

ISC

Bind 9

9.20.23

Integration Guide

u.trust GP HSM

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	2026-07-13
Status	PUBLISHED
Document No.	IG-2026-0057
All rights reserved	No part of this documentation may be reproduced in any form (printing, photocopy or according to any other process) without the written approval of Utimaco IS GmbH or be processed, reproduced or distributed using electronic systems. Utimaco IS GmbH reserves the right to modify or amend the documentation at any time without prior notice. Utimaco IS GmbH assumes no liability for typographical errors and damages incurred due to them. All trademarks and registered trademarks are the property of their respective owners.

Table of Contents

1	About This Guide	5
1.1	Target Audience	5
1.2	Document Conventions	5
1.3	Abbreviations	6
2	Product Overview	8
2.1	Bind9	8
2.2	Utimaco SecurityServer HSM	8
3	Integration Requirements and Prerequisites	9
3.1	Tested Versions	9
3.2	Software Requirements	9
3.3	Hardware Requirements	10
3.4	Prerequisites	10
4	Installing and Configuring Utimaco SecurityServer Software	11
4.1	Download and Install Utimaco Software	11
4.2	SecurityServer PKCS#11 Configuration	12
4.3	Create SO User and Initialize a Slot	13
5	Configuring OpenSSL to Use Utimaco SecurityServer	14
5.1	Installing OpenSSL and pkcs11-provider	14
5.2	Setting up Utimaco SecurityServer Library in OpenSSL Configuration File	15
6	Installing and Running Bind9	17
7	Integrating Bind9 with Utimaco HSM	22
7.1	Generate ZSK and KSK on Utimaco HSM	22
7.2	Generate Reference Key files of HSM Keys for Bind9	26
7.2.1	For RSA Keys	27
7.2.2	For ECC Keys	28
7.3	Sign and Verify the Zone file	31
7.3.1	Sign and Verify Zone Files with RSA Keys	31
7.3.2	Sign and Verify Zone Files with ECC Keys	32
7.4	Configuring Bind9 to Use Signed Zone File	34
7.4.1	Using RSA Reference File	34
7.4.2	Using ECC Reference File	36

7.5 Key Rollover for KSK and ZSK 38

7.5.1 Generate new KSK and ZSK..... 38

7.5.2 KSK Rollover..... 40

7.5.3 ZSK Rollover 43

8 Troubleshooting48

9 Further Information49

10 Contact and Support Information.....50

11 Appendix A: References51

12 Appendix B: Command Summary52

1 About This Guide

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/>.

This guide provides an integration guide explaining how to integrate Utimaco SecurityServer Hardware Security Module (HSM) with Bind9. Utimaco HSM securely generates and stores the KSK and ZSK keys required by Bind9 to sign the zone files.

1.1 Target Audience

This guide is intended for administrators of Bind9 and of Utimaco HSMs.

1.2 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
<code>Monospaced</code>	Code that is given for explanation or as an example, file paths	<code>certreq.exe -new request.inf IISCertRequest.csr</code>
<i>Italic</i>	References and important terms	Operating system listed in <i>Tested Versions</i>

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.

1.3 Abbreviations

Abbreviation	Meaning
BIND	Berkeley Internet Name Domain
CSADM	CryptoServer Command-line Administration Tool
DNS	Domain Name System
DNSSEC	Domain Name System Security Extensions
DS	Delegation Signer
ECC	Elliptic Curve Cryptography
GUI	Graphical User Interface
HSM	Hardware Security Module
IP	Internet Protocol

Abbreviation	Meaning
KSK	Key-signing Key
LAN	Local Area Network
PCI-e	PCI Express Interface
PIN	Personal Identification number
PKCS#11	Public-Key Cryptography Standard #11
RHEL	Red Hat Enterprise Linux
RSA	Rivest-Shamir-Adleman
SO	Security Officer
URL	Uniform Resource Locator
ZSK	Zone-signing Key
URI	Uniform Resource Identifier
TTL	Time to Live
PKI	Public Key Infrastructure

Table 2: List of abbreviations

2 Product Overview

2.1 Bind9

BIND9 is an open-source DNS server software used for secure DNS services. It supports DNSSEC, dynamic updates, zone transfers, and load. The Berkeley Internet Name Domain (BIND) software implements a domain name server for several operating systems. BIND 9 can be configured as an authoritative name server, a resolver, and, on supported hosts, a stub resolver. While large operators usually dedicate DNS servers to a single function per system, smaller operators will find that BIND 9's flexible configuration features support multiple functions, such as a single DNS server acting as both an authoritative name server and a resolver.

2.2 Utimaco SecurityServer HSM

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

3 Integration Requirements and Prerequisites

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

3.1 Tested Versions

The integrations that have been successfully tested with the Utimaco HSM and Bind9.

Operating System	Bind9 Version	Utimaco Security Server Version	Utimaco HSM
RHEL 8	9.20.23	SecurityServer 6.5.0	GP HSM CSeSeries/ Se-Series

Table 3: List of tested versions

3.2 Software Requirements

Software	Software Requirements
OpenSSL	3.5.1
pkcs11 provider	pkcs11 provider 1.2.0
HSM Utility	PKCS#11 Tool Version 2 (p11tool2)
HSM Interfaces	SecurityServer PKCS#11 Provider

Table 4: List of software requirements

3.3 Hardware Requirements

Hardware	Hardware Requirements
Utimaco LAN HSM	u.trust GP HSM CSe-Series/Se-Series LAN with firmware SecurityServer 6.5.0 or higher
Utimaco PCI-e HSM	u.trust GP HSM CSe-Series/Se-Series PCI-e with firmware SecurityServer 6.5.0 or higher

Table 5: List of hardware requirements



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

3.4 Prerequisites

Before you begin, please ensure that:

- The operating system used is listed in *Tested Versions*.
- The SecurityServer used is listed in *Tested Versions*.
- The SecurityServer Default Admin has been replaced with a new admin user.
- The SecurityServer is set up and configured. Refer to the SecurityServer documentation to set up the HSM.
- You have admin access as it is required to install the software.
- You familiarize yourself with the Bind9 documents and setup process.
- Port 53 is allowed through the firewall.

4 Installing and Configuring Utimaco SecurityServer Software

4.1 Download and Install Utimaco Software

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

1. Copy the downloaded software at the appropriate location on the Bind9 Server.
2. Create utimaco folder under `/opt` directory and further create 2 directories: `/opt/utimaco/bin` and `/opt/utimaco/lib`.

›_ Console

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

3. Copy pkcs11 library file `libcs_pkcs11_R3.so` from Utimaco SecurityServer software to the `/opt/utimaco/lib` directory.

›_ Console

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

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

›_ Console

```
# cd ~/path_to_application_folder
# cp csadm p11tool2 /opt/utimaco/bin
# chmod +x /opt/utimaco/bin/csadm /opt/utimaco/bin/p11tool2
```

4.2 SecurityServer PKCS#11 Configuration

1. Create the directory `/home/admin/Bind9`. Locate the Utimaco PKCS#11 configuration file in your SecurityServer directory, `Linux/x86-64/Crypto_APIS/PKCS11_R3/sample`. Copy the Utimaco PKCS#11 configuration file `cs_pkcs11_R3.cfg` into the `/home/admin/Bind9/` directory.

