Hyperledger Fabric

1.5.4

Integration Guide

CryptoServer HSM

SecurityServer 4.45.5

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	2026-07-27
Status	PUBLISHED
Document No.	IG-2026-0082

Table of Contents

1 Introduction

This guide is part of the information and support provided by Utimaco. Additional documentation 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 from Utimaco's website 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 Opensource Hyperledger Fabric. Utimaco HSM protects the private keys and handles cryptographic operations, allowing the peers and orderer nodes to sign and endorse transactions without exposing their private keys.

1.2 Target Audience for This Guide

This guide is intended for administrators of Opensource Hyperledger Fabric 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
BCCSP	Blockchain Crypto Service Provider
CA	Certificate Authority
CD	Compact Disc
CSAR	Cloud Service Architecture
cURL	Client URL (Client Uniform Resource Locator)
DLT	Distributed Ledger Technology
GUI	Graphical User Interface

Abbreviation	Meaning
HMAC	Hash Message Authentication Code
HSM	Hardware Security Module
ID	Identity
IoT	Internet of Things
JRE	Java Runtime Environment
LAN	Local Area Network
NPM	Node Package Manager
PCI-e	PCI Express Interface
PKCS#11	Public-Key Cryptography Standard #11
YAML	Yet Another Markup Language

Table 2: List of abbreviations

2 Overview

2.1 Opensource Hyperledger Fabric

To expand and develop blockchain technology, certain tools and frameworks are required. Hyperledger is one of the blockchain frameworks which is popularly used in the blockchain ecosystem.

The process of developing blockchain applications and systems can be implemented using the opensource tools known as Hyperledger. It is done by collaborating with developers and businesses working with the Distributed Ledger Technology (DLT).

The main purpose of Hyperledger systems is to provide setup, tools, framework, guidelines, and standards for building blockchain-based applications by collaborating with the business and developers using DLT. This Global collaboration involves the organization dealing with finances, manufacturing, Internet of Things (IoT), supply chain management, banking, technology, etc.

Hyperledger Fabric is a platform that has two core features, namely, modularity and versatility, for developing blockchain-based applications, solutions, products, and services, especially for business enterprises. It uses modular architecture and sets the foundation to implement various blockchain use cases and satisfy business needs. Also, Hyperledger fabric uses components like plug-and-play, offering consensus and membership services.

Hyperledger Fabric is the distributed ledger software that is popularly used as a blockchain framework. IBM designed this decentralized ledger technology (DLT) platform for industrial use. Due to its permission to access and private features, it offers suitable solutions to businesses to segregate the information or data and increase the transaction speed.

Below are some features of Fabric:

- Highly modular: Hyperledger Fabric offers modularity as it has been designed with modular architecture.
- Pluggable consensus: Hyperledger features plug-and-play consensus.
- Permissioned architecture: This Hyperledger feature increases privacy as only permission to access is available.
- Open smart contract model: Hyperledger offers flexibility to implement the desired solution to the system.

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 Prerequisites and Requirements

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

3.1 Tested Versions

The integrations that have been successfully tested with the Utimaco HSM with Opensource Hyperledger Fabric.

Operating Systems	Hyperledger	Utimaco Security Server Version	Utimaco HSM
Ubuntu 20.04	Fabric 1.5.4	SecurityServer 4.45.5	CryptoServer CSeSeries/Se-Series

Table 3: List of tested versions

3.2 Software Requirements

Software	Software Requirements
Opensource Hyperledger	Fabric 1.5.4
JRE 8	Java Runtime Environment 8
HSM Interface	SecurityServer PKCS#11 Provider
Docker	Docker 20.10.12
cURL	Latest Version
Docker Compose	Docker Compose — version 1.14.0 or greater

Software	Software Requirements
Golang	go1.18.3 linux/amd64
Nodejs	Nodejs — version 8.x (other versions are not in support yet)
NPM	NPM — version 5.x
Python	Python 2.7
Libtool	Latest

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 4.45.5 or higher
Utimaco PCI-e HSM	CryptoServer CSe-Series/Se-Series PCI-e with firmware SecurityServer 4.45.5 or higher

Table 5: List of hardware requirements

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 installed/set up is listed in *Tested Versions*.
- The SecurityServer installed/set u[is listed in *Tested Versions*.
- The PKCS#11 library is set up and configured as per your environment. Refer to the CryptoServer documentations to set up and configure the PKCS#11 library.
- There is admin access as it is required to install software.
- You have installed docker engine on your Linux operating system. Refer to the docker documentation website.
- You have determined the directory where you want to download the fabric samples.
- You install and set environment variable path for Golang. For this, refer to ***https://go.dev/doc/install***.
- Throughout this integration, we have used two terminals. One with the host and another with the docker container.
- Network ports 7055 and 9444 used in this Integration guide need to be allowed through the firewall. However, they can be changed as per the environment.

4 Hyperledger Fabric SecurityServer Container: Download, Build, and Run

4.1 To Download Hyperledger Fabric

To download the Hyperledger fabric repository, run the following commands.

›_ Console

```
# mkdir -p go/src/github.com/Hyperledger

# cd go/src/github.com/Hyperledger

# git clone https://github.com/hyperledger/fabric-ca

# cd fabric-ca

# git checkout v1.5.4
```

4.2 Creating a Docker Image with Utimaco Library and Utilities

1. Go to the path `/root/go/src/github.com/hyperledger/fabric-ca`.
2. Create one directory with name `utimaco_utilities` on path `/root/go/src/github.com/hyperledger/fabric-ca` and copy below files inside this directory.

