmAEHXLnHxKMGjoxuD5uLSJnHdUxXWG4jTLKxa/O8TyHZqXuwK/NZ
2NcYTHeNUYUP17NDSbHOgpS5SukIxzpgfvJy1yKfVts8oFrGBbJSHwCobj3P63nj
Jfkl3ybaBurnozqLbqRpMROEBRblrE1CeDHAI6fzfdJnF1hFFxT3E7IYiWapFONn
UK/BLQIDAQABoAAwDQYJKoZIhvcNAQELBQADggEBAIQRfUz7bCQaxGeokmgEAL2Y
lUKQJ+mJ9M+mXBSmFwA4oJwXjmabQKgbtwfstYR9Tg4pqiRFke2lTEcq3cyNGGps
dWFD+RLPfdOBcDRK9xReTJO4kqqNeKlfWMI84/lGkPg1d0vCLXk9SgyGkpSrqJVO
XK9V1y43wsvjauW5a3jRY4/lGhq7HT9G/1m/tAP1FkG95/vHW7snyBBTn4Pg8wHd
kxjCYdCDmoyuXIhf9tiWPF2BSKt4y90GWZYgJLCjumuSBEAx+4F+VqZrFlF5d61J
HpcFpt1KQwV5bUu4rjTdzHCN/Fn6AtLoVWeQdgxjwNJFQqiJlHsCp4C2yr0p4wc=
-----END CERTIFICATE REQUEST-----
[root@Nginx nginx]#
```

Figure 10 : Content of certificate request file

4. Get this CSR signed by your CA and copy the signed certificate to NGINX server.
5. Alternatively, you can create the self-signed certificate based on the generated key.

›_ Console

```
# openssl req -engine pkcs11 -new -x509 -days 365 -key

"pkcs11:token=Nginx;object=RSAKey" -keyform engine -out RSA.cert
```

Here, `Nginx` is the token label and `RSAKey` is the key on the HSM. Provide Cryptouser PIN when prompted. Also provide other required information for certificate.

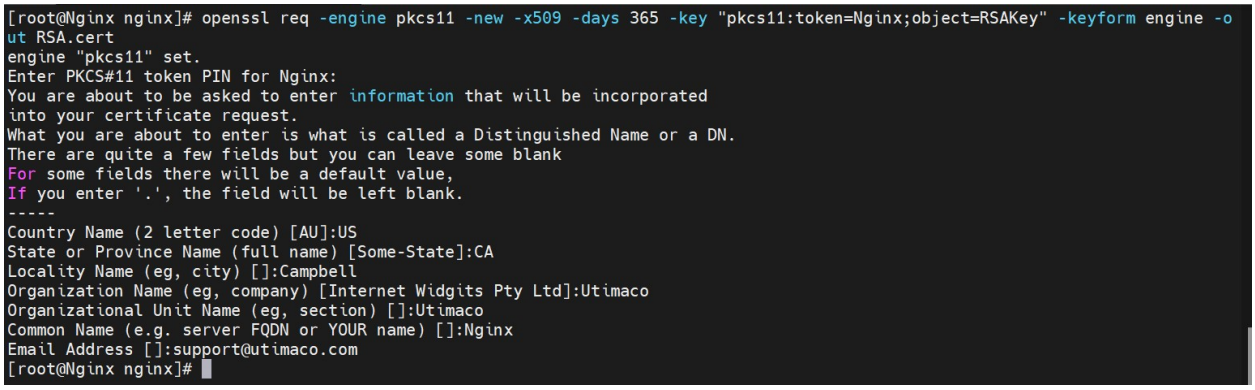

Figure 11 : Self-signed certificate generation output