›_ Console

```
# cd <install directory>/Software/Linux/x86-64/Crypto_APIS/PKCS11_R3/sample
# cp cs_pkcs11_R3.cfg /home/admin/Bind9
# cd /home/admin/Bind9
```

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

cs_pkcs11_R3.cfg

```
[Global]
# For unix:
Logpath = /tmp
# Loglevel (0 = NONE; 1 = ERROR; 2 = WARNING; 3 = INFO; 4 = TRACE)
Logging = 1
Keepalive = false
# Set the Device to connect with
[CryptoServer]
# Device specifier
Device = <HSM_IP>
```

3. Set pkcs11 environment variable `'CS_PKCS11_R3_CFG'`.

```
export CS_PKCS11_R3_CFG=/home/admin/Bind9/cs_pkcs11_R3.cfg
```




For more information regarding the commands and command parameters please, check the Utimaco SecurityServer 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 Create SO User and Initialize a Slot

You must initialize a slot with a custom label using p11tool2.

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

> _ Console

```
# p11tool2 slot=<slot_no> Label=<token_label> Login=ADMIN,ADMIN.key  
InitToken=ask  
# p11tool2 slot=<slot_no> LoginSO=ask InitPin=ask
```

5 Configuring OpenSSL to Use Utimaco SecurityServer

Bind9 uses OpenSSL together with the PKCS#11 provider to perform DNSSEC zone-signing operations using keys stored in the Utimaco SecurityServer HSM. Follow the steps below to configure the PKCS#11 provider with OpenSSL.

5.1 Installing OpenSSL and pkcs11-provider

1. Download and install OpenSSL 3.5.1.
2. Verify the version of OpenSSL.

```
openssl version -a
```

```
[admin@localhost ~]$  
[admin@localhost ~]$  
[admin@localhost ~]$ openssl version  
OpenSSL 3.5.1 1 Jul 2025 (Library: OpenSSL 3.5.1 1 Jul 2025)  
[admin@localhost ~]$  
[admin@localhost ~]$  
[admin@localhost ~]$
```

Figure 1 : OpenSSL version details

3. Download and install pkcs11-provider. Installation steps are available here <https://integration-guides.utimaco.com/openssl-3.5-and-u.trust-gp-hsm-se-series/1.0.0/installing-pkcs11-provider>.
4. Verify the pkcs11-provider.

```
openssl list -providers
```

```
[admin@localhost ~]$  
[admin@localhost ~]$ openssl list -providers  
Providers:  
  default  
    name: OpenSSL Default Provider  
    version: 3.5.1  
    status: active  
  pkcs11  
    name: PKCS#11 Provider  
    version: 1.2.0  
    status: active  
[admin@localhost ~]$  
[admin@localhost ~]$
```

Figure 2 : Pkcs11-provider version details

5.2 Setting up Utimaco SecurityServer Library in OpenSSL Configuration File

1. Open the file `/etc/pki/tls/openssl.cnf` and enter the following line in the first line of the file.

openssl.cnf

```
openssl_conf = openssl_init
```

2. The `openssl.cnf` file changes highlighted in the below screenshot, update those in the existing `openssl.cnf`.

openssl.cnf

```
# List of providers to load
[provider_sect]
default = default_sect
pkcs11 = pkcs11_sect

[default_sect]
activate = 1

[pkcs11_sect]
module = /usr/lib64/openssl-modules/pkcs11.so
pkcs11-module-path = /opt/utimaco/lib/libcs_pkcs11_R3.so
pkcs11-module-login-behavior = always
activate = 1
```

```
[admin@localhost ~]$  
[admin@localhost ~]$ diff -Nurb /etc/pki/tls/org_openssl.cnf /etc/pki/tls/openssl.cnf  
--- /etc/pki/tls/org_openssl.cnf      2026-06-08 00:28:01.388983291 -0700  
+++ /etc/pki/tls/openssl.cnf         2026-06-17 06:51:06.262345988 -0700  
@@ -64,11 +64,18 @@  
  # List of providers to load  
  [provider_sect]  
  default = default_sect  
+pkcs11 = pkcs11_sect  
  ##legacy = legacy_sect  
  ##  
  [default_sect]  
  activate = 1  
  
+[pkcs11_sect]  
+module = /usr/lib64/openssl-modules/pkcs11.so  
+pkcs11-module-path = /opt/utimaco/lib/libcs_pkcs11_R3.so  
+pkcs11-module-login-behavior = always  
+activate = 1  
+  
  ##[legacy_sect]  
  ##activate = 1  
  
[admin@localhost ~]$
```

Figure 3 : Diff of the openssl.cnf file



Dynamic path and Module path will change according to the user environment.

6 Installing and Running Bind9

1. Download the latest version of source code file from [BIND 9 - ISC](#).
2. Extract the `.tar` file.

›_ Console

```
# tar -xvf bind9-9.20.23.tar.gz
```

3. Go to the extracted folder and run the `autoreconf` command.

›_ Console

```
# autoreconf -fi
```

4. Install the dependent packages for Bind9 .

›_ Console

```
# dnf install openssl-devel perl libicu-devel pkgconf userspace-rcu-devel  
libcap-devel libuv-devel
```

5. Run the `configure` command.

›_ Console

```
# configure CC="gcc -m64" -enable-threads -disable-doh
```

6. Build and install Bind9 as described below:

> _ Console

```
# make
# make install
```



It will install Bind9 on default directory as `/usr/local/`.

7. Move the file `named.root.key` and `named.rfc1912.zones` to `/etc` directory.

8. Check the version of Bind9 as described below.

```
[admin@localhost ~]$
[admin@localhost ~]$ /usr/local/sbin/named -version
BIND 9.20.23 (Stable Release) <id:>
[admin@localhost ~]$
[admin@localhost ~]$
```

Figure 4 : Bind9 version output

9. Create a forward and reverse zone file at the `/var/named` directory.

For example, below are sample files `example.net` and `exampleRev.net` for forward and reverse zones respectively.

```
[admin@localhost ~]$ cat /var/named/example.net
$TTL 1D
$ORIGIN example.net.
@      IN      SOA      example.net. hostmaster.example.net. (
                                0          ; serial
                                1D         ; refresh
                                1H         ; retry
                                1W         ; expire
                                3H         ; minimum
)
      IN      NS       @
      IN      A        172.31.1.75
test   IN      A        172.31.1.75
mail   IN      A        172.31.1.75
neo    IN      A        172.31.1.75
      IN      AAAA     ::1
[admin@localhost ~]$
[admin@localhost ~]$
```

Figure 5 : example.net

```
[admin@localhost ~]$  
[admin@localhost ~]$ cat /var/named/exampleRev.net  
$TTL 86400  
@      IN      SOA      example.net. hostmaster.example.net. (  
                                2009072852 ; Serial  
                                28800      ; Refresh  
                                14400      ; Retry  
                                3600000    ; Expire  
                                86400      ; Minimum  
)  
      IN      NS       example.net.  
75     IN      PTR      example.net.  
[admin@localhost ~]$
```

Figure 6 : exampleRev.net

10. Make the appropriate changes in the configuration file of Bind9 that is `/usr/local/etc/named.conf` related to zone as highlighted. Below is the sample file `named.conf`.