›_ Console

```
# cp /etc/utimaco/cs_pkcs11_R3.cfg

/root/go/src/github.com/hyperledger/fabric-ca/utimaco_utilities/

# cp /opt/utimaco/lib/libcs_pkcs11_R3.so

/root/go/src/github.com/hyperledger/fabric-ca/utimaco_utilities/

# cp /opt/utimaco/bin/p11tool2 /root/go/src/github.com/hyperledger/fabricca/
utimaco_utilities

# cp /opt/utimaco/bin/ADMIN.key

/root/go/src/github.com/hyperledger/fabric-ca/utimaco_utilities

# cp /opt/utimaco/bin/csadm /root/go/src/github.com/hyperledger/fabricca/
utimaco_utilities
```

Figure 1 : Copy library and configuration files

3. Verify that files are copied at `/root/go/src/github.com/hyperledger/fabric-ca/utimaco_utilities`.

›_ Console

```
# cd /root/go/src/github.com/hyperledger/fabric-ca/utimaco_utilities
root@Fabric-CA:~/go/src/github.com/hyperledger/fabricca/utimaco_utilities# ll
```

```
root@Fabric-CA:~/go/src/github.com/hyperledger/fabric-ca/utimaco_utilities# ll
total 10500
drwxr-xr-x  2 root root    4096 Aug 22 11:26 ./
drwxr-xr-x 21 root root    4096 Oct 19 09:45 ../
-rw-r--r--  1 root root    1241 Aug 26 06:57 ADMIN.key
-rw-r--r--  1 root root    2289 Aug 22 11:26 cs_pkcs11_R3.cfg
-rw-r--r--  1 root root 1442600 Aug 22 10:38 csadm
-rw-r--r--  1 root root 4080024 Aug 22 11:24 libcs_pkcs11_R3.so
-rw-r--r--  1 root root 5205248 Aug 22 10:38 p11tool2
root@Fabric-CA:~/go/src/github.com/hyperledger/fabric-ca/utimaco_utilities#
```

Figure 2 : List files

4. Create a new file Dockerfile to build a Fabric image.

›_ Console

```
# cd /root/go/src/github.com/hyperledger/fabric-ca

# vim Dockerfile
```

5. Copy the following lines into the file (Dockerfile).

›_ Console

```
ARG GO_VER

ARG ALPINE_VER

FROM ubuntu:focal

ARG GO_LDFLAGS

ARG GO_TAGS

RUN apt-get update

RUN apt-get install -y curl gcc git musl-dev

RUN rm -rf /var/lib/apt/lists/* ENV GOLANG_VERSION 1.18.3

RUN curl -sSL

https://storage.googleapis.com/golang/go$GOLANG_VERSION.linux-amd64.tar.gz

| tar -v -C /usr/local -xz

ENV PATH /usr/local/go/bin:$PATH

RUN mkdir -p /go/src /go/bin && chmod -R 777 /go

ENV GOROOT /usr/local/go

ENV GOPATH /go

ENV PATH /go/bin:$PATH

RUN mkdir -p /etc/utimaco/

RUN mkdir -p /opt/utimaco/bin

RUN mkdir -p /opt/utimaco/lib

COPY /utimaco_utilities/cs_pkcs11_R3.cfg /etc/utimaco/

COPY /utimaco_utilities/libcs_pkcs11_R3.so /opt/utimaco/lib

COPY /utimaco_utilities/p11tool2 /opt/utimaco/bin

COPY /utimaco_utilities/ADMIN.key /opt/utimaco/bin

COPY /utimaco_utilities/csadm /opt/utimaco/bin
RUN chmod -R 550 /opt/utimaco
```

›_ Console

```
RUN chmod -R 600 /etc/utimaco

ADD . $GOPATH/src/github.com/hyperledger/fabric-ca

RUN go install -tags "${GO_TAGS}" -ldflags "${GO_LDFLAGS}" \ github.com/
hyperledger/fabric-ca/cmd/fabric-ca-server \ && go install -tags "${GO_TAGS}"
-ldflags "${GO_LDFLAGS}" \ github.com/hyperledger/fabric-ca/cmd/fabric-ca-client
ENV FABRIC_CA_HOME /etc/hyperledger/fabric-ca-server

EXPOSE 7055

CMD fabric-ca-server start -b admin:adminpw
```

4.3 Building and Running the Hyperledger Fabric Image

Use the build command inside the project directory, created in the previous step to build the container image with the tag of demohyperledger version.

›_ Console

```
# cd /root/go/src/github.com/hyperledger/fabric-ca # docker build -t
demohyperledger:1.5.4 .

# GO_TAGS=pkcs11 make fabric-ca-server

# GO_TAGS=pkcs11 make fabric-ca-client
```

```
root@Fabric-CA:~/go/src/github.com/hyperledger/fabric-ca# docker build -t demohyperledger:1.5.4 .
Sending build context to Docker daemon  174.4MB
Step 1/32 : ARG GO_VER
Step 2/32 : ARG ALPINE_VER
Step 3/32 : FROM ubuntu:focal
 ---> 3bc6e9f30f51
Step 4/32 : ARG HSMPort=288
 ---> Running in c74819e0bb71
Removing intermediate container c74819e0bb71
 ---> 2edb99ac163a
Step 5/32 : ARG GO_LDFLAGS
 ---> Running in bf14941b2fa7
Removing intermediate container bf14941b2fa7
 ---> 4b19ca915898
Step 6/32 : ARG GO_TAGS
 ---> Running in 41caf25f0e13
Removing intermediate container 41caf25f0e13
 ---> 48f2ebcdf953
Step 7/32 : RUN apt-get update
 ---> Running in b7c8ba766fb9
Get:1 http://archive.ubuntu.com/ubuntu focal InRelease [265 kB]
Get:2 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Get:3 http://archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:4 http://security.ubuntu.com/ubuntu focal-security/multiverse amd64 Packages [27.5 kB]
Get:5 http://archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
Get:6 http://security.ubuntu.com/ubuntu focal-security/universe amd64 Packages [910 kB]
Get:7 http://archive.ubuntu.com/ubuntu focal/main amd64 Packages [1275 kB]
Get:8 http://security.ubuntu.com/ubuntu focal-security/restricted amd64 Packages [1552 kB]
Get:9 http://archive.ubuntu.com/ubuntu focal/universe amd64 Packages [11.3 MB]
Get:10 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [2178 kB]
Get:11 http://archive.ubuntu.com/ubuntu focal/restricted amd64 Packages [33.4 kB]
Get:12 http://archive.ubuntu.com/ubuntu focal/multiverse amd64 Packages [177 kB]
Get:13 http://archive.ubuntu.com/ubuntu focal-updates/restricted amd64 Packages [1667 kB]
Get:14 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [2644 kB]
Get:15 http://archive.ubuntu.com/ubuntu focal-updates/universe amd64 Packages [1206 kB]
Get:16 http://archive.ubuntu.com/ubuntu focal-updates/multiverse amd64 Packages [30.2 kB]
Get:17 http://archive.ubuntu.com/ubuntu focal-backports/main amd64 Packages [55.2 kB]
Get:18 http://archive.ubuntu.com/ubuntu focal-backports/universe amd64 Packages [27.4 kB]
Fetched 23.7 MB in 5s (5040 kB/s)
Reading package lists...
Removing intermediate container b7c8ba766fb9
 ---> 6f6102888c7a
Step 8/32 : RUN apt-get install -y curl gcc git musl-dev
 ---> Running in 253b7ddc871f
```

