Diamond Key Security

Diamond-HSM™ Prototype Manual

Change Log

Date	Description
11/28/2018	Initial document.
12/17/2018	Added LED description and detailed information on console commands.
12/19/2018	Terminology fixes and removed ‘DRAFT’ watermark.

Contents

1. Introduction

Diamond Key Security (DKS) is bringing to market a unique set of security products that are designed and built upon the open and transparent CrypTech technology for a multitude of enterprise and individual applications. DKS is utilizing open source software from CrypTech and likely other sources to create revolutionary security solutions. The CrypTech project initiative has succeeded in building a generalized module for cryptography functions that underpin secure communications and Internet technologies. The Diamond-HSM™ has been built using multiple CrypTech devices. This manual lists the features of the Diamond-HSM and explains how to set it up for use on an internal network.

All procedures have been tested and refined by Diamond Key Security, NFP. For a complete list of references please see section A, References and Endnotes.

1.1. Acknowledgements

The CrypTech project commenced development in 2014. The CrypTech Alpha was released in July 2016 with a substantial base of completed code. The Alpha and all the developed code have been well documented by the CrypTech development team. This document makes use of that documentation and we would like to acknowledge the good work that has gone into making that code and its accompanying documentation clear and useful. We take responsibility for the contents of this document.

1.2. Scope of Document

The Diamond-HSM is a physical device that can manage digital keys for strong authentication and cryptoprocessing[vi]. DNSSEC is a suite of security extension for the Domain Name System used to resolve domain names to IP addresses on the Internet. This document will explain how to setup a new Diamond-HSM on the Linux operating system. A PKCS #11 (Public Key Cryptography Standards #11) library has been included to make the HSM compatible other software such as DNS servers using DNSSEC. Examples of this include BIND, Open DNSSEC, Power DNS, and Knot DNS. Please see the documentation for those systems for information on how to connect a PKCS #11 library.

1.3. Organization of This Document

This document is organized into sections. It first introduces Diamond Key Security and the CrypTech. It also defines important terms and provides special notes that may be needed to understand the hardware and software to be able to correctly set up the HSM. Next it gives more details and configuring the HSM using the 'dks_setup_console' application. Finally, it gives sample instruction on how to use the HSM with some common DNS servers for DNSSEC.

1.4. Definitions

BIND – Berkeley Internet Name Domain. BIND is open source software for publishing DNS information on the Internet, and to resolve DNS queries from users. The software originated in the early 1980s at the University of California at Berkeley.[i]

Bootloader – The bootloader is a program in the HSM that runs when the HSM is powered on. It starts the firmware and also is used to upgrade the firmware.

CrypTech – CrypTech is a loose international collective of engineers trying to improve assurance and privacy on the Internet. [ii]

DNS – Domain Name System. DNS is a decentralized system for naming computers and other resources on the Internet. It maps domain names to numerical IP address which are used to identify computer services and devices on the Internet.[iii]

DNSSEC – Domain Name System Security Extensions. DNSSEC is a suite of Internet Engineering Task Force (IETF) specifications for securing certain kinds of information provided by the Domain Name System as used on IP networks. It is an extension to DNS which provides DNS clients(resolvers) origin authentication of DNS data, authenticated denial of existence, and data integrity, but not availability or confidentiality.[iv]

Firmware – The firmware is the program on the HSM that is used for the hardware cryptographic engine.

Git – Git is a free and open source distributed version control system. [v]

HSM – A hardware security module (HSM) is a physical computing device that safeguards and manages digital keys for strong authentication and provides cryptoprocessing[vi].

IETF – Internet Security Engineering Task Force

OpenDNSSEC – OpenDNSSEC is a policy-based zone signer that automates the process of keeping track of DNSSEC keys and the signing of zones.[vii]

OpenSC – OpenSC provides a set of libraries and utilities to work with smart cards. Its main focus is on cards that support cryptographic operations and facilitate their use in security applications such as authentication, mail encryption and digital signatures. OpenSC implements the standard APIs to smart cards, e.g. PKCS#11 API, Windows' Smart Card Minidriver and MacOS Tokend. [viii]

PKCS 11 – Public-Key Cryptography Standards #11. The PKCS #11 standard defines a platform-independent API to cryptographic tokens, such as HSMs and smart cards.

Python – Python is an interpreted high-level programming language for general use programming. [ix] Much of the CrypTech software was written in Python 2.

2. Diamond-HSM Description and Functionality

The Diamond-HSM is an open-source module used to store cryptographic information and perform cryptographic functions. The HSM contains multiple CrypTech devices connected by a single-board computer inside a single secure case. It is tamper-resistant and if an intrusion occurs, it can clear its private data. There is an ethernet port for connecting to the HSM over s secure TLS TCP/IP connection.

2.1. CrypTech Devices

The Diamond-HSM contains two internal CrypTech devices to share the workload and provide redundancy. The CrypTech devices provide a secure way to store keys, sign and encrypt data, generate true random numbers, and authenticate HSM users. Each device has three main parts:

The FPGA Sub System – Xilinx XC7A200T
FPGA stands for Field Programmable Gate Array and the alpha board uses one to implement CrypTech crypto/security cores accessible by the CPU as coprocessors.

ARM Processor – STM32F42
The ARM processor talks to host systems and handles incoming commands. The Diamond-HSM's single-board computer communicates directly with the ARM processor which then communicates with the FPGA.

Tamper Circuit – ATtiny828
The tamper circuit is responsible for implementing tamper detection and control/alarm as a separate functionality from the CPU. On the Alpha board this system is fairly simplistic. When a tamper event has been detected, with will clear the Master Key Memory (MKM). Tamper events are detected by the Diamond Sensor card. [x]

2.2. Diamond Sensor Card

Each CrypTech device has a Diamond Key Security sensor card attached to its GPIO port. The sensor card can detect physical tampering of the HSM and will signal the tamper circuit to clear the Master Key Memory (MKM). The sensor card responds to the following tamper events:

- Case open
- Change in ambient light
- Change in temperature
- Motion

2.3. User-roles

The user-roles and authentication are performed on each CrypTech device. Because of this, the Diamond-HSM supports the three user-roles defined by CrypTech: ‘wheel’, ‘so’, and ‘user’. ‘so’ and ‘user’ are PKCS#11 defined user types, and ‘wheel’ is used to control the HSM. The passwords for all user-roles are kept synchronized on each CrypTech device by the Diamond-HSM’s single-board computer.

2.3.1. ‘wheel’

The ‘wheel’ user-role is used to control the HSM. Diamond Key Security has provided a setup console to configure the HSM. The ‘wheel’ role must be used to login to the console and must be reentered when performing special task such as: updating the internal HSM software and updating the bootloader or firmware of a CrypTech device. On every restart of the HSM, a user must login to the HSM at least once to unlock it and make it available for use. ‘wheel’ can be used to change all user PINs.

2.3.2. ‘so’

The ‘so’ role is the “security officer” role as defined in PKCS#11. The ‘so’ role can only be used to set the ‘user’ PIN.

2.3.3. ‘user’

The ‘user’ role is also a PKCS#11 role. When an application uses PKCS#11 to access the HSM, the ‘user’ PIN is always used.

2.4. Master Key Memory

Internally, each CrypTech device uses a 256-bit master key to lock all of its contents. The master key is stored in volatile memory. To prevent the master key from being accidentally cleared when the HSM is moved or during a power failure, there is a battery backup. The master key should be generated when the HSM is initially configured. In a tamper event, the MKM will be cleared rendering all data in the HSM useless. If you would like to be able to recover from a tamper event, the 256-bit master key should be written down and kept in a safe place, such as another HSM or in a safe. To recover the data, the master key can be reentered in the setup console. If you would like the data to not be recoverable, do not save the generated master key.

If the master key is ever re-entered, it must be identical to the previous master key in order to recover the data. The HSM does not store a history of previous master keys so if an incorrect master key is used, the results will be undefined.