named.conf

```
#cat /usr/local/etc/named.conf

options {
    listen-on port 53 {
        127.0.0.1;
        172.31.1.75;
    };

    listen-on-v6 port 53 {
        ::1;
    };

    directory "/var/named";

    dump-file          "/var/named/data/cache_dump.db";
    statistics-file    "/var/named/data/named_stats.txt";
    memstatistics-file  "/var/named/data/named_mem_stats.txt";
    secroots-file      "/var/named/data/named.secroots";
    recursing-file     "/var/named/data/named.recursing";

    allow-query {
        localhost;
    };

    recursion no;
    dnssec-validation no;

    managed-keys-directory "/var/named/dynamic";

    pid-file "/run/named/named.pid";
    session-keyfile "/run/named/session.key";

    include "/etc/crypto-policies/back-ends/bind.config";
};

logging {
    channel default_debug {
        file "data/named.run";
        severity dynamic;
    };
};

zone "." IN {
    type hint;
    file "named.ca";
};

zone "example.net" {
```


named.conf

```
type primary;
file "example.net";
};
```

11. Start the 'named' service for Bind9 using below command and verify the 'named' process running.

_ Console

```
#/usr/local/sbin/named -f -c /usr/local/etc/named.conf
```

```
[admin@localhost ~]$
[admin@localhost ~]$ /usr/local/sbin/named -f -c /usr/local/etc/named.conf
[admin@localhost ~]$
[admin@localhost ~]$ ps -ef | grep named
root      600038      1   0  04:20 ?                00:00:00 sudo CS_PKCS11_R3_CFG=/home/admin/Bind9/cs_pkcs11_R3.cfg /u
sr/local/sbin/named -f -4 -c /usr/local/etc/named.conf -g
root      600039    600038   0  04:20 ?                00:00:00 /usr/local/sbin/named -f -4 -c /usr/local/etc/named.conf -g
admin     601699    601310   0  08:08 pts/2          00:00:00 grep --color=auto named
[admin@localhost ~]$
[admin@localhost ~]$
```

Figure 7 : Start named service

7 Integrating Bind9 with Utimaco HSM

7.1 Generate ZSK and KSK on Utimaco HSM

To generate a zone-signing key (ZSK) and a key-signing key (KSK) on the HSM use the `pkcs11-tool` utility. After the keys are generated on HSM, use `dnssec-keyfromlabel` provided by BIND to generate the key files for BIND containing a public key and an identifier of the actual private key.

Two slots need to be initialized in HSM for RSA and ECC keys. The slot labels should be unique. For this integration, slot labels are 'Bind9_RSA' and 'Bind9_EC'.

Generating KSK and ZSK with RSA Algorithm

1. Run the following commands to generate a zone-signing key and a key-signing key using RSA algorithm:

_ Console

```
pkcs11-tool --module /opt/utimaco/lib/libcs_pkcs11_R3.so --slot 0 --login --keypairgen --key-type RSA:2048 --label ksk --id 01

pkcs11-tool --module /opt/utimaco/lib/libcs_pkcs11_R3.so --slot 0 --login --keypairgen --key-type RSA:2048 --label zsk --id 02
```

```
[admin@localhost ~]$
[admin@localhost ~]$ pkcs11-tool --module /opt/utimaco/lib/libcs_pkcs11_R3.so --slot 0 --login --keypairgen --key-type RSA:2048 --label ksk --id 01
Logging in to "Bind9_RSA".
Please enter User PIN:
Key pair generated:
Private Key Object; RSA
  label:      ksk
  ID:         01
  Usage:      decrypt, sign, unwrap
  Access:     sensitive, always sensitive, never extractable, local
  Unique ID:  9zR...
  uri:        pkcs11:model=CryptoServer;manufacturer=Utimaco%20IS%20GmbH;serial=SI003001_0000;token=Bind9_RSA;id=01;object=ksk;type=private
Public Key Object; RSA 2048 bits
  label:      ksk
  ID:         01
  Usage:      encrypt, verify, wrap
  Access:     local
  Unique ID:  C...
  uri:        pkcs11:model=CryptoServer;manufacturer=Utimaco%20IS%20GmbH;serial=SI003001_0000;token=Bind9_RSA;id=01;object=ksk;type=public
[admin@localhost ~]$
```

Figure 8 : Key generation output for KSK -RSA

```
[admin@localhost ~]$
[admin@localhost ~]$ pkcs11-tool --module /opt/utimaco/lib/libcs_pkcs11_R3.so --slot 0 --login --keypairgen --key-type RSA:2048 --label zsk --id 02
Logging in to "Bind9_RSA".
Please enter User PIN:
Key pair generated:
Private Key Object; RSA
  label:      zsk
  ID:         02
  Usage:      decrypt, sign, unwrap
  Access:     sensitive, always sensitive, never extractable, local
  Unique ID:  qX[REDACTED]E[REDACTED]@BLH[REDACTED]
  uri:        pkcs11:model=CryptoServer;manufacturer=Utimaco%20IS%20GmbH;serial=SI003001_0000;token=Bind9_RSA;id=%02;object=zsk;type=private
Public Key Object; RSA 2048 bits
  label:      zsk
  ID:         02
  Usage:      encrypt, verify, wrap
  Access:     local
  Unique ID:  [REDACTED]O[REDACTED]ZjF[REDACTED]
  uri:        pkcs11:model=CryptoServer;manufacturer=Utimaco%20IS%20GmbH;serial=SI003001_0000;token=Bind9_RSA;id=%02;object=zsk;type=public
[admin@localhost ~]$
```

Figure 9 : Key generation output for ZSK -RSA



Each key should have a unique label that will be used to refer to the private key in the next steps.

2. List the keys using the `p11tool2` command.

›_ Console

```
p11tool2 slot=0 LoginUser=ask ListObjects
```

```
[admin@localhost Administration]$
[admin@localhost Administration]$ ./p11tool2 slot=0 LoginUser=ask ListObjects
Enter normal user PIN:

CKO_PUBLIC_KEY:

+ 1.1
  CKA_KEY_TYPE           = CKK_RSA
  CKA_UNIQUE_ID           = 47F3D5BF-A118-44DC-8B19-0DAE9EF76E98
  CKA_LABEL               = ksk
  CKA_ID                  = 0x01 ()

+ 1.2
  CKA_KEY_TYPE           = CKK_RSA
  CKA_UNIQUE_ID           = D20F49D3-F4A9-4FB4-8D5A-DAA846CD9E83
  CKA_LABEL               = zsk
  CKA_ID                  = 0x02 ()

CKO_PRIVATE_KEY:

+ 2.1
  CKA_KEY_TYPE           = CKK_RSA
  CKA_UNIQUE_ID           = 7158C7FD-CC7C-451E-9040-420F4C48B2A4
  CKA_SENSITIVE           = CK_TRUE
  CKA_EXTRACTABLE         = CK_FALSE
  CKA_LABEL               = zsk
  CKA_ID                  = 0x02 ()

+ 2.2
  CKA_KEY_TYPE           = CKK_RSA
  CKA_UNIQUE_ID           = 397AD08B-8E00-4D81-9EE2-CA97CC430AD6
  CKA_SENSITIVE           = CK_TRUE
  CKA_EXTRACTABLE         = CK_FALSE
  CKA_LABEL               = ksk
  CKA_ID                  = 0x01 ()
[admin@localhost Administration]$
[admin@localhost Administration]$
```