Figure 3 : Docker build image

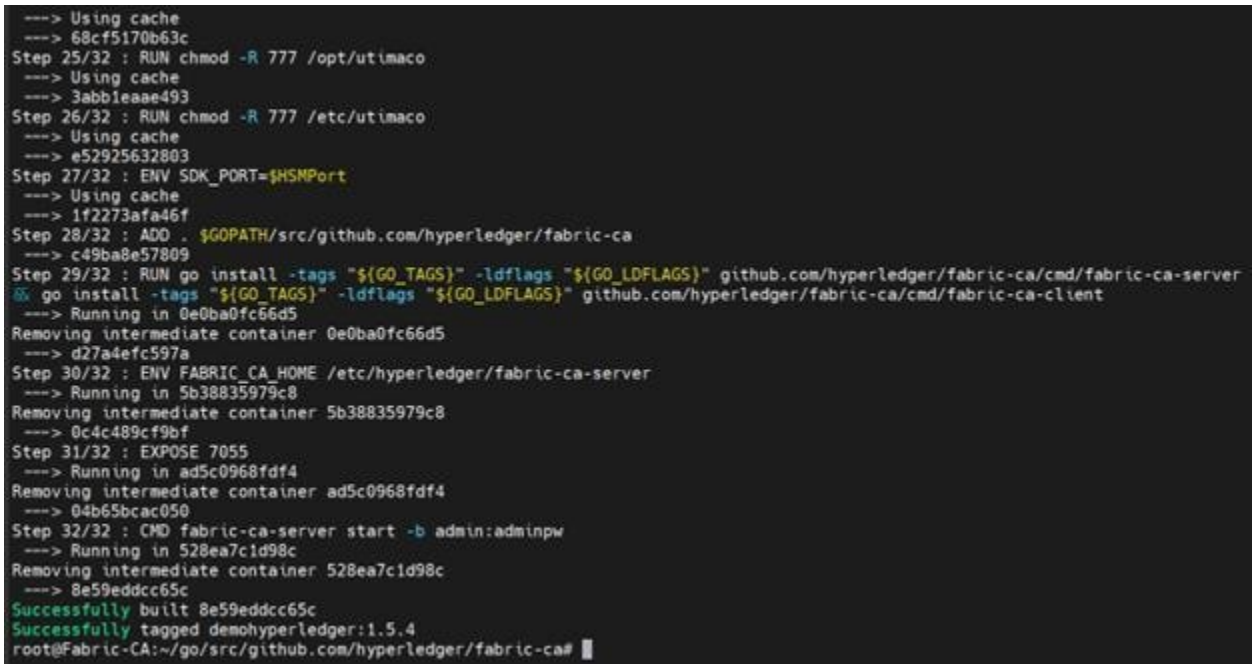

Figure 4 : Docker build image

4.3.1 To List the Created Docker Image

Run below command on the host machine, to verify that the Docker image is created.

›_ Console

```
# docker image ls
```

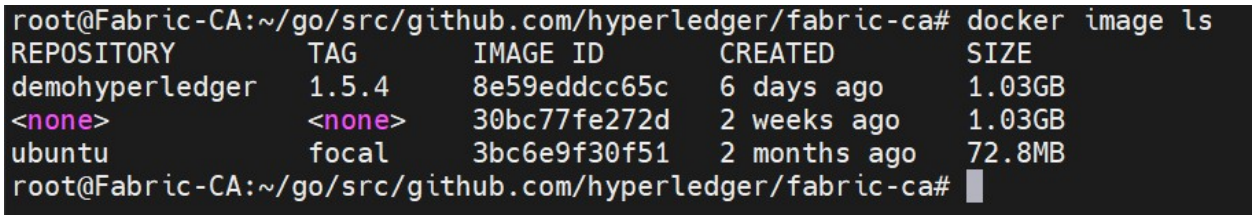

Figure 5 : Docker image

4.3.2 Run the Hyperledger Fabric Image

Now that you have successfully created a docker image for Hyperledger Fabric, we can start a Hyperledger container. After you have started the container, you can verify the functionality of the CryptoServer HSM from container.

Run the hyperledger Image.