```
[root@Nginx nginx]# cat RSA.cert
-----BEGIN CERTIFICATE-----
MIID7TCCAtWgAwIBAgIURArMMc1pdY3m58xYmnBOt0g7skEwDQYJKoZIhvcNAQEL
BQAwgYUxCzAJBgNVBAYTAlVTMQswCQYDVQQIDAJDQTERMA8GA1UEBwwIQ2FtcGJl
bGwxEDAOBgNVBAoMB1V0aW1hY28xEDAOBgNVBAsMB1V0aW1hY28xDjAMBgNVBAMM
BU5naW54MSIwIAYJKoZIhvcNAQkBFhNzdXBwb3J0QHV0aW1hY28uY29tMB4XDTIy
MDkxNjA5NDIxNloXDTIzMDkxNjA5NDIxNlowgYUxCzAJBgNVBAYTAlVTMQswCQYD
VQQIDAJDQTERMA8GA1UEBwwIQ2FtcGJlbGwxEDAOBgNVBAoMB1V0aW1hY28xEDAO
BgNVBAsMB1V0aW1hY28xDjAMBgNVBAMMBU5naW54MSIwIAYJKoZIhvcNAQkBFhNz
dXBwb3J0QHV0aW1hY28uY29tMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKC
AQEAhk2Kr3pxpBAPid9jxZJ036/LwpAZXfZt+uR4QPH3oTgwLi4qZQQpWbm4dYFj
ZPWSN75RgU/Oscz7CQjK+xM21CN2uED6Vtkl1vwRLvS5tDehQTIC3BSl4/2UMYLp
tHdZL6gcekzb9zsLN8ncelmdWvdd8cnqmAEHXLnHxKMGjoxuD5uLSJnHdUxXWG4j
TLKxa/O8TyHZqXuwK/NZ2NcYTHeNUYUP17NDSbHOgpS5SukIxzpgfvJy1yKfVts8
oFrGBbJSHwCobj3P63njJfkl3ybaBurnozqLbqRpMROEBRblrE1CeDHAI6fzfdJn
F1hFFxT3E7IYiWapFONnUK/BLQIDAQABo1MwUTAdBgNVHQ4EFgQUGDTV0FWI0ENC
EUquZlsUP8P0mBcwHwYDVR0jBBgwFoAUGDTV0FWI0ENCEUquZlsUP8P0mBcwDwYD
VR0TAQH/BAUwAwEB/zANBgkqhkiG9w0BAQsFAAOCAQEAWNpN7lAAZi9HBFw4TNXL
7jI8eBhBrPrDMO9sFy92oHqn0JK6wJVnnA21PeBLfILgazCoxF3PFaocmZYp1lSw
hYKqLzOQdj+wAsYfm4yI+bcO/frRxjfnjP8AFqzZH4VbUUnlvyWidpDH+4koHFBM
mc69sU/l5jJd4vcQWJpN1+7XfQQcPK2t5vh+lPBIsuq+F6V2d4ERUomytS9EpuAD
yIDYimUQi02N0478ZVPHYGfxG6GoHQm4EKr7gF3D4MpfL9jHSwiHmZz85L1tOYc6
wvJMWV9f7ZSt5dRGHFx7KHnVoBtKO2GS7LfZFyah3UMRgR1QffsDfFTRfk9ky47/
mQ==
-----END CERTIFICATE-----
[root@Nginx nginx]#
```

Figure 12 : Content of self-signed certificate file

It is recommended to use CA signed certificate for production environment.

5.6 Configuring NGINX to use Utimaco HSM

To configure Nginx to use Utimaco Keys for SSL follow these steps.

1. Open the `/usr/local/nginx/nginx.conf` file and make the following changes.

›_ Console

```
ssl_engine pkcs11;

     server {     listen 443 ssl;
     ..................
     ssl_certificate      <certifcate_directory>RSA.cert;
     ssl_certificate_key  engine:pkcs11:slot_<no.>-id_<CKA_ID>;

    ...................
        }

}
```

Below is the sample working configuration file for NGINX for SSL.

›_ Console

```
[root@Nginx nginx]# cat /usr/local/nginx/nginx.conf

 user  root; worker_processes  1; ssl_engine pkcs11; error_log  logs/error.log
info;

 events {     worker_connections  1024;