3. Diamond-HSM™ Prototype Setup

3.1. Physical HSM Installation

The Diamond-HSM is a 1U rack-mountable network appliance. On the back of the unit, there is an ethernet port, a power connector, a power switch, and intake/exhaust fans. On the front of the unit are 3 LEDs, and a USB port. Because of the weight of the Diamond-HSM, shelf rails will need to be purchased separately and installed in the rack. To install the HSM, connect a power cord to the power connector and an ethernet cable to the ethernet port.

3.2. Installing the Diamond-HSM Software Package

After receiving an HSM from Diamond Key Security, users will receive a package from Diamond Key with all the necessary software for the host computer. The package has been built for the Linux operating system. The software used to connect to the Diamond-HSM shown in diagram 3.2.

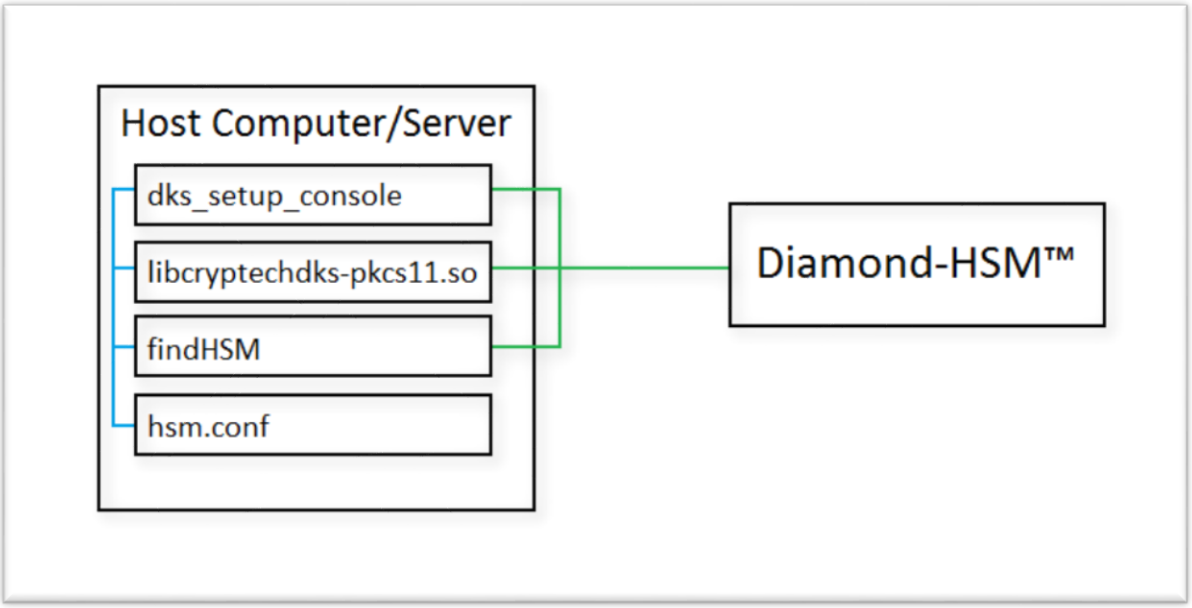


Diagram 3.2. – Diamond-HSM Software Configuration

The package includes the following files:

- ‘dks_setup_console’
  ‘dks_setup_console’ connects to the HSM over an ethernet connection. It uses the IP address of the HSM listed in the file ‘hsm.conf’ which should be placed in the ‘/etc/dkey/hsm/’ folder on Linux. Whenever the HSM restarts, ‘dks_setup_console’ must be used to log into the HSM.

- ‘findHSM’
  If the HSM has been setup to use DHCP, ‘findHSM’ will find the HSM on the network using ‘Zero Configuration Networking’ (Zeroconf) and then write the configuration to ‘/etc/dkey/hsm/hsm.conf’. Once the HSM has been setup, Zeroconf can be disabled. This is a Python script.

- ‘hsm.conf’
  ‘hsm.conf’ is a configuration file that tells the IP address of the HSM on the network. For static IP address without Zeroconf, ‘hsm.conf’ should be modified manually.

- ‘install.sh’
  Simple installation script to copy files to the correct locations on a host computer or server. This script will also install python pip and zeroconf.

- ‘libcryptechdks-pkcs11.so’
  This is an update version of the PKCS#11 library provided by CrypTech that will connect to the Diamond-HSM using a secure TLS connection. It uses the IP address in ‘/etc/dkey/hsm/hsm.conf’ to locate the HSM.

- ‘DKEY-HSM-UPDATE-yy.mm.vv.tar.gz.signed’
  This is updated Diamond-HSM firmware that has been signed by Diamond Key Security.
  Note: Not all software packages will include a Diamond-HSM firmware update.

Once all the files have been extracted, run the install.sh script from the command line. It will copy the files to the necessary folders.

```
$ sudo ./install.sh
```

3.3. Initial connection using DHCP

The Diamond-HSM has support for DHCP and will automatically receive an IP address from a DHCP server. Once the HSM has fully booted up, the LEDs will all be on. This means the HSM also has a valid IP address. To find the HSM use the ‘findHSM’ utility.

```
$ sudo findHSM
```

The utility will list all Diamond-HSMs on the network and their serial numbers. After selecting the correct HSM, the utility will write to ‘/etc/dkey/hsm/hsm.conf’. The HSM has 2 ethernet ports, one for console management request (CTY), and the other is for running HSM operations (RPC). When multiple HSM are available, please select the same HSM for both CTY and RPC.

3.4. Initial connection without using DHCP

To connect to the HSM without using DHCP, a host computer needs to connect to the HSM using a crossover cable, or an unmanaged hub or switch. The HSM will take longer to start because it will wait for DHCP for about one minute. When it has determined that DHCP is not available, it will default to the following static IP configuration

```
IP: 10.10.10.2
Net mask: 255.255.255.0
```

The host computer will need to be setup with a compatible static IP address. The default hsm.conf given in the installation package has been preset to point to an HSM at this static address.

3.5. First time setup using 'dks_setup_console'

Once '/etc/dkey/hsm/hsm.conf' has been configured with the IP address of the HSM, 'dks_setup_console' can now access it.

```
$ dks_setup_console
```

'dks_setup_console' connects to the HSM's console interface. The Diamond-HSM uses the built-in CrypTech devices for all user authentication and uses the default CrypTech password, 'YouReallyNeedToChangeThisPINRightNowWeAreNotKidding'. CrypTech devices also have a special 'wheel' built-in user account. To access the Diamond-HSM's console, the 'wheel' user password must be used. It can be changed after logging into the HSM. Whenever the HSM restarts, 'dks_setup_console' must be used to log into the HSM.

Once you have logged into the HSM, the HSM will check the available CrypTech devices.

3.6. Setting the Master Key

The master key can be set using the 'dks_setup_console'. When logging into the HSM, after a restart, the HSM will ask if the user wants to enter the master key. If the master key has already been set, simply answer no ('n'). To recover data after a tamper event or power failure, you must reenter the master key exactly as it was before. To randomly generate a new master key, leave it blank and press 'enter'. The generated master key will be shown. The generated master key will only be shown this one time so if the user wants to record it, it must be done at this time. The master key can also be set using the 'masterkey set' command from the setup console. If used without arguments, 'masterkey set' will generate a random master key, otherwise the full 256-bit master key should be given in hexadecimal notation.

```
Diamond-HSM > masterkey set


Diamond-HSM > masterkey set 01234567 89ABCDEF 01234567 89ABCDEF 01234567 89ABCDEF 01234567 89ABCDEF
```

The master key can usually be set immediately. Because the master key is synchronized between all internal CrypTech devices, it may sometimes take multiple attempts to set the master key.

3.7. Changing User PINs

The user PINs should be changed immediately after logging into the HSM. A factory reset HSM will only have the default 'wheel' password set. The default password is 'YouReallyNeedToChangeThisPINRightNowWeAreNotKidding'. The 'so' and 'user' PINs are initially undefined.