Figure 10 : List key output in slot 0

Generating KSK and ZSK with ECC algorithm

1. Run the following commands to generate a zone-signing key and a key-signing key using ECC. Curve used is prime256v1.

_ Console

```
pkcs11-tool --module /opt/utimaco/lib/libcs_pkcs11_R3.so --slot 1 --login --
keypairgen --key-type EC:prime256v1 --label ksk --id 01
pkcs11-tool --module /opt/utimaco/lib/libcs_pkcs11_R3.so --slot 1 --login --
keypairgen --key-type EC:prime256v1 --label zsk --id 02
```

```
[admin@localhost ~]$
[admin@localhost ~]$ pkcs11-tool --module /opt/utimaco/lib/libcs_pkcs11_R3.so --slot 1 --login --keypairgen --key-type EC:prime256v1 --label ksk --id 01
Logging in to "Bind9_EC".
Please enter User PIN:
Key pair generated:
Private Key Object; EC
label:      ksk
ID:         01
Usage:      decrypt, sign, unwrap, derive
Access:     sensitive, always sensitive, never extractable, local
Unique ID:  +.4J:8cs-
uri:        pkcs11:model=CryptoServer;manufacturer=Utimaco%20IS%20GmbH;serial=SI003001_0001;token=Bind9_EC;id=01;object=ksk;type=private
Public Key Object; EC EC POINT 256 bits
EC_POINT:   044104e25d245030e8b98256b3b83c9807e7409b823f0985db9fdec9bcbfc8b8cb644a802ff1bbb42c523cdcfa1f97869b7e221d3024e95973c181e22354894822beab
EC_PARAMS:  06082a8648ce3d030107
label:      ksk
ID:         01
Usage:      encrypt, verify, wrap, derive
Access:     local
Unique ID:  3n1KCaF1l{
uri:        pkcs11:model=CryptoServer;manufacturer=Utimaco%20IS%20GmbH;serial=SI003001_0001;token=Bind9_EC;id=01;object=ksk;type=public
[admin@localhost ~]$
```

Figure 11 : Key generation output for KSK -ECC

```
[admin@localhost ~]$
[admin@localhost ~]$ pkcs11-tool --module /opt/utimaco/lib/libcs_pkcs11_R3.so --slot 1 --login --keypairgen --key-type EC:prime256v1 --label zsk --id 02
Logging in to "Bind9_EC".
Please enter User PIN:
Key pair generated:
Private Key Object; EC
label:      zsk
ID:         02
Usage:      decrypt, sign, unwrap, derive
Access:     sensitive, always sensitive, never extractable, local
Unique ID:  L{ R8
uri:        pkcs11:model=CryptoServer;manufacturer=Utimaco%20IS%20GmbH;serial=SI003001_0001;token=Bind9_EC;id=02;object=zsk;type=private
Public Key Object; EC EC POINT 256 bits
EC_POINT:   044104f6ca4d06199bfb1b1524ae6186dae192e8efa87e7776f8be9bb2c52411f0c373edbc9ac33c2116b90ccc405d9b5ab0f591419f4a371e51faecb6eb3f00a9ad948
EC_PARAMS:  06082a8648ce3d030107
label:      zsk
ID:         02
Usage:      encrypt, verify, wrap, derive
Access:     local
Unique ID:  P8AYqZ
uri:        pkcs11:model=CryptoServer;manufacturer=Utimaco%20IS%20GmbH;serial=SI003001_0001;token=Bind9_EC;id=02;object=zsk;type=public
[admin@localhost ~]$
[admin@localhost ~]$
```

Figure 12 : Key generation output for ZSK-ECC



Each key should have a unique label that will be used to refer to the private key in the next steps.

2. List the keys using the `pkcs11-tool` command.

› _ Console

```
# p11tool2 slot=1 LoginUser=ask ListObjects
```

```
[admin@localhost Administration]$
[admin@localhost Administration]$ ./p11tool2 slot=1 LoginUser=ask ListObjects
Enter normal user PIN:

CKO_PUBLIC_KEY:

+ 1.1
  CKA_KEY_TYPE           = CKK_ECDSA
  CKA_UNIQUE_ID           = 33AE6ECB-A54B-4361-8AFB-0146319B6C7B
  CKA_LABEL               = ksk
  CKA_ID                  = 0x01 ()

+ 1.2
  CKA_KEY_TYPE           = CKK_ECDSA
  CKA_UNIQUE_ID           = 8EFE50A1-F5AC-41D2-B0A9-A618F3F5715A
  CKA_LABEL               = zsk
  CKA_ID                  = 0x02 ()

CKO_PRIVATE_KEY:

+ 2.1
  CKA_KEY_TYPE           = CKK_ECDSA
  CKA_UNIQUE_ID           = 2BD02E34-D7F1-4A3A-AC38-9DCDBC732D17
  CKA_SENSITIVE           = CK_TRUE
  CKA_EXTRACTABLE         = CK_FALSE
  CKA_LABEL               = ksk
  CKA_ID                  = 0x01 ()

+ 2.2
  CKA_KEY_TYPE           = CKK_ECDSA
  CKA_UNIQUE_ID           = 4C7B20B6-F052-4808-BD05-0EC885B71F82
  CKA_SENSITIVE           = CK_TRUE
  CKA_EXTRACTABLE         = CK_FALSE
  CKA_LABEL               = zsk
  CKA_ID                  = 0x02 ()
[admin@localhost Administration]$
[admin@localhost Administration]$
```

Figure 13 : List key output in slot 1

7.2 Generate Reference Key files of HSM Keys for Bind9

Convert the RSA keys stored in the HSM into a format that Bind9 understands. The `dnssec-keyfromlabel` tool from Bind9 can link the raw keys stored in the HSM with the

`K<zone>+<alg>+<id>` files. Provide the algorithm and the PKCS#11 label that specify the token, the name of the PKCS#11 object (called label when generating the keys using `pkcs11-tool`) and the HSM slot PIN.

7.2.1 For RSA Keys

1. Create a new directory `RSA_keyfiles/` for keeping RSA reference keys files.
2. Generate the key file for KSK.

›_ Console

```
/usr/local/bin/dnssec-keyfromlabel -f KSK -a RSASHA256 -l  
"pkcs11:token=Bind9_RSA;object=ksk" example.net
```

Where the parameter:

- `-l` specifies the key label in pkcs11 URI format.
- `-f` specifies the key flag.
- `-a` is the algorithm.
- `example.net` is the name of zone file.

```
[admin@localhost RSA_keyfiles]$  
[admin@localhost RSA_keyfiles]$ sudo -E /usr/local/bin/dnssec-keyfromlabel -f KSK -a RSASHA256 -l "pkcs11:token=  
=Bind9_RSA;object=ksk" example.net  
Enter PIN for PKCS#11 Token (Slot 0 - 3001@172.31.1.61 - CLUSTER_0000 - SLOT_0000):  
Kexample.net.+008+44141  
[admin@localhost RSA_keyfiles]$  
[admin@localhost RSA_keyfiles]$  
[admin@localhost RSA_keyfiles]$ ls -lh  
total 8.0K  
-rw-r--r-- 1 root root 605 Jul  1 10:07 Kexample.net.+008+44141.key  
-rw----- 1 root root 554 Jul  1 10:07 Kexample.net.+008+44141.private  
[admin@localhost RSA_keyfiles]$  
[admin@localhost RSA_keyfiles]$
```