}  http {     include       mime.types;     default_type  application/octet-
stream;     sendfile        on;     keepalive_timeout  65;     server
{         listen       80;         server_name  localhost;         location /
{             root   html;             index  index.html index.htm;

        }         error_page   500 502 503 504  /50x.html;         location = /
50x.html {             root   html;

        }

    }



    # HTTPS server

     server {     listen 443 ssl;

     server_name  localhost;     ssl_certificate      /root/nginx/RSA.cert;
ssl_certificate_key  engine:pkcs11:slot_4-id_45;     access_log  /tmp/
sslparams.log;     location / {     root   html;
         index  index.html index.htm;

        }

    }

}
```

Here:

- `/root/nginx/RSA.cert` is the signed certificate.
- `engine:pkcs11:slot_4-id_45` is for slot no. 4 on the HSM and 45 is the `CKA_ID` of the private key on Utimaco HSM.

2. Restart NGINX service.

›_ Console

```
# ./nginx -s reload
```

Provide slot PIN when prompted.

3. Open https://<nginx_server_ip> in any browser and verify that you can see nginx welcome page.

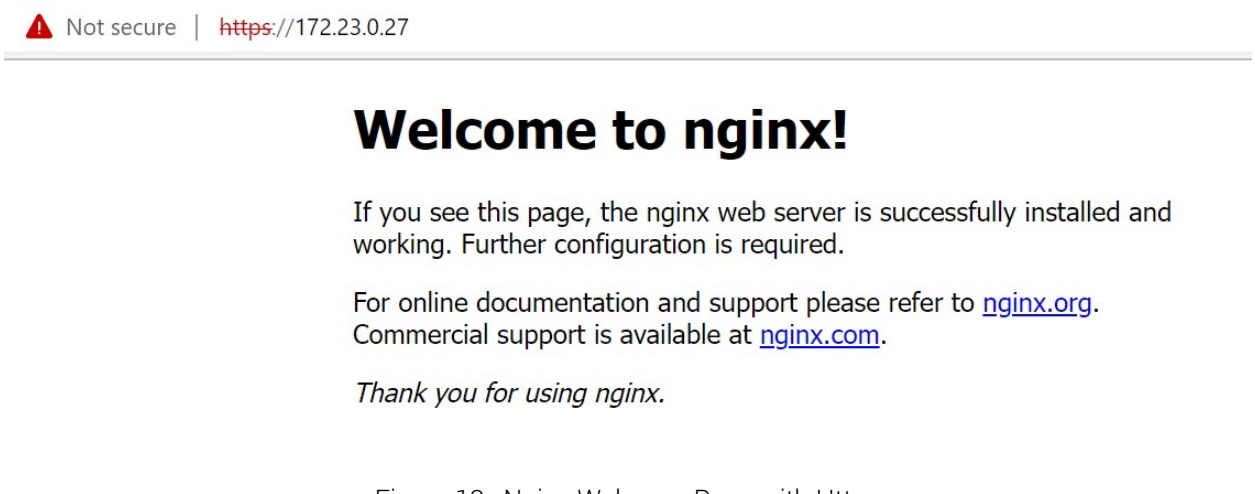

Figure 13 : Nginx Welcome Page with Https

5.7 Migrating Existing Software Keys to Utimaco HSM

If you have NGINX running over SSL with the existing keys locally somewhere on the directory, you can migrate those keys to Utimaco HSM and securely store them.

To migrate the existing key to Utimaco HSM follow these steps:

1. Make sure to complete the steps for installing and configuring openssl and libp11 in section 5.1, 5.2 and 5.3 before proceeding.
2. Convert a private key to pkcs12 format.

›_ Console

```
# openssl pkcs12 -export -nocerts -inkey private.pem -out private.key.p12
```

3. Import converted private key to HSM using p11tool2.

›_ Console