To change a pin in 'dks_setup_console' use the 'set pin' command.

```
Usage: set pin <user> <pin>


Diamond-HSM > set pin wheel password123456
Diamond-HSM > set pin so password1234
Diamond-HSM > set pin user 1234
```

If the 'wheel' password is lost, the HSM will be permanently locked.

3.8. Set the HSM to use a Static IP or DHCP

The Diamond-HSM supports both DHCP and static IP ethernet interfaces. Using a DHCP server is recommended and if a static IP address is needed, the DHCP can be set to give a fixed IP address based on the HSM's MAC address. To get the HSM's MAC address, use the 'show macaddr' command from 'dks_setup_console'. Once the DHCP server has been configured, reboot the HSM.

3.8.1. Configuring a static IP manually on the HSM.

If it is not possible to configure the HSM to use a fixed IP address from a DHCP server, a static IP can be set using 'dks_setup_console' using the 'set ip static' command and following the prompts.

3.8.2. Configuring the HSM to use DHCP.

The HSM will use DHCP by default with a fallback static IP configuration. If the HSM has been set to use a static IP address, the command, 'set ip dhcp', can be used to change the HSM, back to the default DHCP settings.

3.9. Updating the HSM Software

All updates from Diamond Key Security have been digitally signed using an RSA 2048-bit private key and a SHA 512 hash. The public key is stored on the HSM, so all updates can be verified before installing the data on the HSM. Updates have also been dated to help administrators know if an update is needed. The current HSM software version can be found using the 'show version' command.

```
Diamond-HSM > show version
```

```
2018-11-28-02 - 0.0.0.2


Diamond-HSM >
```

If the current version date is before the date of the signed update inside of the installation package, you should update the software. To update the HSM, used the ‘update HSM <update path>’ command. The update HSM command prepare the HSM for a file transfer.

```
Diamond-HSM > update HSM /home/admin/Downloads/DKEY-HSM-UPDATE-18.12.b1.tar.gz.signed
```

Once the HSM is ready, it will request the file from ‘dks_setup_console’. After updating the HSM, you will need to restart the HSM for the changes to take effect. After restarting the HSM, you will need to login to the HSM using ‘dks_setup_console’ but will not need to reset the master key and can ignore the prompt.

4. Detailed HSM Configuration

This section is for advanced configuration of the Diamond-HSM. It assumes that steps for installing the HSM as described in section 3 have been completed.

4.1. Configuration File – hsm.conf

Connections to the Diamond-HSM is done using an ethernet connection. 'hsm.conf' is a configuration file that tells the IP address of the HSM on the network and can be found at the path '/etc/dkey/hsm/hsm.conf'. For static IP address without Zeroconf, 'hsm.conf' should be modified manually. The following is a sample 'hsm.conf' file.

```
<?xml version="1.0" encoding="utf-8" ?>
<diamondhsm>
  <cty>
    <IP>10.1.10.194</IP>
    <port>8081</port>
    <servername>dks-hsm</servername>
    <serial>1000-0002</serial>
  </cty>
  <rpc>
    <IP>10.1.10.194</IP>
    <port>8080</port>
    <servername>dks-hsm</servername>
    <serial>1000-0002</serial>
  </rpc>
</diamondhsm>
```

'hsm.conf' is in an XML file format. There are two nodes, 'cty' and 'rpc'. The 'cty' connection is used by the management interface and uses port 8081. The 'rpc' connection is used by PKCS #11 and listens for connections on port 8080.

When manually configuring 'hsm.conf', only the IP address and serial number should be updated. All other options have been programmed into the Diamond-HSM and are fixed.

4.2. 'dks_setup_console' complete command list

The 'dks_setup_console' has been provided to allow HSM administrators to be able to connect to the Diamond-HSM's management system over an ethernet connection using a console interface. A detailed list of the available console commands follows.

4.2.1. keystore erase YesIAmSure [preservePINs]

The 'keystore erase' will erases the keystore. If 'preservePINs' is used, the PINs will not be reset.

```
Diamond-HSM > keystore erase YesIAmSure
```

4.2.2. list keys

To list all the keys in the HSM by the CrypTech device UUID, execute the command ‘list keys’.

```
Diamond-HSM > list keys
```

4.2.3. masterkey set

Use ‘masterkey set’ to set the master key on all internal CrypTech devices. The master key will be synchronized across all CrypTech devices. Because of the synchronization procedure, the master key may not be set on the first attempt. If the master key has not been set, the HSM will respond with an error. If this happens, please wait about 45 seconds and then try again.

The master key can be entered by the user by entering 64 hexadecimal digits. For a random master key, use the command without any options.

```
Diamond-HSM > masterkey set


Diamond-HSM > masterkey set 01234567 89ABCDEF 01234567 89ABCDEF 01234567 89ABCDEF 01234567 89ABCDEF
```

4.2.4. restore [preservePINs] [preserveKEYs]

The ‘restore’ command will restore the HSM to its default factory settings. This will not remove any HSM updates. If the preservePINs option is not used, the ‘user’ and ‘so’ pins will be deleted, and the ‘wheel’ pin will revert to ‘YouReallyNeedToChangeThisPINRightNowWeAreNotKidding’.

```
Diamond-HSM > restore preservePINs
```

4.2.5. set ENABLE_EXPORTABLE_PRIVATE_KEYS <TRUE or FALSE>

Set the ENABLE_EXPORTABLE_PRIVATE_KEYS flag. Use TRUE to create private keys with the exportable flag so key wrapping can be used to back up the private keys. The default value is FALSE. Only private keys with the EXPORTABLE flag set to TRUE can be mirrored to other CrypTech devices inside the HSM. If a key is not portable, there is no way for it to ever be extracted off the CrypTech device. This flag only affects keys generated after it has been set. The HSM currently only allows internal backing up of keys.

```
Diamond-HSM > set ENABLE_EXPORTABLE_PRIVATE_KEYS TRUE
```

4.2.6. set ip <DHCP or STATIC>

The set ip command will allow the user to change the HSM’s IP address settings. This will require the HSM to reboot. After this reboot, the master key will not need to be reset.

```
Diamond-HSM > set ip DHCP
```

```
Diamond-HSM > set ip STATIC
```

4.2.7. set pin <user>

The ‘set pin’ command can be used to set the 'wheel', 'user', or 'so' pin. After executing the command, the HSM will prompt the user to enter the new pin.

```
Diamond-HSM > set pin wheel
```

```
Diamond-HSM > set pin user
```

```
Diamond-HSM > set pin so
```

4.2.8. show devices

The ‘show devices’ command will show a list of all CrypTech devices inside the HSM. It can be used to diagnose problems connecting to the CrypTech devices.

```
Diamond-HSM > show devices
```

4.2.9. show ipaddr

‘show ipaddr’ shows the IP4 IP address of the HSM

```
Diamond-HSM > show ipaddr
```

4.2.10. show macaddr

‘show macaddr’ shows the MAC address of the HSM

```
Diamond-HSM > show macaddr
```

4.2.11. show time

Show's the current time set in the HSM.

```
Diamond-HSM > show time
```

4.2.12. show version

Use ‘show version’ to see the software version installed on the HSM’s single-board computer.

```
Diamond-HSM > show version
```

4.2.13. show serial-number

Use ‘show serial-number’ to see the serial number of the HSM.

```
Diamond-HSM > show serial-number
```

4.2.14. sync cache

To speed up key look up on the internal CrypTech devices, the Diamond-HSM keeps a table of all of the keys. To rebuild the cache, uses ‘sync cache’. In most circumstances, this command will not be needed.

```
Diamond-HSM > sync cache
```