Figure 14 : Key file generation for KSK key

3. Generate the key file for ZSK.

›_ Console

```
/usr/local/bin/dnssec-keyfromlabel -a RSASHA256 -l  
"pkcs11:token=Bind9_RSA;object=zsk" example.net
```

```
[admin@localhost RSA_keyfiles]$
[admin@localhost RSA_keyfiles]$
[admin@localhost RSA_keyfiles]$ sudo -E /usr/local/bin/dnsssec-keyfromlabel -a RSASHA256 -l "pkcs11:token=Bind9_
RSA;object=zsk" example.net
Enter PIN for PKCS#11 Token (Slot 0 - 3001@172.31.1.61 - CLUSTER_0000 - SLOT_0000):
Kexample.net.+008+22449
[admin@localhost RSA_keyfiles]$
[admin@localhost RSA_keyfiles]$ ls -lh
total 16K
-rw-r--r-- 1 root root 606 Jul  1 10:18 Kexample.net.+008+22449.key
-rw----- 1 root root 554 Jul  1 10:18 Kexample.net.+008+22449.private
-rw-r--r-- 1 root root 605 Jul  1 10:07 Kexample.net.+008+44141.key
-rw----- 1 root root 554 Jul  1 10:07 Kexample.net.+008+44141.private
[admin@localhost RSA_keyfiles]$
[admin@localhost RSA_keyfiles]$
```

Figure 15 : Key file generation for the ZSK Key

Where the parameter:

- -l specifies the key label in pkcs11 URI format.
- -a is the algorithm.
- example.net is the name of zone file.

4. Verify that you have two KSK and two ZSK key files available.

› Console

```
# ls -lh
```

```
[admin@localhost RSA_keyfiles]$ ls -lh
total 16K
-rw-r--r-- 1 root root 606 Jul  1 10:18 Kexample.net.+008+22449.key
-rw----- 1 root root 554 Jul  1 10:18 Kexample.net.+008+22449.private
-rw-r--r-- 1 root root 605 Jul  1 10:07 Kexample.net.+008+44141.key
-rw----- 1 root root 554 Jul  1 10:07 Kexample.net.+008+44141.private
[admin@localhost RSA_keyfiles]$
```

Figure 16 : List files

7.2.2 For ECC Keys

1. Create a new directory `ECC_keyfiles/` for keeping ECC reference key files.

2. Make sure `exampleecc.net` exists. If not, create similarly to `example.net`.
3. Generate the key file for KSK.

_ Console

```
/usr/local/bin/dnssec-keyfromlabel -f KSK -a ECDSAP256SHA256 -l  
"pkcs11:token=Bind9_EC;object=ksk" exampleecc.net
```

Where the parameter:

- `-l` specifies the key label in pkcs11 URI format.
- `-f` specifies the key flag.
- `-a` is the algorithm.
- `exampleecc.net` is the name of zone file.

```
[admin@localhost ECC_keyfiles]$  
[admin@localhost ECC_keyfiles]$ sudo -E /usr/local/bin/dnssec-keyfromlabel -f KSK -a ECDSAP256SHA256 -l "pkcs11:  
:token=Bind9_EC;object=ksk" exampleecc.net  
Enter PIN for PKCS#11 Token (Slot 0 - 3001@172.31.1.61 - CLUSTER_0000 - SLOT_0000):  
Enter PIN for PKCS#11 Token (Slot 1 - 3001@172.31.1.61 - CLUSTER_0000 - SLOT_0001):  
Kexampleecc.net.+013+15983  
[admin@localhost ECC_keyfiles]$  
[admin@localhost ECC_keyfiles]$ ls -lh  
total 8.0K  
-rw-r--r-- 1 root root 347 Jul 1 10:27 Kexampleecc.net.+013+15983.key  
-rw----- 1 root root 182 Jul 1 10:27 Kexampleecc.net.+013+15983.private  
[admin@localhost ECC_keyfiles]$
```

Figure 17 : Key file generation for the KSK key

4. Generate the key file for ZSK.

_ Console

```
/usr/local/bin/dnssec-keyfromlabel -a ECDSAP256SHA256 -l  
"pkcs11:token=Bind9_EC;object=zsk" exampleecc.net
```

```
[admin@localhost ECC_keyfiles]$
[admin@localhost ECC_keyfiles]$
[admin@localhost ECC_keyfiles]$ sudo -E /usr/local/bin/dnssec-keyfromlabel -a ECDSA256SHA256 -l "pkcs11:token=
Bind9_EC;object=zsk" exampleecc.net
Enter PIN for PKCS#11 Token (Slot 0 - 3001@172.31.1.61 - CLUSTER_0000 - SLOT_0000):
Enter PIN for PKCS#11 Token (Slot 1 - 3001@172.31.1.61 - CLUSTER_0000 - SLOT_0001):
Kexampleecc.net.+013+23348
[admin@localhost ECC_keyfiles]$
[admin@localhost ECC_keyfiles]$ ls -lh
total 16K
-rw-r--r-- 1 root root 347 Jul  1 10:27 Kexampleecc.net.+013+15983.key
-rw----- 1 root root 182 Jul  1 10:27 Kexampleecc.net.+013+15983.private
-rw-r--r-- 1 root root 348 Jul  1 10:39 Kexampleecc.net.+013+23348.key
-rw----- 1 root root 182 Jul  1 10:39 Kexampleecc.net.+013+23348.private
[admin@localhost ECC_keyfiles]$
[admin@localhost ECC_keyfiles]$
```

Figure 18 : Key file generation for the ZSK key

Where the parameter:

- -l specifies the key label in pkcs11 URI format.
- -f specifies the key flag.
- -a is the algorithm.
- `exampleecc.net` is the name of zone file.

3. Verify that you have two KSK and two ZSK key files available.

› Console

```
# ls -lh
```

```
[admin@localhost ECC_keyfiles]$ ls -lh
total 16K
-rw-r--r-- 1 root root 347 Jul  1 10:27 Kexampleecc.net.+013+15983.key
-rw----- 1 root root 182 Jul  1 10:27 Kexampleecc.net.+013+15983.private
-rw-r--r-- 1 root root 348 Jul  1 10:39 Kexampleecc.net.+013+23348.key
-rw----- 1 root root 182 Jul  1 10:39 Kexampleecc.net.+013+23348.private
[admin@localhost ECC_keyfiles]$
```

Figure 19 : List files



The `.private` file contains a reference to the HSM object, not the private key itself.