```
# p11tool2 slot=4 loginuser=123456 PubKeyAttr=CKA_LABEL="RSA Public

Key",CKA_ID=0x25 PrvKeyAttr=CKA_LABEL="RSA Private Key",CKA_ID=0x25
ImportP12=private.key.p12,ask
```

Only private key is required to import to Utimaco HSM, as certificate contains only public key information.

4. List the generated keys using p11tool2.

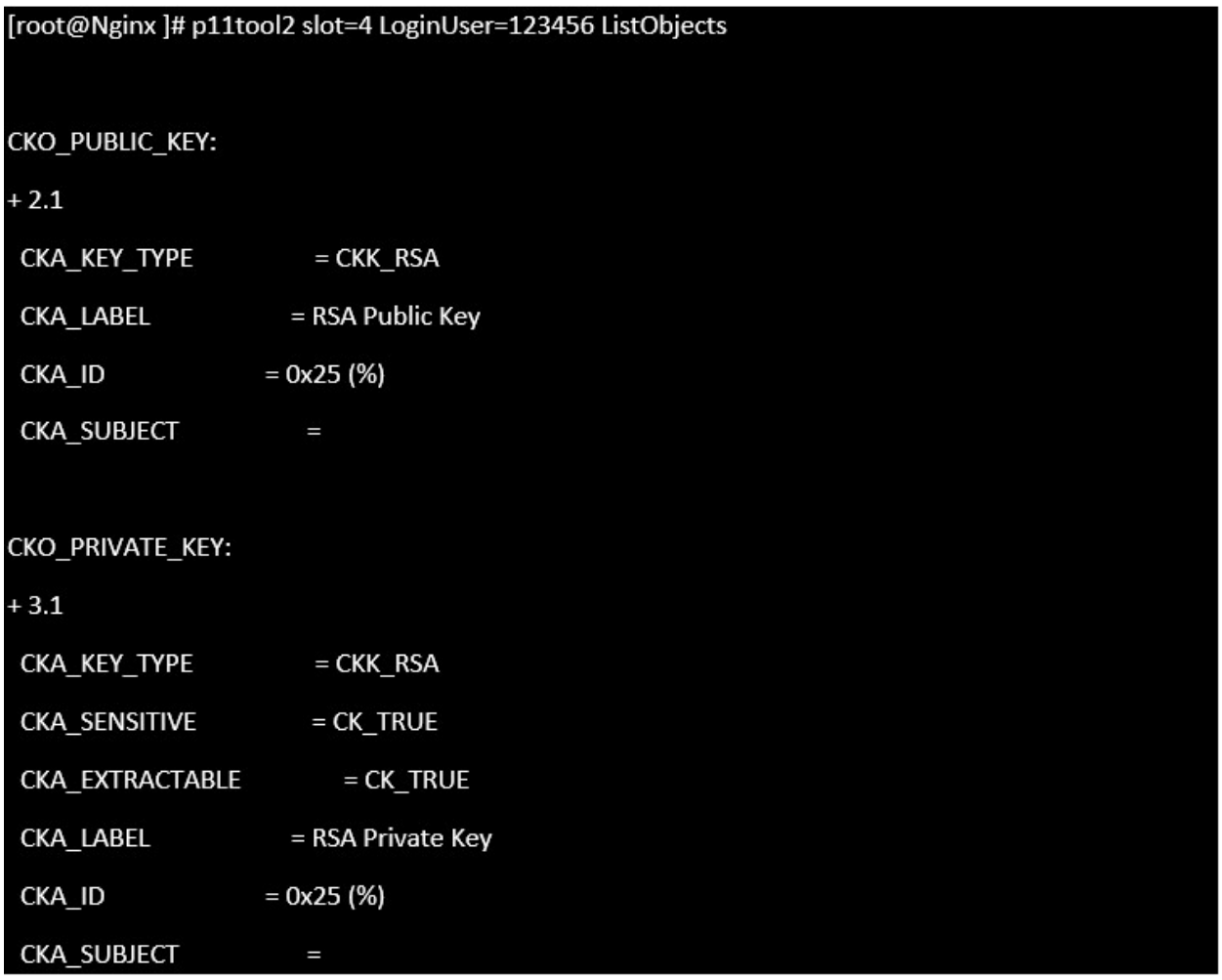

Figure 14 : List keys

5. Open the `nginx.conf` file and make the following changes.

›_ Console