4.2.15. update firmware

This command will update the firmware on the internal CrypTech devices and is not recommended under most circumstances. The firmware used for the update will be the CrypTech binaries already loaded within the Diamond-HSM so in most cases, this command will reinstall the same firmware on the CrypTech devices. Failure during the update process can cause a CrypTech device to become nonoperational. Diamond Key Security will specify when to update the CrypTech device firmware. The ‘wheel’ password will need to be reentered.

```
Diamond-HSM > update firmware
```

4.2.16. update bootloader

This command will update the bootloader on the internal CrypTech devices and is not recommended under most circumstances. The bootloader used for the update will be the CrypTech binaries already loaded within the Diamond-HSM so in most cases, this command will reinstall the same bootloader on the CrypTech devices. Failure during the update process can cause a CrypTech device to become nonoperational. Diamond Key Security will specify when to update the CrypTech device bootloader. The ‘wheel’ password will need to be reentered.

```
Diamond-HSM > update bootloader
```

4.2.17. update HSM

The update HSM command is used to send a signed update file to the HSM. The update will include all of the software that’s executed on the HSM’s internal single board computer. It will also include the binary files for the firmware, bootloader, and FPGA bitstreams for the the internal CrypTech devices. To update the HSM the full path of the update file must be given. The ‘wheel’ password will need to be reentered.

```
Diamond-HSM > update HSM /home/dkey/Downloads/DKEY-HSM-UPDATE-18.12.b1.tar.gz.signed
```

4.2.18. update fpga

This command will update the FPGA bit stream on the internal CrypTech devices and is not recommended under most circumstances. The FPGA bit stream used for the update will be the CrypTech binaries already loaded within the Diamond-HSM so in most cases, this command will reinstall the same FPGA bit stream on the CrypTech devices. Failure during the update process can cause a CrypTech device to become nonoperational. Diamond Key Security will specify when to update the CrypTech device FPGA bit stream.

```
Diamond-HSM > update fpga
```

5. HSM LEDs

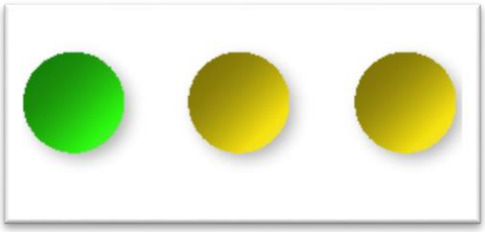
The LEDs on the Diamond-HSM have been designed to provide feedback on the internal state and health of the HSM.

5.1. Power-up Sequences

When the Diamond-HSM is powered-up or rebooted, it will go through a specific set of steps, the can be followed by watching the three LEDs on the front panel. The amount of time the HSM will stay in each stage is relatively fixed so delays may reveal a problem in the way the HSM has been configured.

5.1.1. Power ON

POWER SYSTEM TAMPER

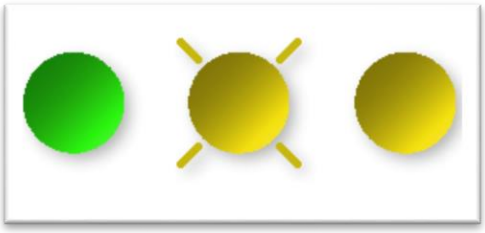


When the HSM is first powered up, the LEDs will all turn on in this base configuration. At startup, this may last for 30 to 90 seconds as the OS boots. If the HSM has been configured to use DHCP, but a DHCP server cannot be found, the HSM will stay in this state for an additional 90 seconds while it waits for a DHCP server. If DHCP fails, the HSM will fall back to the static IP address 10.10.10.2.

NOTE: The HSM will also return to this state if the 'shutdown YesIAmSure' command is used from the console.

5.1.2. Determining Network Configuration

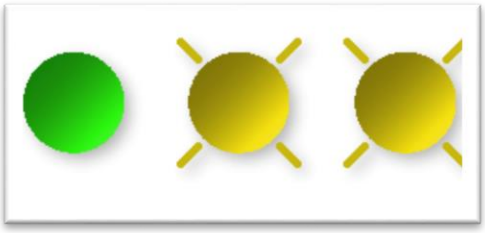
POWER SYSTEM TAMPER



This LED configuration happens just after the HSM's internal single-board computer software starts. The HSM will check its network configuration so it can be used by internal processes.

5.1.3. Checking CrypTech Devices

POWER SYSTEM TAMPER

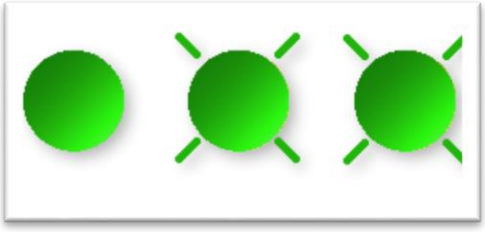


When the system and tamper lights flash yellow, the HSM has started to check it's connection to the CrypTech devices. In most cases, this will only last a few seconds. If the lights continue to flash yellow, the HSM is having difficulties communicating with the CrypTech devices.

Troubleshooting Issues: If the HSM has problems detecting a CrypTech device on startup, wait 90 seconds, then power off the HSM using the power switch. Wait 60 seconds, and then restart the HSM. After all the lights turn green, wait an additional 60 seconds before accessing the HSM.

5.1.4. Starting TCP Servers

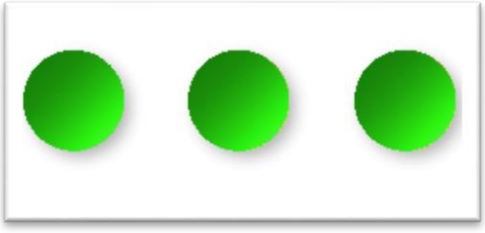
POWER SYSTEM TAMPER



All external connections to the HSM are over TCP connections. Once the HSM has checked it's internal state, it will listen to TCP connections on three network ports. The HSM will stay in this state for about 30 seconds to allow the CrypTech devices to fully start up.

5.1.5. System Ready

POWER SYSTEM TAMPER



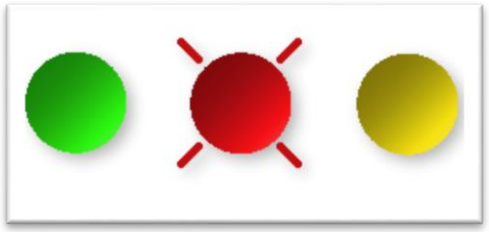
When all of the LEDs are solid green, the HSM has started correctly and is ready for use.

5.2. HSM Problem States

The following states occur when there is an internal error or a tamper event.

5.2.1. Failure on all CrypTech Devices

POWER SYSTEM TAMPER

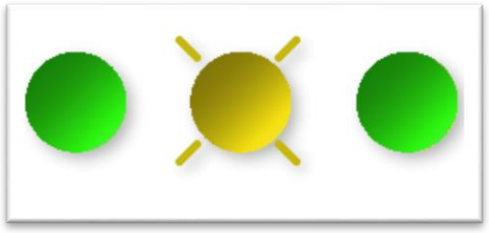


In some events, the Diamond-HSM will not be able to communicate with any of it's internal CrypTech devices. When this happens, cryptographic operations will not be available.

Troubleshooting Issues: As described in section 5.1.3, this failure state should be handled the same way as a delay in detecting the CypTech devices. First wait 90 seconds, and then power off the HSM using the power switch. Wait 60 seconds, and then restart the HSM. After all the lights turn green, wait an additional 60 seconds before accessing the HSM.

5.2.2. Failure on one CrypTech Device

POWER SYSTEM TAMPER



In some events, the Diamond-HSM will not be able to communicate with one internal CrypTech devices. When this happens, cryptographic operations will be available, but keys on the malfunctioning CrypTech device will not be available.

Troubleshooting Issues: As described in sections 5.1.3, this failure state should be handled the same way as a delay in detecting the CypTech devices. First wait 90 seconds, and then power off the HSM using the power switch. Wait 60 seconds, and then restart the HSM. After all the lights turn green, wait an additional 60 seconds before accessing the HSM.