›_ Console

```
# docker run -dt --name demo demohyperledger:1.5.4
```

```
root@Fabric-CA:~/go/src/github.com/hyperledger/fabric-ca# docker run -dt --name demo
demohyperledger:1.5.4

8dc5d676844b4955112d58c1bab297259c06c31a54f65a395a18482917fd0a3e

root@Fabric-CA:~/go/src/github.com/hyperledger/fabric-ca#
```

Figure 6 : Docker container: demo

4.3.3 Verifying the Status of the Container

›_ Console

```
# docker container list
```

```
root@Fabric-CA:~/go/src/github.com/hyperledger/fabric-ca# docker container list

CONTAINER ID   IMAGE                  COMMAND                  CREATED          STATUS          PORTS
NAMES

8dc5d676844b   demohyperledger:1.5.4  "/bin/sh -c 'fabric-..."  About a minute ago  Up About a minute
7055/tcp                              demo

root@Fabric-CA:~/go/src/github.com/hyperledger/fabric-ca#
```

Figure 7 : Docker container list

4.3.4 CryptoServer PKCS#11 Configuration in Fabric Container

1. Go inside the running container using the command below.

›_ Console

```
# docker exec -it 8dc5d676844b /bin/bash
```

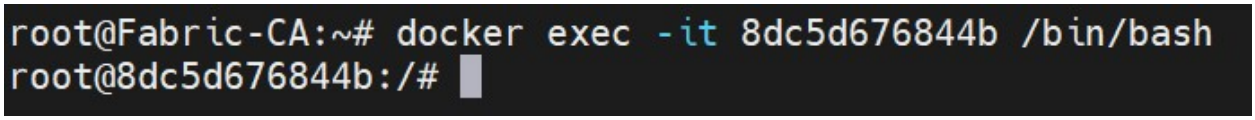

Figure 8 : Docker container

Here the value `8dc5d676844b` is the container ID from section *4.3.3 Verifying the Status of the Container*.

›_ Console

```
root@8dc5d676844b:/# ls -la
```

```
root@8dc5d676844b:/# ls -la
total 76
drwxr-xr-x   1 root root 4096 Oct  6 17:25 .
drwxr-xr-x   1 root root 4096 Oct  6 17:25 ..
-rwxr-xr-x   1 root root    0 Oct  6 17:25 .dockerenv
lrwxrwxrwx   1 root root    7 Aug  1 13:22 bin → usr/bin
drwxr-xr-x   2 root root 4096 Apr 15  2020 boot
drwxr-xr-x   5 root root  360 Oct  6 17:25 dev
drwxr-xr-x   1 root root 4096 Oct  6 17:25 etc
drwxrwxrwx   1 root root 4096 Sep 30 06:42 go
drwxr-xr-x   2 root root 4096 Apr 15  2020 home
lrwxrwxrwx   1 root root    7 Aug  1 13:22 lib → usr/lib
lrwxrwxrwx   1 root root    9 Aug  1 13:22 lib32 → usr/lib32
lrwxrwxrwx   1 root root    9 Aug  1 13:22 lib64 → usr/lib64
lrwxrwxrwx   1 root root   10 Aug  1 13:22 libx32 → usr/libx32
drwxr-xr-x   2 root root 4096 Aug  1 13:22 media
drwxr-xr-x   2 root root 4096 Aug  1 13:22 mnt
drwxr-xr-x   1 root root 4096 Sep 30 06:42 opt
dr-xr-xr-x 191 root root    0 Oct  6 17:25 proc
drwx------   1 root root 4096 Sep 30 06:43 root
drwxr-xr-x   5 root root 4096 Aug  1 13:25 run
lrwxrwxrwx   1 root root    8 Aug  1 13:22 sbin → usr/sbin
drwxr-xr-x   2 root root 4096 Aug  1 13:22 srv
dr-xr-xr-x  12 root root    0 Oct  6 17:25 sys
drwxrwxrwt   2 root root 4096 Aug  1 13:25 tmp
drwxr-xr-x   1 root root 4096 Aug  1 13:22 usr
drwxr-xr-x   1 root root 4096 Aug  1 13:25 var
root@8dc5d676844b:/#
```

Figure 9 : Container/Directory

2. Copy the newly generated binaries `fabric-ca-server` and `fabric-ca-client` from host machine's `/root/go/src/github.com/hyperledger/fabric-ca/bin` directory to container's `/go/bin` directory.

›_ Console

```
root@Fabric-CA: # cd /root/go/src/github.com/hyperledger/fabric-ca/bin
root@Fabric-CA:~/go/src/github.com/hyperledger/fabric-ca/bin# docker cp fabric-
ca-client demo:/go/bin

root@Fabric-CA:~/go/src/github.com/hyperledger/fabric-ca/bin# docker cp fabric-
ca-server demo:/go/bin
```