```
ssl_engine pkcs11;

     server {     listen 443 ssl;
     ..............
 ssl_certificate /root/nginx/cert.pem;
 ssl_certificate_key engine:pkcs11:slot_4-id_25;

 ...................
 }
}
```

Here:

- `/root/nginx/RSA.cert` is the existing signed certificate.
- `engine:pkcs11:slot_4-id_25` is for slot no. 4 HSM and 25 is the `CKA_ID` of the private key on Utimaco HSM.

Below is the sample working configuration file for NGINX for SSL.

›_ Console

```
[root@Nginx nginx]# vi /usr/local/nginx/nginx.conf
user root;
worker_processes 1;
ssl_engine pkcs11;
error_log logs/error.log info;
events {
 worker_connections 1024;
}
http {
 include mime.types;
 default_type application/octet-stream;
 sendfile on;
 keepalive_timeout 65;
 server {
 listen 80;
 server_name localhost;
 location / {
 root html;
 index index.html index.htm;
 }
 error_page 500 502 503 504 /50x.html;
 location = /50x.html {
 root html;
 }
 }
 #HTTPS server
 server {
 listen 443 ssl;
 server_name localhost;
 ssl_certificate /root/nginx/cert.pem;
 ssl_certificate_key engine:pkcs11:slot_4-id_25;
 access_log /tmp/sslparams.log;
 location / {
 root html;
 index index.html index.htm;
 }
 }
}
```

6. Restart NGINX Open Source:

›_ Console