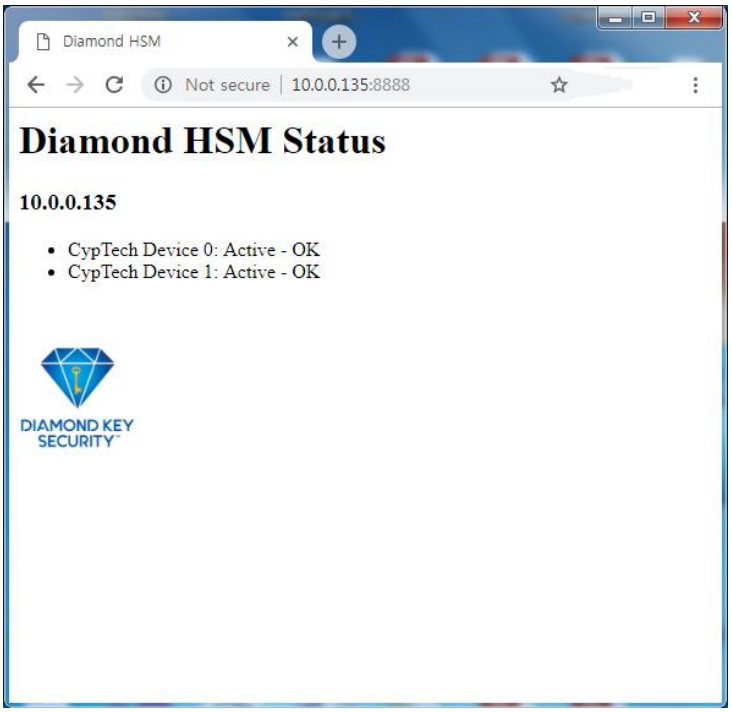
5.2.3. Tamper Event Detected

POWER SYSTEM TAMPER

When the Diamond-HSM detects a tamper event the system and tamper LEDs will flash red. During the tamper event, the master key memory will be continuously erased causing all cryptographic operations to fail.

6. Web browser Interface

The Diamond-HSM has a browser interface that can be used to check the status of the HSM. To access it, type the IP address of the HSM into a web browser with port ‘80’.



7. OpenDNSSEC Support

This section assumes that Diamond-HSM has been configured according to section 3 of this document. Please see sections 3 if you have not already done so. OpenDNSSEC and OpenSC also need to be installed. Please consult the OpenDNSSEC documentation for the most recent instructions for configuring OpenDNSSEC to work with an HSM. The following instructions are not meant to be final.

On Debian and Ubuntu Linux, the following command can be run to install OpenDNSSEC and OpenSC.

```
apt-get install opendnssec opensc
```

7.1. Create a Sample OpenDNSSEC Zone File

```
mkdir /var/lib/opendnssec/cryptech


cat > /var/lib/opendnssec/unsigned/example.com << EOF
\$TTL 600
example.com.  IN SOA  hidden-master.example.com. hostmaster.example.com. (
                                2016041401 ; serial
                                720        ; 28800      ; refresh (8 hours)
                                720        ; 7200       ; retry (2 hours)
                                300        ; 604800     ; expire (1 week)
                                120        ; 3600       ; minimum (1 hour)
                                )

               NS      lab.cryptech.is.
test           A       127.0.0.1
EOF


chown -R opendnssec: /var/lib/opendnssec/*
```

7.2. OpenDNSSEC configuration changes

The following changes need to be made to the OpenDNSSEC configuration files.

/etc/opendnssec/conf.xml:

```
<Repository name="Cryptech">
        <Module>/usr/lib/libcryptechdks-pkcs11.so</Module>
        <TokenLabel>Cryptech Token</TokenLabel>
        <PIN>1234</PIN>
        <SkipPublicKey/>
```

```
</Repository>
```

The PIN is whatever was chosen as PIN for 'user' above. The TokenLabel must be "Cryptech Token" and cannot be changed.

/etc/opendnssec/kasp.xml:

In the file '/etc/opendnssec/kasp.xml', under the 'Keys' section for the policy, the repository needs to be set to 'Cryptech'. If there are multiple policies that will use the Cryptech repository, this change must be made in each one.

```
<!-- Parameters for KSK only -->
<KSK>
        <Algorithm length="2048">8</Algorithm>
        <Lifetime>P1Y</LifeTime>
        <Repository>Cryptech</Repository>
</KSK>

<!-- Parameters for ZSK only -->
<ZSK>
        <Algorithm length="1024">8</Algorithm>
        <Lifetime>P90D</Lifetime>
        <Repository>Cryptech</Repository>
        <!-- <ManualRollover/> -->
</ZSK>
```

/etc/opendnssec/zonelist.xml:

```
<Zone name="example.com">
        <Policy>lab</Policy>
        <SignerConfiguration>/var/lib/opendnssec/signconf/example.com.xml</SignerConfiguration>
        <Adapters>
                <Input>
                        <Adapter type="File">/var/lib/opendnssec/unsigned/example.com</Adapter>
                </Input>
                <Output>
                        <Adapter type="File">/var/lib/opendnssec/signed/example.com</Adapter>
                </Output>
        </Adapters>
</Zone>
```

7.3. Initialization and signing

Initialize opendnssec:

```
$ ods-hsmutil setup
```

Make the daemons reload their configuration:

```
$ service opendnssec-enforcer restart
$ service opendnssec-signer restart
```

That should be it!
See /var/log/syslog for output from ods-kaspcheck, ods-enforcerd and ods-signerd. See /var/lib/opendnssec/signed/ for a signed example.com zone.
To list keys using ods-ksmutil, accessing the HSM using pkcs11 directly (rather than going through any of the opendnssec daemons), export the environment variables from /etc/default/opendnssec and run "ods-ksmutil keys list --verbose":

```
# ods-ksmutil keys list --verbose
SQLite database set to: /var/lib/opendnssec/kasp.db
Keys:
Zone:                            Keytype:      State:    Date of next transition (to):  Size:
Algorithm:  CKA_ID:                            Repository:                      Keytag:
example.com                      KSK           ready     waiting for ds-seen (active)   2048    8
7f9b9329480ebe5dc81054ccb293e261  Cryptech                          62642
example.com                      ZSK           active    2016-07-13 19:04:30 (retire)   1024    8
97e972633613bd605944a0531ff5399b  Cryptech                          56620
```

If the output for repository is "Cryptech NOT IN repository", ods-hsmutil has not been able to actually list the keys in the HSM.

8. BIND 9.12 Support

This section assumes that Diamond-HSM has been configured according to section 3 of this document. Please see sections 3 if you have not already done so. Please consult the BIND documentation for the most recent instructions for configuring BIND to work with an HSM. The following instructions are not meant to be final.

BIND 9 is an open source implements of the Domain Name System (DNS) protocols for the Internet that is suitable for use in high-volume and high-reliability applications. Domain Name System Security Extensions (DNSSEC) extends standard DNS to provide a measure of security. BIND supports the full set of DNSSEC standards. [xi]

Because BIND is complete software package with extensive documentation, this document will only cover the specific steps for configuring BIND with the CrypTech Alpha and does not provide information on configuring BIND. More information on BIND and DNSSEC can be found at https://ftp.isc.org/isc/dnssec-guide/html/dnssec-guide.html which is the official BIND documentation. Please see that documentation for more details and for troubleshooting. BIND is distributed in binary form on Windows, but for other operating systems such as Linux, it needs to be built from source. The following guide was created using BIND 9.12.1-P2 and OpenSSL 1.0.2h.

OpenSSL-based PKCS#11
The CrypTech Alpha HSM doesn't not support Native PKCS#11 because it doesn't support every cryptographic operation that BIND 9 may need. As mentioned in the BIN documentation, OpenSSL-based PKCS#11 uses a modified version of the OpenSSL library and BIND must be built using that library.