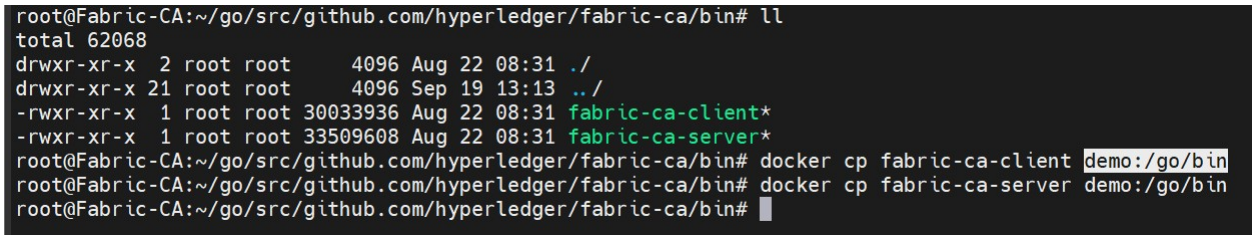

Figure 10 : Copy binary files

3. Verify the above binary files are copied at `/go/bin` inside the container.

›_ Console

```
root@8dc5d676844b:/go/bin# ll
```

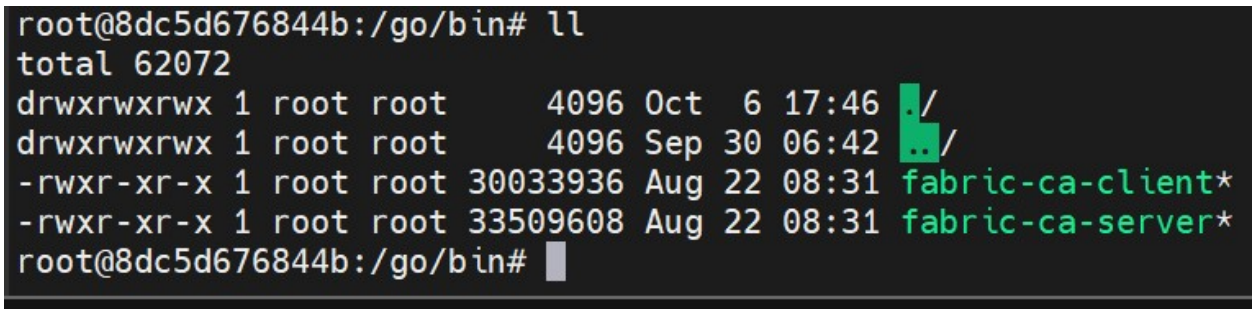

Figure 11 : Verified binary files

4. Install vim editor with `apt-get` commands inside the container as shown below.

›_ Console

```
# apt-get update

# apt-get install vim

# cd /etc/utimaco
```

```
root@8dc5d676844b:/# apt-get update
Get:1 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Get:2 http://archive.ubuntu.com/ubuntu focal InRelease [265 kB]
Get:3 http://archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:4 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [2183 kB]
Get:5 http://archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
Get:6 http://archive.ubuntu.com/ubuntu focal/main amd64 Packages [1275 kB]
Get:7 http://archive.ubuntu.com/ubuntu focal/restricted amd64 Packages [33.4 kB]
Get:8 http://archive.ubuntu.com/ubuntu focal/universe amd64 Packages [11.3 MB]
Get:9 http://security.ubuntu.com/ubuntu focal-security/universe amd64 Packages [916 kB]
Get:10 http://security.ubuntu.com/ubuntu focal-security/restricted amd64 Packages [1556 kB]
Get:11 http://security.ubuntu.com/ubuntu focal-security/multiverse amd64 Packages [27.5 kB]
Get:12 http://archive.ubuntu.com/ubuntu focal/multiverse amd64 Packages [177 kB]
Get:13 http://archive.ubuntu.com/ubuntu focal-updates/universe amd64 Packages [1215 kB]
Get:14 http://archive.ubuntu.com/ubuntu focal-updates/restricted amd64 Packages [1671 kB]
Get:15 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [2650 kB]
Get:16 http://archive.ubuntu.com/ubuntu focal-updates/multiverse amd64 Packages [30.2 kB]
Get:17 http://archive.ubuntu.com/ubuntu focal-backports/universe amd64 Packages [27.4 kB]
Get:18 http://archive.ubuntu.com/ubuntu focal-backports/main amd64 Packages [55.2 kB]
Fetched 23.8 MB in 4s (5419 kB/s)
Reading package lists ... Done
root@8dc5d676844b:/# apt-get install vim
Reading package lists ... Done
Building dependency tree
Reading state information ... Done
The following additional packages will be installed:
  alsa-topology-conf alsa-ucm-conf file libasound2 libasound2-data libcanberra0 libgpm2 libltdl7 libmagic-mgc libmagic1
  libmpdec2 libogg0 libpython3.8 libpython3.8-minimal libpython3.8-stdlib libreadline8 libtdb1 libvorbis0a libvorbisfile3
  mime-support readline-common sound-theme-freedesktop vim-common vim-runtime xxd xz-utils
Suggested packages:
  libasound2-plugins alsa-utils libcanberra-gtk0 libcanberra-pulse gpm readline-doc ctags vim-doc vim-scripts
The following NEW packages will be installed:
  alsa-topology-conf alsa-ucm-conf file libasound2 libasound2-data libcanberra0 libgpm2 libltdl7 libmagic-mgc libmagic1
  libmpdec2 libogg0 libpython3.8 libpython3.8-minimal libpython3.8-stdlib libreadline8 libtdb1 libvorbis0a libvorbisfile3
  mime-support readline-common sound-theme-freedesktop vim vim-common vim-runtime xxd xz-utils
0 upgraded, 27 newly installed, 0 to remove and 6 not upgraded.
Need to get 13.0 MB of archives.
```