```
# ./nginx -s reload
```

Provide slot PIN when prompted.

7. Open https://<nginx_server_ip> in any browser and verify that you can see nginx welcome page.

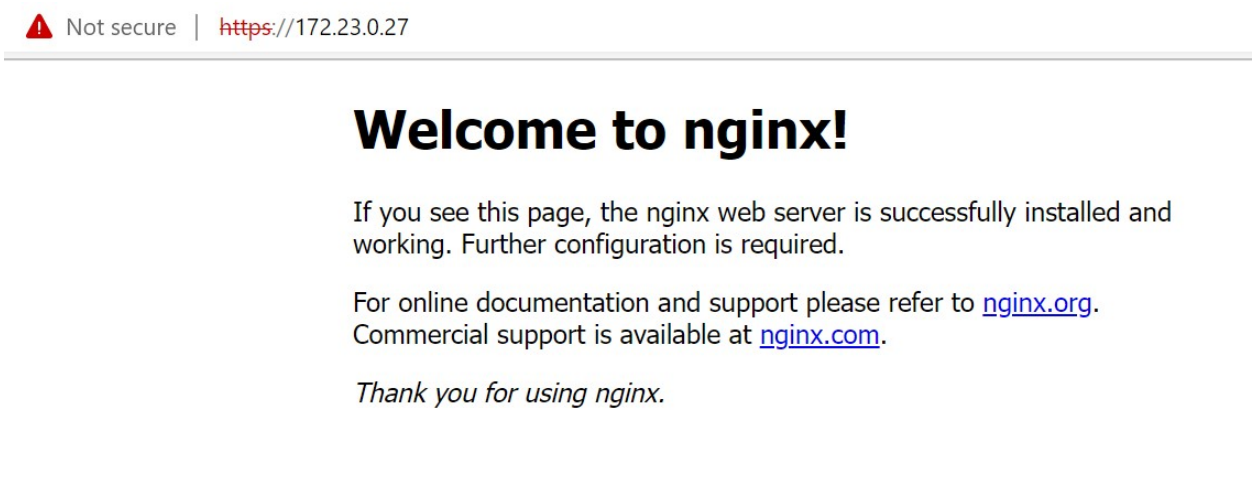

Figure 15 : Nginx Page with Https

Once NGINX is running successfully after migrating the keys on Utimaco HSM you can delete the private key from the software location.

This completes the integration of NGINX with Utimaco HSM.

6 Troubleshooting

Error	Diagnosis
The CryptoServer PKCS#11 Library R3 is not initialized. Error CKR_CRYPTOKI_NOT_INITIALIZED occurred	PKCS#11 Slot is not initialized. Refer Initialize a Slot
OpenSSL> engine -t dynamic -pre SO_PATH:/usr/lib64/openssl/engines/pkcs11.so -pre ID:pkcs11 -pre LIST_ADD:1 -pre LOAD -pre MODULE_PATH:/opt/utimaco/lib/libcs_pkcs11_R3.soengine: Cannot mix flags and engine names. engine: Use -help for summary. error in engine	Run the correct command OpenSSL> engine dynamic -pre SO_PATH:/usr/lib64/engines-1.1/pkcs11.so pre ID:pkcs11 -pre LIST_ADD:1 -pre LOAD pre MODULE_PATH:/opt/utimaco/lib/libcs_pkcs 11_R3.so

Error	Diagnosis
openssl req -engine pkcs11 -new -key 4F70656E73736C4B6579 -keyform engine -out req.pem -text -x509 -subj "CN=Utimaco" invalid engine "pkcs11" 139703122831248:error:25066067:DSO support routines:DLFCN_LOAD:could not load the shared library:dso_dlfcn.c:187:filename(/usr/lib64/open ssl/engines/libpkcs11.so): libcrypto.so.1.1: cannot open shared object file: No such file or directory 139703122831248:error:25070067:DSO support routines:DSO_load:could not load the shared library:dso_lib.c:233: 139703122831248:error:260B6084:engine routines:DYNAMIC_LOAD:dso not found:eng_dyn.c:467: 139703122831248:error:2606A074:engine routines:ENGINE_by_id:no such engine:eng_list.c:392:id=pkcs11 139703122831248:error:25066067:DSO support routines:DLFCN_LOAD:could not load the shared library:dso_dlfcn.c:187:filename(libpkcs11.so): libpkcs11.so: cannot open shared object file: No such file or directory 139703122831248:error:25070067:DSO support routines:DSO_load:could not load the shared library:dso_lib.c:233: 139703122831248:error:260B6084:engine routines:DYNAMIC_LOAD:dso not found:eng_dyn.c:467: no engine specified unable to load Private Key	Export the below value of LD_LIBRARY_PATH and Path for Openssl and libp11 to avoid the above error. export LD_LIBRARY_PATH=/usr/local/openssl/lib:/u sr/local/libp11/lib export PATH=/usr/local/openssl/bin:$PATH

Table 6: List of errors and their diagnoses

7 Further Information

This document forms a part of the information and support which is provided by Utimaco IS GmbH. Additional documentation can be found on the product CD in the Documentation directory.

All CryptoServer product documentation is also available at the Utimaco IS GmbH website: *http://hsm.utimaco.com*.

8 References

Reference	Title/Company	Document No.
[CSADMIN]	CryptoServer – csadm Manual/Utimaco IS GmbH	2009-0003
[CSTrSh]	CryptoServer Troubleshooting/Utimaco IS GmbH	M011-0008-en
[CSADMIN2]	CryptoServer_csadm_Manual_Systemadministrators.pdf	2009-0003
[CSP11Tool2]	CryptoServer_p11tool2_Manual.pdf	2012-0004
[CSPKCSM]	CryptoServer - PKCS#11 P11CAT Manual	M013-0001-en
[CSLAN5]	CryptoServerLAN_Manual_Systemadministrators.pdf	2018-0004

Table 7: References

9 Contact and Support Information

You can reach us from Monday to Friday, 09.00 a.m. to 05.00 p.m., Central European Time (CET).

Utimaco IS GmbH
Krefelder Str. 220
52070 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.com
  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.