Updating the Paths
The following examples will use the following paths:

- Patched Openssl → /opt/pkcs11/usr
- BIND → /opt/bind9
- BIND sysconfdir → /etc/bind

Because the standard install paths for OpenSSL and the BIND tools are not being used, it's very important to set the PATH environment variable. The secure path should also be set using 'sudo visudo'. The 'LD_LIBRARY_PATH' environment variable is the last path to be set as well. It is recommended that these variables be set before making OpenSSL and BIND from source. '/opt/bind9' should be substituted with the path used with --prefix when configuring the BIND build. To make the change permanent, you will also need to add it to either you ~/.profile or ~/.bashrc file.

```
$ export PATH=/opt/bind9/sbin:/opt/bind9/bin:/opt/pkcs11/usr/bin:/opt/pkcs11/usr/lib${PATH}
$ export LD_LIBRARY_PATH=/opt/pkcs11/usr/lib:${LD_LIBRARY_PATH}
```

Troubleshooting: Regardless of the error given by BIND either during the signing process or while making the project, most errors are caused by the environment variables not being set or by the Diamond-HSM software not being installed.

8.1. Build BIND to work with the CrypTech Alpha

8.1.1. Download Bind

Even though we will install OpenSSL first, downloading BIND is necessary because it contains an important patch.

```
$ wget http://ftp.isc.org/isc/bind9/9.12.1-P2/bind-9.12.1-P2.tar.gz
$ tar zxf bind-9.12.1-P2.tar.gz
```

8.1.2. Download OpenSSL Source

This manual has been tested using OpenSSL 1.0.2h. That version of OpenSSL has the required patch needed to build OpenSSL for BIND. If you want to use another version, verify that that patch exists in the bind-9/bin/pkcs11 directory.

```
$ wget http://www.openssl.org/source/openssl-1.0.2h.tar.gz
$ tar zxf openssl-1.0.2h.tar.gz
```

8.1.3. Patch OpenSSL

These directions use OpenSSL 1.0.2h and BIND 9.12.1-P2 for its examples. If other versions are used, that versions in the following examples should change.

```
$patch -p1 -d openssl-1.0.2h \
   < bind-9.12.1-P2/bin/pkcs11/openssl-1.0.2h-patch
```

Details about the how OpenSSL is patched to provide PKCS#11 support and the different flavors of support provided has been documented in the BIND 9 Administrator Reference Manual.

> OpenSSL-based PKCS#11 mode uses a modified version of the OpenSSL library; stock OpenSSL does not fully support PKCS#11. ISC provides a patch to OpenSSL to correct this. This patch is based on work originally done by the OpenSolaris project; it has been modified by ISC to provide new features such as PIN management and key-by-reference.
>
> There are two "flavors" of PKCS#11 support provided by the patched OpenSSL, one of which must be chosen at configuration time. The correct choice depends on the HSM hardware:
>
> - Use ’crypto-accelerator’ with HSMs that have hardware cryptographic acceleration features, such as the SCA 6000 board. This causes OpenSSL to run all supported cryptographic operations in the HSM.
>
> - Use ’sign-only’ with HSMs that are designed to function primarily as secure key storage devices, but lack hardware acceleration. These devices are highly secure but are not necessarily any faster at cryptography than the system CPU --- often, they are slower. It is therefore most efficient to use them only for those

> cryptographic functions that require access to the secured private key, such as zone signing, and to use the system CPU for all other computationally-intensive operations. The AEP Keyper is an example of such a device.
>
> The modified OpenSSL code is included in the BIND 9 release, in the form of a context diff against the latest versions of OpenSSL. OpenSSL 0.9.8, 1.0.0, 1.0.1 and 1.0.2 are supported; there are separate diffs for each version. In the examples to follow, we use OpenSSL 0.9.8, but the same methods work with OpenSSL 1.0.0 through 1.0.2.
>
> - BIND 9 Administrator Reference Manual

8.1.4. Build OpenSSL

Per the BIND 9 Administrator Reference Manual, when building OpenSSL, place it in a non-standard location so that it does not interfere with other OpenSSL libraries on the system. This manual will use the location in the BIND 9 reference material, "/opt/pkcs11/usr" and will be used when configuring BIND. After configuring the OpenSSL build, you will need to run 'make depend'. Pay careful attention to the output of 'make' and 'make test' to make sure OpenSSL has been built correctly.

```
$ cd openssl-1.0.2h
$ ./Configure -fPIC linux-x86_64 \
  --pk11-libname=/usr/lib/libcryptech-pkcs11.so \
  --pk11-flavor=sign-only \
  --prefix=/opt/pkcs11/usr
$ make depend
$ make
$ make test
$ sudo make install
```

8.1.5. Build BIND 9

Once OpenSSL has been made and installed, BIND 9 must be installed with the following configuration.

```
$ cd bind-9.12.1-P2
$ ./configure --enable-threads \
  --with-openssl=/opt/pkcs11/usr \
  --with-pkcs11=/usr/lib/libcryptech-pkcs11.so \
  --sysconfdir=/etc/bind \
  --prefix=/opt/bind9
$ make
$ make test
$ sudo make install
```

The sysconfdir is set to '/etc/bind' in this example because it's used later in this document; however, '/etc/bind' does not need to be used.

PKCS#11 Tools

> BIND 9 includes a minimal set of tools to operate the HSM, including pkcs11-keygen to generate a new key pair within the HSM, pkcs11-list to list objects currently available, pkcs11-destroy to remove objects, and pkcs11-tokens to list available tokens.
>
> In UNIX/Linux builds, these tools are built only if BIND 9 is configured with the --with-pkcs11 option. (Note: If --with-pkcs11 is set to "yes", rather than to the path of the PKCS#11 provider, then the tools will be built but the provider will be left undefined. Use the -m option or the PKCS11_PROVIDER environment variable to specify the path to the provider.)
>
> - BIND 9 Administrator Reference Manual

The PATH environment variable will have to be updated if you change the default location using --prefix. See **Updating the Paths** section above.

8.2. Complete Zone Signing Example

Once BIND has been successfully built, you can use the PKCS#11 tools to generate keys and sign zones. This example will go through the signing of a specific zone file and show the expected output. The following zone file will be used for an example domain, 'example.net'. All commands use 'sudo' so the BIND pkcs#11 tools can use the file '/usr/lib/libcryptech-pkcs11.so'.

The following three steps will be considered:

1) Create the KSK (key signing key) and ZSK (zone signing key) key pairs
2) Add the public key to the zone file
3) Sign the zone

For more information on other BIND topics such as creating a DS-set and replication, please see the BIND documentation.

The following zone file will be used in this example. The filename is 'example.com'.

```
$ORIGIN example.net.
$TTL 86400
@               IN SOA    dns1.example.net.   hostmaster.example.net. (
                          2001062501 ; serial
                          21600      ; refresh after 6 hours
                          3600       ; retry after 1 hour
                          604800     ; expire after 1 week
                          86400 )    ; minimum TTL of 1 day


                IN NS     dns1.example.net.


                IN MX     10     mail.example.net.


dns1            IN A      127.0.0.1
```

```
server1          IN A      127.0.0.1

mail             IN CNAME  server1

www              IN CNAME  server1
```

8.2.1. Creating the Key-Signing Key and Zone-Signing Key

Before proceeding, please make sure the the CrypTech Alpha is connected to the computer and that cryptech_muxd is running without any errors. To create the keys, run the following commands in the terminal. In the folder of your zone.

```
$ sudo pkcs11-keygen -b 1024 -l example-net-ksk
$ sudo pkcs11-keygen -b 2048 -l example-net-zsk
```

Depending on the size of the key, it may take a few minustes to generate. Each command should return 'Key pair generation complete.' The private key will remain on the HSM, but you'll have to add the public keys to the zone file.