Figure 12 : Updating container

```
root@8dc5d676844b:/# apt-get install vim
Reading package lists ... Done
Building dependency tree
Reading state information ... Done
The following additional packages will be installed:
  alsa-topology-conf alsa-ucm-conf file libasound2 libasound2-data libcanberra0 libgpm2 libltdl7 libmagic-mgc libmagic1 libmpdec2
  libogg0 libpython3.8 libpython3.8-minimal libpython3.8-stdlib libreadline8 libtdb1 libvorbis0a libvorbisfile3 mime-support
  readline-common sound-theme-freedesktop vim-common vim-runtime xxd xz-utils
Suggested packages:
  libasound2-plugins alsa-utils libcanberra-gtk0 libcanberra-pulse gpm readline-doc ctags vim-doc vim-scripts
The following NEW packages will be installed:
  alsa-topology-conf alsa-ucm-conf file libasound2 libasound2-data libcanberra0 libgpm2 libltdl7 libmagic-mgc libmagic1 libmpdec2
  libogg0 libpython3.8 libpython3.8-minimal libpython3.8-stdlib libreadline8 libtdb1 libvorbis0a libvorbisfile3 mime-support
  readline-common sound-theme-freedesktop vim vim-common vim-runtime xxd xz-utils
0 upgraded, 27 newly installed, 0 to remove and 6 not upgraded.
Need to get 13.0 MB of archives.
After this operation, 64.7 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu focal/main amd64 libmagic-mgc amd64 1:5.38-4 [218 kB]
Get:2 http://archive.ubuntu.com/ubuntu focal/main amd64 libmagic1 amd64 1:5.38-4 [75.9 kB]
Get:3 http://archive.ubuntu.com/ubuntu focal/main amd64 file amd64 1:5.38-4 [23.3 kB]
Get:4 http://archive.ubuntu.com/ubuntu focal/main amd64 libmpdec2 amd64 2.4.2-3 [81.1 kB]
Get:5 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 libpython3.8-minimal amd64 3.8.10-0ubuntu1~20.04.5 [717 kB]
Get:6 http://archive.ubuntu.com/ubuntu focal/main amd64 mime-support all 3.64ubuntu1 [30.6 kB]
Get:7 http://archive.ubuntu.com/ubuntu focal/main amd64 readline-common all 8.0-4 [53.5 kB]
Get:8 http://archive.ubuntu.com/ubuntu focal/main amd64 libreadline8 amd64 8.0-4 [131 kB]
Get:9 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 libpython3.8-stdlib amd64 3.8.10-0ubuntu1~20.04.5 [1675 kB]
Get:10 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 xxd amd64 2:8.1.2269-1ubuntu5.9 [50.0 kB]
Get:11 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 vim-common all 2:8.1.2269-1ubuntu5.9 [85.0 kB]
Get:12 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 xz-utils amd64 5.2.4-1ubuntu1.1 [82.6 kB]
Get:13 http://archive.ubuntu.com/ubuntu focal/main amd64 alsa-topology-conf all 1.2.2-1 [7364 B]
Get:14 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 alsa-ucm-conf all 1.2.2-1ubuntu0.13 [27.0 kB]
Get:15 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 libasound2-data all 1.2.2-2.1ubuntu2.5 [20.1 kB]
Get:16 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 libasound2 amd64 1.2.2-2.1ubuntu2.5 [335 kB]
Get:17 http://archive.ubuntu.com/ubuntu focal/main amd64 libltdl7 amd64 2.4.6-14 [38.5 kB]
Get:18 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 libtdb1 amd64 1.4.3-0ubuntu0.20.04.1 [44.2 kB]
Get:19 http://archive.ubuntu.com/ubuntu focal/main amd64 libogg0 amd64 1.3.4-0ubuntu1 [24.0 kB]
```

Figure 13 : Installing vim editor

5. Open the `cs_pkcs11_R3.cfg` file inside the container with vim editor and make the following changes.

›_ Console

```
# vim cs_pkcs11_R3.cfg
```

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_I
```

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 4.

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.

4.3.5 Initialize and Verify the PKCS#11 Slot

First, using p11tool2 create the SO or Security Officer and then using the p11tool2 command initialize the slot that you want to use, as well as the slot user, as shown below inside the container.

›_ Console

```
root@8dc5d676844b:/# cd /opt/utimaco/bin

root@8dc5d676844b:/# ./p11tool2 slot=<slot_no> Label=<token_label>
Login=ADMIN,ADMIN.key InitToken=<SO_PIN>

root@8dc5d676844b:/# ./p11tool2 slot=<slot_no> LoginSO=<SO_PIN>
InitPin=<CryptoUser_PIN>
```

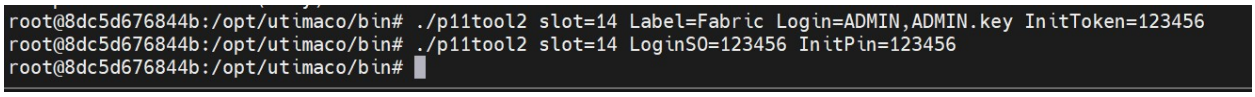

Figure 14 : Initialize and verify the pkcs11 slot

5 Configure and Start the Hyperledger Fabric CA Server

1. Go to `/etc/hyperledger/fabric-ca-server` directory and open `fabric-ca-server-config.yaml` file.

›_ Console

```
# cd /etc/hyperledger/fabric-ca-server

# vim fabric-ca-server-config.yaml
```

2. Locate the **bccsp** section and add the PKCS #11 settings. For example:

```
###############################################################################
bccsp:
    default: PKCS11
    pkcs11:
        Library: /opt/utimaco/lib/libcs_pkcs11_R3.so
        Pin: 123456
        Label: Fabric
        hash: SHA2
        security: 256
        #filekeystore:
            # The directory used for the software file-based keystore
            #keystore: msp/keystore

###############################################################################
```

Figure 15 : Fabric-CA-Server-config file

Here we need to update 2 port numbers in the file `fabric-ca-server-config.yaml` .

- search for 7054 and replace it with 7055
- search for 9443 and replace it with 9444

3. Delete any keystore in `/etc/hyperledger/fabric-ca-server` such as the msp directory and the `ca-cert.pem` file so that new ones are generated with the keys on the HSM when the server is started.

5.1 Start the Fabric-CA Server

To start the Fabric-CA Server, run the below command.

›_ Console