7.3 Sign and Verify the Zone file

7.3.1 Sign and Verify Zone Files with RSA Keys

1. Perform zone signing.

_ Console

```
/usr/local/bin/dnssec-signzone -S -o example.net /var/named/example.net
```

```
[admin@localhost RSA_keyfiles]$ sudo -E /usr/local/bin/dnssec-signzone -S -o example.net /var/named/example.net
Enter PIN for PKCS#11 Token (Slot 0 - 3001@172.31.1.61 - CLUSTER_0000 - SLOT_0000):
Fetching example.net/RSASHA256/44141 (KSK) from key repository.
Fetching example.net/RSASHA256/22449 (ZSK) from key repository.
Verifying the zone using the following algorithms:
- RSASHA256
Zone fully signed:
Algorithm: RSASHA256: KSKs: 1 active, 0 stand-by, 0 revoked
                  ZSKs: 1 active, 0 stand-by, 0 revoked
/var/named/example.net.signed
[admin@localhost RSA_keyfiles]$
```

Figure 20 : Zone signing

2. This generates the `example.net.signed` file. A DS record file will be automatically generated.

```
[admin@localhost RSA_keyfiles]$
[admin@localhost RSA_keyfiles]$ ls -lh
total 20K
-rw-r--r-- 1 root root 96 Jul 1 10:53 dsset-example.net.
-rw-r--r-- 1 root root 606 Jul 1 10:18 Kexample.net.+008+22449.key
-rw-r--r-- 1 root root 554 Jul 1 10:18 Kexample.net.+008+22449.private
-rw-r--r-- 1 root root 605 Jul 1 10:07 Kexample.net.+008+44141.key
-rw-r--r-- 1 root root 554 Jul 1 10:07 Kexample.net.+008+44141.private
[admin@localhost RSA_keyfiles]$
[admin@localhost RSA_keyfiles]$
```

Figure 21 : DS records files generated

3. Verify the signed zone file. The signed zone file verification is done using the `dnssec-verify` command.

_ Console

```
/usr/local/bin/dnssec-verify -z -o example.net /var/named/example.net.signed  
Kexample.net.+008+44141.key Kexample.net.+008+22449.key
```

```
[admin@localhost RSA_keyfiles]$  
[admin@localhost RSA_keyfiles]$ sudo -E /usr/local/bin/dnssec-verify -z -o example.net /var/named/example.net.signed  
Loading zone 'example.net' from file '/var/named/example.net.signed'  
  
Verifying the zone using the following algorithms:  
- RSASHA256  
Zone fully signed:  
Algorithm: RSASHA256: KSKs: 1 active, 0 stand-by, 0 revoked  
                              ZSKs: 1 active, 0 stand-by, 0 revoked  
[admin@localhost RSA_keyfiles]$  
[admin@localhost RSA_keyfiles]$
```

Figure 22 : Zone verification for RSA key

Where `Kexample.net.+008+44141.key` is the key file generated for the ksk key and `Kexample.net.+008+22449.key` is the key file generated for the zsk key.

7.3.2 Sign and Verify Zone Files with ECC Keys

1. Use the `dnssec-signzone` command below to perform zone signing.

_ Console

```
/usr/local/bin/dnssec-signzone -S -z -o exampleecc.net /var/named/exampleecc.net
```

```
[admin@localhost ECC_keyfiles]$ sudo -E /usr/local/bin/dnssec-signzone -S -z -o exampleecc.net /var/named/exampleecc.net  
Enter PIN for PKCS#11 Token (Slot 0 - 3001@172.31.1.61 - CLUSTER_0000 - SLOT_0000):  
Enter PIN for PKCS#11 Token (Slot 1 - 3001@172.31.1.61 - CLUSTER_0000 - SLOT_0001):  
Fetching exampleecc.net/ECDSAP256SHA256/15983 (KSK) from key repository.  
Fetching exampleecc.net/ECDSAP256SHA256/23348 (ZSK) from key repository.  
Verifying the zone using the following algorithms:  
- ECDSAP256SHA256  
Zone fully signed:  
Algorithm: ECDSAP256SHA256: KSKs: 1 active, 0 stand-by, 0 revoked  
                              ZSKs: 1 active, 0 stand-by, 0 revoked  
/var/named/exampleecc.net.signed  
[admin@localhost ECC_keyfiles]$
```

Figure 23 : Zone signing for ECC key

This generates the `exampleecc.net.signed` file. A DS record file will be automatically generated.

```
[admin@localhost ECC_keyfiles]$
[admin@localhost ECC_keyfiles]$ ls -lh
total 20K
-rw-r--r-- 1 root root 100 Jul  1 11:06 dsset-exampleecc.net.
-rw-r--r-- 1 root root 347 Jul  1 10:27 Kexampleecc.net.+013+15983.key
-rw----- 1 root root 182 Jul  1 10:27 Kexampleecc.net.+013+15983.private
-rw-r--r-- 1 root root 348 Jul  1 10:39 Kexampleecc.net.+013+23348.key
-rw----- 1 root root 182 Jul  1 10:39 Kexampleecc.net.+013+23348.private
[admin@localhost ECC_keyfiles]$
[admin@localhost ECC_keyfiles]$
```

Figure 24 : DS record file details

2. Verify the signed zone file.

› _ Console

```
/usr/local/bin/dnssec-verify -z -o exampleecc.net /var/named/
exampleecc.net.signed Kexampleecc.net.+013+15983.key Kexampleecc.net.
+013+23348.key
```

```
[admin@localhost ECC_keyfiles]$
[admin@localhost ECC_keyfiles]$ sudo -E /usr/local/bin/dnssec-verify -z -o exampleecc.net /var/named/exampleecc
.net.signed Kexampleecc.net.+013+15983.key Kexampleecc.net.+013+23348.key
Loading zone 'exampleecc.net' from file '/var/named/exampleecc.net.signed'

Verifying the zone using the following algorithms:
- ECDSAP256SHA256
Zone fully signed:
Algorithm: ECDSAP256SHA256: KSKs: 1 active, 0 stand-by, 0 revoked
                        ZSKs: 1 active, 0 stand-by, 0 revoked
[admin@localhost ECC_keyfiles]$
[admin@localhost ECC_keyfiles]$
```

Figure 25 : Zone verification for ECC key

7.4 Configuring Bind9 to Use Signed Zone File

7.4.1 Using RSA Reference File

1. Once the zone signing is done, you need to make changes in the configuration file `named.conf` as below.

```
[admin@localhost etc]$
[admin@localhost etc]$ diff -Nurb named.conf mod/named.conf
--- named.conf      2026-07-01 03:45:58.634596035 -0700
+++ mod/named.conf   2026-07-01 11:30:18.363903360 -0700
@@ -21,7 +21,7 @@
     };

     recursion no;
-    dnssec-validation no;
+    dnssec-validation auto;

     managed-keys-directory "/var/named/dynamic";

@@ -45,6 +45,6 @@

     zone "example.net" {
         type primary;
-        file "example.net";
+        file "example.net.signed";
     };

[admin@localhost etc]$
[admin@localhost etc]$
```