To confirm that the key exists:

```
$ sudo pkcs11-list
Enter PIN:
object[0]: handle 2147483658 class 3 label[8] 'example-net-ksk' id[0]
object[1]: handle 2147483657 class 2 label[8] 'example-net-ksk' id[0]
```

dnssec-keyfromlabel generates a key pair of files that referencing a key object stored in a cryptographic hardware service module (HSM). The private key file can be used for DNSSEC signing of zone data as if it were a conventional signing key created by dnssec-keygen, but the key material is stored within the HSM, and the actual signing takes place there. [xii] The algorithm must be set using the '-a' flag and the engine must also be set using '-E pkcs11'.

```
$ sudo dnssec-keyfromlabel -a RSAMD5 -E pkcs11 -l example-net-zsk example.net
Enter PIN:
Kexample.net+001+37518
$ sudo dnssec-keyfromlabel -a RSAMD5 -E pkcs11 -l example-net-ksk -f KSK example.net
Enter PIN:
Kexample.net+001+57242
```

It's important to save the values returned from 'dnssec-keyfromlabel' because it tell the name of the key file for the particular key pair. 'dnssec-keyfromlabel' will create a set of two file. One will have

the extension .key and the other .private. The .key file should be included in the zone file as it the public key. The .private file is not the actual private key because the private key is on the HSM, but it gives information on how to sign using the HSM.

8.2.2. Adding the Public Key to the Zone File

The public key needs to be added to the zone file. This can be done using BIND's '$include' statement as shown below.

```
$ORIGIN example.net.
$TTL 86400
@               IN SOA    dns1.example.net.   hostmaster.example.net. (
                          2001062501 ; serial
                          21600      ; refresh after 6 hours
                          3600       ; retry after 1 hour
                          604800     ; expire after 1 week
                          86400 )    ; minimum TTL of 1 day


                IN NS     dns1.example.net.


                IN MX     10     mail.example.net.


dns1            IN A      127.0.0.1


server1         IN A      127.0.0.1


mail            IN CNAME  server1


www             IN CNAME  server1

$include /etc/bind/zones/example.net.test/Kexample.+001+37518.key
$include /etc/bind/zones/example.net.test/Kexample.+001+57242.key
```

8.2.3. Signing the Zone

'dnssec-signzone' is used to sign the zone. The '-S' option is for smart signing and will create the appropriate .signed file.

```
$sudo dnssec-signzone -S -E pkcs11 example.net
Enter PIN:
Verifying the zone using the following algorithms: RSAMD5.
Zone fully signed:
Algorithm: RSAMD5: KSKs 1 active, 0 stand-by, 0 revoked
                   ZSKs: 1 active, 0 stand-by, 0 revoked
example.net.signed
```

This is the resulting zone file.