```
# fabric-ca-server start -b admin:adminpw
```

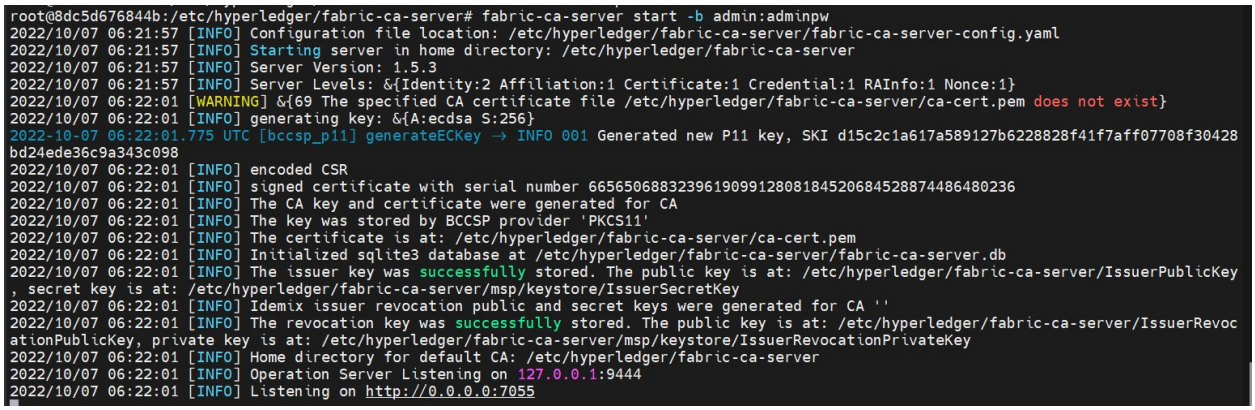

Figure 16 : Fabric-CA start

6 Enroll and Register a Fabric CA Client

1. Edit the `fabric-ca-server-config.yaml` file inside the container and change the identity section as needed.
2. Start the server if it is not currently running. It must be running for the enroll command to work.
3. Create a directory environment variable.

›_ Console

```
# export FABRIC_CA_CLIENT_HOME=$HOME/fabric-ca/clients/admin
```

4. Enroll under the identity in the server YAML file.

›_ Console

```
# fabric-ca-client enroll -u http://admin:adminpw@localhost:7055
```

```
root@8dc5d676844b:/etc/hyperledger/fabric-ca-server# fabric-ca-client enroll -u http://admin:adminpw@localhost:7055
2022/10/07 06:27:49 [INFO] Created a default configuration file at /root/fabric-ca/clients/admin/fabric-ca-client-config.yaml
2022/10/07 06:27:49 [INFO] generating key: &{A:ecdsa S:256}
2022/10/07 06:27:49 [INFO] encoded CSR
2022/10/07 06:27:49 [INFO] Stored client certificate at /root/fabric-ca/clients/admin/msp/signcerts/cert.pem
2022/10/07 06:27:49 [INFO] Stored root CA certificate at /root/fabric-ca/clients/admin/msp/cacerts/localhost-7055.pem
2022/10/07 06:27:49 [INFO] Stored Issuer public key at /root/fabric-ca/clients/admin/msp/IssuerPublicKey
2022/10/07 06:27:49 [INFO] Stored Issuer revocation public key at /root/fabric-ca/clients/admin/msp/IssuerRevocationPublicKey
root@8dc5d676844b:/etc/hyperledger/fabric-ca-server#
```

Figure 17 : Fabric-CA-Client enroll

This command will generate default `fabric-ca-client-config.yaml` configuration file and msp directory. This configuration file will be used to make HSM related configuration changes in the next step. The client is not yet enrolled through the HSM.

5. Edit the client YAML file in your home directory.
6. Delete the `msp` directory and edit the YAML file in the home directory.

7. Edit the **bccsp** section and mirror the same configuration from fabric-ca-server `config.yaml` file for BCCSP section for HSM.

›_ Console

```
# cd /root/fabric-ca/clients/admin

# vim fabric-ca-client-config.yaml
```

```
##############################################################################
bccsp:
    default: PKCS11
    pkcs11:
        Library: /opt/utimaco/lib/libcs_pkcs11_R3.so
        Pin: 123456
        Label: Fabric
        hash: SHA2
        security: 256
        #filekeystore:
            # The directory used for the software file-based keystore
            #keystore: msp/keystore
```

Figure 18 : Fabric-CA-Client config file

8. Run the enroll command again with the server identity.

›_ Console

```
# fabric-ca-client enroll -u http://admin:adminpw@localhost:7055
```

```
root@8dc5d676844b:~/fabric-ca/clients/admin# fabric-ca-client enroll -u http://admin:adminpw@localhost:7055
2022/10/07 06:36:11 [INFO] generating key: &{A:ecdsa S:256}
2022-10-07 06:36:11.484 UTC [bccsp_p11] generateECKey → INFO 001 Generated new P11 key, SKI 8baf100964ec846ead316f01313a622d7
6211710304126ea5e697fd05795ecd6
2022/10/07 06:36:12 [INFO] encoded CSR
2022/10/07 06:36:13 [INFO] Stored client certificate at /root/fabric-ca/clients/admin/msp/signcerts/cert.pem
2022/10/07 06:36:13 [INFO] Stored root CA certificate at /root/fabric-ca/clients/admin/msp/cacerts/localhost-7055.pem
2022/10/07 06:36:13 [INFO] Stored Issuer public key at /root/fabric-ca/clients/admin/msp/IssuerPublicKey
2022/10/07 06:36:13 [INFO] Stored Issuer revocation public key at /root/fabric-ca/clients/admin/msp/IssuerRevocationPublicKey
root@8dc5d676844b:~/fabric-ca/clients/admin#
```

Figure 19 : Fabric-CA-Client enroll

9. Register the client. For example.

›_ Console

```
# fabric-ca-client register --id.name ica.example --id.type client -id.secret
root --csr.names C=es,ST=madrid,L=Madrid,O=example.com --csr.cn ica.example -m
ica.example --id.attrs '"hf.IntermediateCA=true"' u http://localhost:7055 --
loglevel debug
```

