CypherShield Key Certification Server Guide

Table of Contents

Introduction

The CypherShield Key Certification Server provides client to client digital signature authentication services to customer applications using the CypherShield Data-in-Motion Data Encryption SDK. This service is to prevent session hijack security breaches. One example of this threat is Man-in-the-Middle (MITM) attacks.

Network architectures which use the CypherShield technologies encapsulated inside TLS do not require the CypherShield Key certification server explicitly. TLS provides authentication protection via Public/Private PKI certificates.

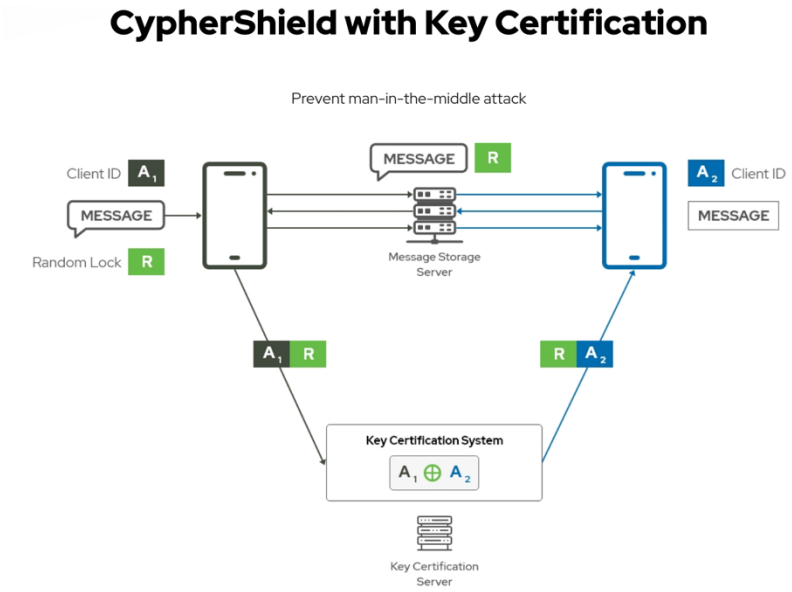
TLS is not used nor appropriate in many network architectures including

- High-Performance private networks
- Wireless architectures including Bluetooth, RFID, IoT and others
- Internal high-security private communications

The database that is part of the key certification server contains no identifying information and so does not need to be encrypted directly. The database should be secured and replicated to prevent potential ransom-ware attacks.

Architecture

The below diagram outlines the overall architecture of the key certification server.



Infrastructure Features

- Containerized Infrastructure
- NGINX for SSL processing, scale-out and high-reliability services
- Scale, HA (Redis)
- Local DB - Temporary Cache Storage

Getting Started

The server is deployed as a complete containerized environment running inside a single VM or hardware server node. This environment uses the docker-compose tool.

This environment is deployed by running a single script on the server or VM.

This script will:

1. Install the prerequisite packages including node.js, docker, etc.
2. Build configured a **`docker-compose.yml`** file
3. Create temporary certificates (if needed)
4. Install and configure NGINX
5. Pull the container images from the Docker public repository

This script is named authgen.sh and is distributed with this package.

Create Server

1. Create a Linux virtual machine or allocate a hardware server
2. Install one of the supported Linux operating systems

   Ubuntu version 20+

   Redhat or CentOS version 8+
3. Update your DNS system with this server name
4. Open ports

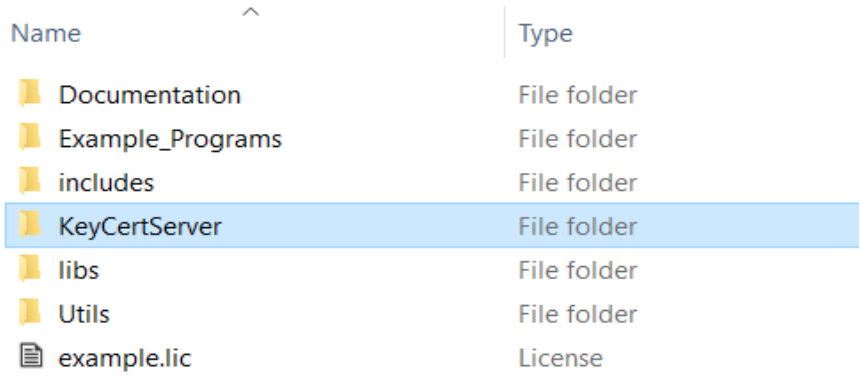
```
22 – ssh
80 – http
443 – https
4200 – key certification server
```

5. Download authgen.sh script to this server at **`/srv`**

   Change permissions to execute

```
chmod +x authgen.sh
```

6. The scripts for deploying this server are included with the CypherShield SDK package.in the KeyCertServer folder.



Copy the scripts from the KeyCertServer folder to **/srv** on your new host server.

Add Server PKI Certificates

Add PKI certificates to this new server. Public Key Infrastructure certificates are required for the Key Certification Server during deployment.

> *Note: if the server certificates are not inserted, the authgen.sh script will create free certificates from LetsEncrypt automatically. These certificates expire and the customer will be required to either replace the certificates with their own certificates or configure LetsEncrypt to renew these certificates automatically.*

To install a customer provided certificate, complete the following step:

1 CD to the **/srv** folder on your new server

2 Add your certificate PEM files to this folder and use the file names below.

/srv/fullchain.pem

/srv/privkey.pem

> *Note: These certificate files will be moved during deployment to a shared location between the running containers and the NGINX front-end system.*

Add CypherShield Keys and Certificates

The CypherShield SDK includes a command line utility (genkeypair) to generate customer unique keys and certificate files. These files should be generated and copied to the Key Certification host instance.

Note: Refer to the SDK documentation included in the CypherShield Encryption SDK package documentation

These security files include public and private key files which are common to the CypherShield client implementations as well as the Key Certificate Server and provide additional security between client-to-client communications as well as client to Key Certificate Servers.

There should be four Key and Certificate files generated.

Copy these files to the new server at **/srv**

Deploy the CypherShield Server Environment

Execute the **authgen.sh** script to deploy all prerequisites and build the containerized application stack.

The **authgen.sh** script needs only a few parameters

User **-h** to show the parameters:

```
/srv/authgen.sh -h
Usage: ./authgen.sh
-H STRING HOST
```

```
-D STRING DOMAIN
-W OPTION Enable WatchTower in docker
```

WatchTower is an optional service which will monitor for updates to the docker images and automatically install the latest releases.

Add **-W** to the command if this is desired for your environment.

Example:

```
./authgen.sh -H YourServer -D YourDomain.com [-W]

-H your server's name
-D Domain Name
-W is for WatchTower, this will auto update the docker image and should
only be used in development.
```

After the deployment script completes, you should have a fully functioning Key certification server environment.

Execute the command: **/srv/manage ps**

```
NAMES                   CREATED                 STATUS
srv_backend_1           2 months ago      Up 2 months
srv_watchtower_1        2 months ago      Up 2 months
srv_redis_1             2 months ago      Up 2 months
```

This will display the running containers.

If the containers are created running, the server is installed and available for use.

Server Application Management

The SDK package includes a management script. This should have been copied to the server host instance at **/srv**.

/srv/manage will allow you to see the running docker image, stop, start, and refresh the running container images.

Usage:

options: stop | start | ps | kill | build | rebuild

stop - stop Docker
start - start Docker
ps - shows Docker running processes
kill - stops Nginx and Cleans out Docker's containers and volumes
build - refreshes the Docker image from the repository and start Nginx and Docker
rebuild - stops Nginx and docker, does a refresh, and starts Nginx and Docker