example.net.signed:

```
; File written on Wed Jun 20 07:01:50 2018
; dnssec_signzone version 9.12.1-P2
example.net.        86400  IN SOA dns1.example.net. hostmaster.example.net. (
                                    2001062501 ; serial
                                    21600      ; refresh (6 hours)
                                    3600       ; retry (1 hour)
                                    604800     ; expire (1 week)
                                    86400      ; minimum (1 day)
                                    )
                            86400  RRSIG  SOA 1 2 86400 (
                                    20180720110145 20180620110145 37518 example.net.
                                    MU3U0aMVBuz1YzaZfbBhIlmqr2/yBvJEUqqC
                                    Ha1uGV0c2ghcH39NYfxxiQpbhdnNBgZrMY2D
                                    0fr1igsET59TvQ5qopxbTsWshnsmHE0+JiG7
                                    zT1aUei5Iqste+lA/6y7YBoeoqRQ8qum7yY3
                                    DkZyv8DdqyqU4ZPL6IqzHWvdSp4= )
                            86400  NS     dns1.example.net.
                            86400  RRSIG  NS 1 2 86400 (
                                    20180720110145 20180620110145 37518 example.net.
                                    mB2DNklqKxCAD1ysXkyXs8iqEOPRu1yADQ5t
                                    b6x8nPkHMBR0cHZcz64WvxuGBomUksN40uOl
                                    rYWw7uGQdQOu+fBa/qO6FDLMKGte4UHw58Ic
                                    M4/kjyOii4L+D2UCZSwF5UtJrrW2l+XUfyvv
                                    fVjpyWHfnrCHdJc0YPT7htP739I= )
                            86400  MX     10 mail.example.net.
                            86400  RRSIG  MX 1 2 86400 (
                                    20180720110145 20180620110145 37518 example.net.
                                    G/1eQldAcctWoPgiPMvvNFIFjmjWYMTsUIcC
                                    dR41qsEn7CTHchu0FmWt7x2gOcLW+1xSkQ6K
                                    gTLQQf5EQGT7+bRrHkKip/cqWU9zL+6mF+jM
                                    yLAwMhWuT7zl3ug779X6EY8GRignqSELyCa4
                                    e3Rzp6aQwolQ+7SXHwDVx7vEUpY= )
                            86400  NSEC   dns1.example.net. NS SOA MX RRSIG NSEC DNSKEY
                            86400  RRSIG  NSEC 1 2 86400 (
                                    20180720110145 20180620110145 37518 example.net.
                                    G0EpXW59+VM7hq9Q1abcSsVAHEs6EIJ6H/r5
                                    AIKbdXHXdmWV2u1SjqQ8UeCks9RXk0LWeeUq
                                    v5RZAFpJpVmXpislOduAk5uzj3W07oF0Xnmp
                                    XIykg9VAcp/TBek2Dtol3j6CL6Yl9SYKL0VM
                                    AV1lnXwxUM5fG50gqxFDnFj+Aes= )
                            86400  DNSKEY 256 3 1 (
                                    AwEAAeKkz7moqoWX8dx3K5iOr5YeQnrIOLfE
                                    4OUtep23G+NmMcOawDELBMphBlCAcPjtkAvF
                                    QOdty6K+OsCeRh/+0PKX+cTisbV/HdjmJ58/
                                    gSFckeepiueoomkzewdJoNPGW7O//3rn8dqz
                                    1QJett+QHQHh0qH6hSe1FT4mVOkfko6h
                                    ) ; ZSK; alg = RSAMD5 ; key id = 37518
                            86400  DNSKEY 257 3 1 (
                                    AwEAAcJ03xso21S5H4AxCd4AEAptff2EcL2e
                                    G4bYgMINAVYSla28Y6MbUDFg5wUXGHAbc5cz
                                    Tjml4SiReALoiKLQ8Httiq1Iwxcj4fzaYhV3
                                    4BIy9fqGGalY2TFHtmcxCXSMCFPO7nihkzIn
                                    O4koi9fGGkSeUn8LhT5n8TgbD0FssBLud1X1
                                    hddy4Fky6LMZWsri5veF150ZdgszDFqcB2UG
                                    BH1fyJzETKlR7DgHgbsgh3B8L8Ya7q98iydk
                                    orO6nJe4HdDsS9egDdfra9pW71skDUcyye4K
                                    WGQ8/x7wUHsJ3joFrI7CzFDn1q+Nq4Z4Z+Wt
                                    sB00noQaiopMv7u57gHfmt8=
                                    ) ; KSK; alg = RSAMD5 ; key id = 57242
                            86400  RRSIG  DNSKEY 1 2 86400 (
                                    20180720110145 20180620110145 37518 example.net.
```

```
                                gyUFDNjJERDKcxOmzJpNJZp1jrcsf+q2wAeM
                                KMEMFx0GKUy3jTk97D5JMbjAbfQMC/uFu6km
                                3uXS9XPec0tjBHBR7TOmd2nTi67ggZz2OYBW
                                pGxX723t9SRF5XnAB07C4ucQZa6goSs74q3Y
                                ioFPl+sjTkVXmtGh7Bc35LlR9zk= )
                        86400   RRSIG   DNSKEY 1 2 86400 (
                                20180720110145 20180620110145 57242 example.net.
                                KdeLT56n/CxjZrcfBY4l2982wRX4LHpssvMg
                                t2FOF/rVv2v3/KfZuBTKqW6kwnut5X9ioLD4
                                qBH4MVBu5Q0Tzrmiggwo+0xR9oDIyDakK6JJ
                                cfpVvgKcMz2rb2Y55Xrp0pM7FCqdrTS003i8
                                3RJGFzLfkec59l6lMOz16m3ZoJGxG8NCdqhv
                                c7/+VNhdkcMkEKFdRzPG+UJD8cboraZjrEV4
                                /tzgYIA5Z9ipimCcqvCLac3AAkS4TA/R5hVn
                                NDyUnSOLtH/uxC/RWI6m2Se8JOBbQh2ti1Zh
                                /bphBmUBumfuCshq60x7bN0smEcaS1Ph+I43
                                eWjX5/Wt0VHkTVrefw== )
dns1.example.net.86400  IN A    127.0.0.1
                        86400   RRSIG   A 1 3 86400 (
                                20180720110145 20180620110145 37518 example.net.
                                gYJl92B+fOMZlP8r1pEObbF738KH/MkiMfyg
                                p85ruUtT1H5TXPZJFaTBoL881v25K7eunyOR
                                qAriFE7NAWzhV7ubuWZB8F1E1dMaVL9TWKJF
                                wXMsCPIH6jHNdsodlUyV+CGeJgxOxKc77KUf
                                rgTPwh2uMF3wbxfU1HXP+0aVaHU= )
                        86400   NSEC    mail.example.net. A RRSIG NSEC
                        86400   RRSIG   NSEC 1 3 86400 (
                                20180720110145 20180620110145 37518 example.net.
                                ulLpUUkbDC27qMcgTlvc0rIm7SZR22RL1G+S
                                R8VZldkeFF8uJ3PYnfKe+ExEsCMB3F5tMNhT
                                3YRJf1v2ES6D+YPT7yEV6xBxFg08oIwqUDVd
                                5gnhwobH/4aGkMlZTJyWyqlAgIMQEHyR1+z6
                                iRah6RE/1kmE4S0M9gIksLbcgFg= )
www.example.net. 86400  IN CNAME server1.example.net.
                        86400   RRSIG   CNAME 1 3 86400 (
                                20180720110145 20180620110145 37518 example.net.
                                ekjwXqc/+VyDqGWiF/Y8ThDKoZDNNvQGbLdu
                                wzBbJVX8juCvCJmi+qK7p15Dj4lC2fmSeyML
                                zaMz6rMh5Ynl1jozLw06yfaEpT/41P3VohVK
                                bJpMdy1IqP4bbyw4j/GS8kSj3Kk6jGfBxZ+6
                                2oWH7mo33TO2cA8uuIkTT7zx87s= )
                        86400   NSEC    example.net. CNAME RRSIG NSEC
                        86400   RRSIG   NSEC 1 3 86400 (
                                20180720110145 20180620110145 37518 example.net.
                                dq8O5chwXvgsI055AgPNSQaGbjDO5XmnENpP
                                ZTC3B4lwhVf09noM+k3l4HJRew//xPk2RSE1
                                Wf76gyq+CGKqrjci2jRmt+4EedTrn6nrnkxs
                                EHrqd//4ZV5wz0AoDKwudPzHwb0yrBqPfH12
                                f6C87sxqE/OI7yrihqQtl87XfTQ= )
mail.example.net.86400  IN CNAME server1.example.net.
                        86400   RRSIG   CNAME 1 3 86400 (
                                20180720110145 20180620110145 37518 example.net.
                                c836bzTL5DrvUxNjwRT2/bPScv7nLMtVZcAk
                                w3XaFUJlwjGkRvcImyIVAr0d/+/VnjtKlfE3
                                /miwcYJgoteCrTZx5xSFFSetSDob/sZqCwvm
                                yeCAwCOSoFEiToms1Prb+NKpkGuNySTg3TrW
                                BL1n+HCJ2A8VqoaciYuk4cwNlpM= )
                        86400   NSEC    server1.example.net. CNAME RRSIG NSEC
                        86400   RRSIG   NSEC 1 3 86400 (
                                20180720110145 20180620110145 37518 example.net.
                                JH5q4sxWFlDTFixo2gcAB78eR7O3TkTtwXAU
                                4278jQlasCGy2g2PcmGYhpAhP0JWfBEd/22M
                                vCZLCkIxDSZnBerntCUIYpxYs5tOWdYwo55p
                                G34xPAZI1JVjmZ2ctwO7zgq4jqDU+pQmv7EW
```

```
                                        v2jQG2fBIZyNaRnvdNjBtAqLLhg= )
server1.example.net.        86400  IN A    127.0.0.1
                            86400  RRSIG   A 1 3 86400 (
                                           20180720110145 20180620110145 37518 example.net.
                                           bTexyOvXy8igKm2HP9GAALvfZ1lEk0wYINp4
                                           9nqogyUU74dZchOOmkos0ZOWvvqr1kZ3PR8X
                                           WD/JzavV7N7Q1iw7Um2rZrCFCKepXJ9Gfban
                                           dQe1rP5N648Pb2Q+zmdx5hE4FrLGYUOOZH9/
                                           H1p36vxSq1pP2XPRE3MueopKcNY= )
                            86400  NSEC    www.example.net. A RRSIG NSEC
                            86400  RRSIG   NSEC 1 3 86400 (
                                           20180720110145 20180620110145 37518 example.net.
                                           KcWhgubez09s298vN7WfOoHiY18ROxsBZCB2
                                           HUn3z0KHWB5W2vjZxInH8QcrBF+KctJemOlP
                                           xSzPkO+lvbGgnRDSH+RWsvXfSVz5veHSmrQn
                                           tqd9lBjpjwfMykiRNq2jA2gYf2Bl5HbddPPh
                                           EfVgFglQDHvKA2PW4qJ3JSmD/r8= )
```

8.2.4. Running named with automatic zone re-signing

If you want named to dynamically re-sign zones using HSM keys, and/or to sign new records inserted via nsupdate, then named must have access to the HSM PIN. In OpenSSL based PKCS#11, this is accomplished by placing the PIN into the openssl.cnf file (in the above examples, `/opt/pkcs11/usr/ssl/openssl.cnf`).

The location of the openssl.cnf file can be overridden by setting the `OPENSSL_CONF` environment variable before running named.

Sample `openssl.cnf`:

```
openssl_conf = openssl_def

[ openssl_def ]

engines = engine_section

[ engine_section ]

pkcs11 = pkcs11_section

[ pkcs11_section ]

PIN = <PLACE PIN HERE>
```

This will also allow the `dnssec-*` tools to access the HSM without PIN entry. (The `pkcs11-*` tools access the HSM directly, not via OpenSSL, so a PIN will still be required to use them.)

Placing the HSM's PIN in a text file in this manner may reduce the security advantage of using an HSM. Be sure this is what you want to do before configuring the system in this way.

A. References and Endnotes

[i] https://www.isc.org/downloads/bind/
[ii] https://trac.cryptech.is/
[iii] https://en.wikipedia.org/wiki/Domain_Name_System
[iv] https://en.wikipedia.org/wiki/Domain_Name_System_Security_Extensions
[v] https://git-scm.com
[vi] https://en.wikipedia.org/wiki/Hardware_security_module
[vii] https://www.opendnssec.org/
[viii] https://github.com/OpenSC/OpenSC/wiki
[ix] https://en.wikipedia.org/wiki/Python_(programming_language)
[x] https://trac.cryptech.is/wiki/AlphaBoardComponents
[xi] https://ftp.isc.org/isc/dnssec-guide/html/dnssec-guide.html#hardware-security-modules
[xii] https://ftp.isc.org/isc/bind9/cur/9.10/doc/arm/man.dnssec-keyfromlabel.html