Figure 26 : Diff details of named.conf

2. Stop the Bind9 service.

```
[admin@localhost ECC_keyfiles]$
[admin@localhost ECC_keyfiles]$
[admin@localhost ECC_keyfiles]$ ps -ef | grep named
root      600038      1  0 04:20 ?        00:00:00 sudo CS_PKCS11_R3_CFG=/home/admin/Bind9/cs_pkcs11_R3.cfg /u
sr/local/sbin/named -f -4 -c /usr/local/etc/named.conf -g
root      600039    600038  0 04:20 ?        00:00:00 /usr/local/sbin/named -f -4 -c /usr/local/etc/named.conf -g
admin     602779    602547  0 11:34 pts/3    00:00:00 grep --color=auto named
[admin@localhost ECC_keyfiles]$
[admin@localhost ECC_keyfiles]$ sudo kill 600039
[sudo] password for admin:
[admin@localhost ECC_keyfiles]$
[admin@localhost ECC_keyfiles]$ ps -ef | grep named
admin     602787    602547  0 11:34 pts/3    00:00:00 grep --color=auto named
[admin@localhost ECC_keyfiles]$
[admin@localhost ECC_keyfiles]$
```

Figure 27 : Stop Bind9

3. Start the Bind9 service using the command below and verify the signed zone is loaded successfully.

_ Console

```
/usr/local/sbin/named -f -4 -g -c /usr/local/etc/named.conf
```

```
[admin@localhost RSA_keyfiles]$
[admin@localhost RSA_keyfiles]$ sudo -E /usr/local/sbin/named -f -4 -g -c /usr/local/etc/named.conf
01-Jul-2026 12:13:24.759 starting BIND 9.20.23 (Stable Release) <id:>
01-Jul-2026 12:13:24.759 running on Linux x86_64 5.14.0-427.42.1.el9_4.x86_64 #1 SMP PREEMPT_DYNAMIC Fri Oct 18
14:35:40 EDT 2024
01-Jul-2026 12:13:24.760 built with 'CC=gcc -m64' '--disable-doh'
01-Jul-2026 12:13:24.760 running as: named -f -4 -g -c /usr/local/etc/named.conf
01-Jul-2026 12:13:24.760 compiled by GCC 11.5.0 20240719 (Red Hat 11.5.0-5)
01-Jul-2026 12:13:24.760 compiled with OpenSSL version: OpenSSL 3.5.1 1 Jul 2025
01-Jul-2026 12:13:24.760 linked to OpenSSL version: OpenSSL 3.5.1 1 Jul 2025
01-Jul-2026 12:13:24.760 compiled with libuv version: 1.42.0
01-Jul-2026 12:13:24.760 linked to libuv version: 1.42.0
01-Jul-2026 12:13:24.760 compiled with liburcu version: 0.12.1
01-Jul-2026 12:13:24.760 compiled with libxml2 version: 2.9.13
01-Jul-2026 12:13:24.760 linked to libxml2 version: 20913
01-Jul-2026 12:13:24.760 compiled with zlib version: 1.2.11
01-Jul-2026 12:13:24.760 linked to zlib version: 1.2.11
01-Jul-2026 12:13:24.760 -----
01-Jul-2026 12:13:24.760 BIND 9 is maintained by Internet Systems Consortium,
01-Jul-2026 12:13:24.760 Inc. (ISC), a non-profit 501(c)(3) public-benefit
01-Jul-2026 12:13:24.760 corporation. Support and training for BIND 9 are
01-Jul-2026 12:13:24.760 available at https://www.isc.org/support
01-Jul-2026 12:13:24.760 -----
01-Jul-2026 12:13:24.760 adjusted limit on open files from 1024 to 524288
01-Jul-2026 12:13:24.760 found 2 CPUs, using 2 worker threads
01-Jul-2026 12:13:24.764 DNSSEC algorithms: RSASHA256 RSASHA512 ECDSA256SHA256 ECDSA384SHA384 ED25519 ED448
01-Jul-2026 12:13:24.764 DS algorithms: SHA-1 SHA-256 SHA-384
01-Jul-2026 12:13:24.764 HMAC algorithms: HMAC-MD5 HMAC-SHA1 HMAC-SHA224 HMAC-SHA256 HMAC-SHA384 HMAC-SHA512
01-Jul-2026 12:13:24.764 TKEY mode 2 support (Diffie-Hellman): no
01-Jul-2026 12:13:24.764 TKEY mode 3 support (GSS-API): no
01-Jul-2026 12:13:24.768 the initial working directory is '/home/admin/Bind9/RSA_keyfiles'
01-Jul-2026 12:13:24.768 loading configuration from '/usr/local/etc/named.conf'
01-Jul-2026 12:13:24.768 the working directory is now '/var/named'
01-Jul-2026 12:13:24.769 using default UDP/IPv4 port range: [9000, 65500]
01-Jul-2026 12:13:24.772 listening on IPv4 interface lo, 127.0.0.1#53
01-Jul-2026 12:13:24.773 listening on IPv4 interface ens192, 172.31.1.75#53
01-Jul-2026 12:13:24.773 Disabling periodic interface re-scans timer
01-Jul-2026 12:13:24.774 generating session key for dynamic DNS
01-Jul-2026 12:13:24.776 using built-in root key for view _default
01-Jul-2026 12:13:24.777 set up managed keys zone for view _default, file '/var/named/dynamic/managed-keys.bind
'
01-Jul-2026 12:13:24.779 not using config file logging statement for logging due to -g option
01-Jul-2026 12:13:24.779 configuring command channel from '/usr/local/etc/rndc.key'
01-Jul-2026 12:13:24.780 command channel listening on 127.0.0.1#953
01-Jul-2026 12:13:24.785 managed-keys-zone: loaded serial 320
01-Jul-2026 12:13:24.787 zone example.net/IN: loaded serial 0 (DNSSEC signed)
01-Jul-2026 12:13:24.787 all zones loaded
01-Jul-2026 12:13:24.787 FIPS mode is disabled
01-Jul-2026 12:13:24.787 running
```

Figure 28 : Start Bind9

4. Verify the Bind9 startup logs and confirm the signed zone is loaded successfully.

7.4.2 Using ECC Reference File

1. Make changes in the configuration file `named.conf` as below.

```
[admin@localhost etc]$
[admin@localhost etc]$ diff -Nurb named.conf mod/named.conf
--- named.conf      2026-07-01 11:57:38.711196783 -0700
+++ mod/named.conf  2026-07-01 11:57:53.766089289 -0700
@@ -21,7 +21,7 @@
     };

     recursion no;
-   dnssec-validation no;
+   dnssec-validation auto;

     managed-keys-directory "/var/named/dynamic";

@@ -43,8 +43,8 @@
     file "named.ca";
 };

-zone "example.net" {
+zone "exampleecc.net" {
     type primary;
-   file "example.net";
+   file "exampleecc.net.signed";
 };

[admin@localhost etc]$
[admin@localhost etc]$
```

Figure 29 : Diff details of file named.conf

2. Stop the Bind9 service.

```
[admin@localhost etc]$
[admin@localhost etc]$ ps -ef | grep named
root      603059   602547   0 11:51 pts/3      00:00:00 sudo -E /usr/local/sbin/named -f -4 -g -c /usr/local/etc/named.conf
root      603060   603059   0 11:51 pts/3      00:00:00 /usr/local/sbin/named -f -4 -g -c /usr/local/etc/named.conf
admin     603168   602634   0 12:02 pts/4      00:00:00 grep --color=auto named
[admin@localhost etc]$
[admin@localhost etc]$ sudo kill 603060
[admin@localhost etc]$
[admin@localhost etc]$ ps -ef | grep named
admin     603172   602634   0 12:02 pts/4      00:00:00 grep --color=auto named
[admin@localhost etc]$
[admin@localhost etc]$
```