```
2022/10/07 06:38:10 [DEBUG] Register { Name:ica.example Type:client Secret:**** MaxEnrollments:0 Affiliation: Attributes:[{hf.
IntermediateCA true false}] CAName:  }
2022/10/07 06:38:10 [DEBUG] Adding token-based authorization header
2022/10/07 06:38:10 [DEBUG] Sending request
POST http://localhost:7055/register
{"id":"ica.example","type":"client","secret":"root","affiliation":"","attrs":[{"name":"hf.IntermediateCA","value":"true"}]}
2022/10/07 06:38:12 [DEBUG] Received response
statusCode=201 (201 Created)
2022/10/07 06:38:12 [DEBUG] Response body result: map[secret:root]
2022/10/07 06:38:12 [DEBUG] The register request completed successfully
Password: root
root@8dc5d676844b:~/fabric-ca/clients/admin#
```

Figure 20 : Register the client

10. Run the enroll command again with the client identity.

›_ Console

```
# fabric-ca-client enroll -u http://ica.example:root@localhost:7055
```

```
root@8dc5d676844b:~/fabric-ca/clients/admin# fabric-ca-client enroll -u http://ica.example:root@localhost:7055
2022/10/07 06:39:44 [INFO] generating key: &{A:ecdsa S:256}
2022-10-07 06:39:44.410 UTC [bccsp_p11] generateECKey → INFO 001 Generated new P11 key, SKI c011fc7e101a12062d3a965a872f50c3b
dc1d63d5602e2416242c1c8a746cc80
2022/10/07 06:39:44 [INFO] encoded CSR
2022/10/07 06:39:52 [INFO] Stored client certificate at /root/fabric-ca/clients/admin/msp/signcerts/cert.pem
2022/10/07 06:39:52 [INFO] Stored root CA certificate at /root/fabric-ca/clients/admin/msp/cacerts/localhost-7055.pem
2022/10/07 06:39:52 [INFO] Stored Issuer public key at /root/fabric-ca/clients/admin/msp/IssuerPublicKey
2022/10/07 06:39:52 [INFO] Stored Issuer revocation public key at /root/fabric-ca/clients/admin/msp/IssuerRevocationPublicKey
root@8dc5d676844b:~/fabric-ca/clients/admin#
```

Figure 21 : Fabric-CA-Client enroll

7 Configure Peer Nodes to use Utimaco HSM

To configure BCCSP to use the Utimaco HSM for peer nodes:

1. Open `core.yaml` file and make the following changes in the **BCCSP** section.

›_ Console

```
BCCSP:

Default: PKCS11

PKCS11:

Library: /opt/utimaco/lib/libcs_pkcs11_R3.so

Pin: 123456 Label: Fabric hash: SHA2 security: 256
```

```
    BCCSP:
        Default: PKCS11
        PKCS11:
            Library:/opt/utimaco/lib/libcs_pkcs11_R3.so
            Label:Fabric
            Pin:123456
            Hash:SHA2
            Security:256
            #FileKeyStore:
            #KeyStore:
```

Figure 22 : core.yaml file

8 Configure Orderer Nodes to use Utimaco HSM

To configure BCCSP to use the Utimaco HSM for orderer nodes:

1. Open `orderer.yaml` file and make the following changes in the **BCCSP** section.

›_ Console

```
BCCSP:

Default: PKCS11

PKCS11:

Library: /opt/utimaco/lib/libcs_pkcs11_R3.so

Pin: 123456 Label: Fabric hash: SHA2 security: 256
```

```
# BCCSP configures the blockchain crypto service providers.
BCCSP:
    Default: PKCS11
    PKCS11:
        Library:/opt/utimaco/lib/libcs_pkcs11_R3.so
        Label:Fabric
        Pin:123456
        Hash:SHA2
        Security:256
        #FileKeyStore:
        #KeyStore:
```

Figure 23 : orderer.yaml file

9 Troubleshooting

Error	Diagnosis
2022/05/27 08:32:16 [DEBUG] No Idemix credential found at /root/fabricca/ clients/admin/msp/user/SignerConfig 2022/05/27 08:32:16 [DEBUG] Register { Name:ica.example Type:client Secret:**** MaxEnrollments:0 Affiliation: Attributes:[{hf.IntermediateCA true false}] CAName: } 2022/05/27 08:32:16 [DEBUG] Adding tokenbased authorization header 2022/05/27 08:32:16 [DEBUG] Sending request POST http://localhost:7054/register {"id":"ica.example","type":"client","secret":"root", "affiliation":"","attrs": [{"name":"hf.IntermediateCA", "value":"true"}]} 2022/05/27 08:32:16 [DEBUG] Received response statusCode=401 (401 Unauthorized) Error: Response from server: Error Code: 20 - Authentication failure	# vim fabric-ca-server-config.yaml We need to change Server's listening port's default value 7054, 9443 to port value to 7055, 9444 respectively

Error	Diagnosis
2022/06/08 05:12:31 [INFO] Configuration file location: /etc/hyperledger/fabric-caserver/fabric-ca-server-config.yaml 2022/06/08 05:12:31 [INFO] Starting server in home directory: /etc/hyperledger/fabric-caserver 2022/06/08 05:12:31 [INFO] Server Version: 1.5.3 2022/06/08 05:12:31 [INFO] Server Levels: &{Identity:2 Affiliation:1 Certificate:1 Credential:1 RAInfo:1 Nonce:1} Error: Failed to get BCCSP with opts: Could not find BCCSP, no 'PKCS11' provider	After executing GO_TAGS=pkcs11 make fabricca-server, remember to remove the original fabric-ca-server binary and put the newly built binary in `path/to/go/bin` After executing GO_TAGS=pkcs11 make fabricca-client, remember to remove the original fabricca-client binary and put the newly built binary in `path/to/go/bin`

Table 6: List of errors and their diagnoses

10 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.

11 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

Table 7: References

12 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.