Figure 30 : Stop Bind9

3. Start the Bind9 service using the command below and verify the signed zone is loaded successfully.

› Console

```
/usr/local/sbin/named -f -4 -g -c /usr/local/etc/named.conf
```

```
[admin@localhost ECC_keyfiles]$
[admin@localhost ECC_keyfiles]$
[admin@localhost ECC_keyfiles]$ sudo ls
[sudo] password for admin:
dsset-exampleecc.net.      Kexampleecc.net.+013+15983.private  Kexampleecc.net.+013+23348.private
Kexampleecc.net.+013+15983.key  Kexampleecc.net.+013+23348.key
[admin@localhost ECC_keyfiles]$
[admin@localhost ECC_keyfiles]$
[admin@localhost ECC_keyfiles]$ sudo -E /usr/local/sbin/named -f -4 -g -c /usr/local/etc/named.conf
01-Jul-2026 12:05:59.168 starting BIND 9.20.23 (Stable Release) <id:>
01-Jul-2026 12:05:59.169 running on Linux x86_64 5.14.0-427.42.1.el9_4.x86_64 #1 SMP PREEMPT_DYNAMIC Fri Oct 18
14:35:40 EDT 2024
01-Jul-2026 12:05:59.169 built with 'CC=gcc -m64' '--disable-doh'
01-Jul-2026 12:05:59.169 running as: named -f -4 -g -c /usr/local/etc/named.conf
01-Jul-2026 12:05:59.169 compiled by GCC 11.5.0 20240719 (Red Hat 11.5.0-5)
01-Jul-2026 12:05:59.169 compiled with OpenSSL version: OpenSSL 3.5.1 1 Jul 2025
01-Jul-2026 12:05:59.170 linked to OpenSSL version: OpenSSL 3.5.1 1 Jul 2025
01-Jul-2026 12:05:59.170 compiled with libuv version: 1.42.0
01-Jul-2026 12:05:59.170 linked to libuv version: 1.42.0
01-Jul-2026 12:05:59.170 compiled with liburcu version: 0.12.1
01-Jul-2026 12:05:59.170 compiled with libxml2 version: 2.9.13
01-Jul-2026 12:05:59.170 linked to libxml2 version: 20913
01-Jul-2026 12:05:59.170 compiled with zlib version: 1.2.11
01-Jul-2026 12:05:59.170 linked to zlib version: 1.2.11
01-Jul-2026 12:05:59.170 -----
01-Jul-2026 12:05:59.170 BIND 9 is maintained by Internet Systems Consortium,
01-Jul-2026 12:05:59.170 Inc. (ISC), a non-profit 501(c)(3) public-benefit
01-Jul-2026 12:05:59.171 corporation. Support and training for BIND 9 are
01-Jul-2026 12:05:59.171 available at https://www.isc.org/support
01-Jul-2026 12:05:59.171 -----
01-Jul-2026 12:05:59.171 adjusted limit on open files from 1024 to 524288
01-Jul-2026 12:05:59.171 found 2 CPUs, using 2 worker threads
01-Jul-2026 12:05:59.176 DNSSEC algorithms: RSASHA256 RSASHA512 ECDSAP256SHA256 ECDSAP384SHA384 ED25519 ED448
01-Jul-2026 12:05:59.176 DS algorithms: SHA-1 SHA-256 SHA-384
01-Jul-2026 12:05:59.176 HMAC algorithms: HMAC-MD5 HMAC-SHA1 HMAC-SHA224 HMAC-SHA256 HMAC-SHA384 HMAC-SHA512
01-Jul-2026 12:05:59.176 TKEY mode 2 support (Diffie-Hellman): no
01-Jul-2026 12:05:59.176 TKEY mode 3 support (GSS-API): no
01-Jul-2026 12:05:59.181 the initial working directory is '/home/admin/Bind9/ECC_keyfiles'
01-Jul-2026 12:05:59.181 loading configuration from '/usr/local/etc/named.conf'
01-Jul-2026 12:05:59.181 the working directory is now '/var/named'
01-Jul-2026 12:05:59.182 using default UDP/IPv4 port range: [9000, 65500]
01-Jul-2026 12:05:59.185 listening on IPv4 interface lo, 127.0.0.1#53
01-Jul-2026 12:05:59.185 listening on IPv4 interface ens192, 172.31.1.75#53
01-Jul-2026 12:05:59.186 Disabling periodic interface re-scans timer
01-Jul-2026 12:05:59.186 generating session key for dynamic DNS
01-Jul-2026 12:05:59.188 using built-in root key for view _default
01-Jul-2026 12:05:59.189 set up managed keys zone for view _default, file '/var/named/dynamic/managed-keys.bind'
,
01-Jul-2026 12:05:59.191 not using config file logging statement for logging due to -g option
01-Jul-2026 12:05:59.191 configuring command channel from '/usr/local/etc/rndc.key'
01-Jul-2026 12:05:59.191 command channel listening on 127.0.0.1#953
01-Jul-2026 12:05:59.197 managed-keys-zone: loaded serial 319
01-Jul-2026 12:05:59.331 zone exampleecc.net/IN: loaded serial 0 (DNSSEC signed)
01-Jul-2026 12:05:59.332 all zones loaded
01-Jul-2026 12:05:59.332 FIPS mode is disabled
01-Jul-2026 12:05:59.332 running
```

Figure 31 : Start Bind9

7.5 Key Rollover for KSK and ZSK

7.5.1 Generate new KSK and ZSK

1. Generate a new key pair as ksk1 and zsk1 on HSM as shown below. Here the KSK key label used is ksk1 and ZSK key label is zsk1.

```
pkcs11-tool --module /opt/utimaco/lib/libcs_pkcs11_R3.so --slot 0 --login --
keypairgen --key-type RSA:2048 --label ksk1 --id 03

pkcs11-tool --module /opt/utimaco/lib/libcs_pkcs11_R3.so --slot 0 --login --
keypairgen --key-type RSA:2048 --label zsk1 --id 04
```

```
[admin@localhost ~]$
[admin@localhost ~]$ pkcs11-tool --module /opt/utimaco/lib/libcs_pkcs11_R3.so --slot 0 --login --keypairgen --k
ey-type RSA:2048 --label ksk1 --id 03
Logging in to "Bind9_RSA".
Please enter User PIN:
Key pair generated:
Private Key Object; RSA
  label:      ksk1
  ID:         03
  Usage:      decrypt, sign, unwrap
  Access:     sensitive, always sensitive, never extractable, local
  Unique ID:  [redacted]
  uri:        pkcs11:model=CryptoServer;manufacturer=Utimaco%20IS%20GmbH;serial=SI003001_0000;token=Bind9_RSA;i
d=%03;object=ksk1;type=private
Public Key Object; RSA 2048 bits
  label:      ksk1
  ID:         03
  Usage:      encrypt, verify, wrap
  Access:     local
  Unique ID:  [redacted]
  uri:        pkcs11:model=CryptoServer;manufacturer=Utimaco%20IS%20GmbH;serial=SI003001_0000;token=Bind9_RSA;i
d=%03;object=ksk1;type=public
[admin@localhost ~]$
[admin@localhost ~]$
```

Figure 32 : New KSK generated

```
[admin@localhost ~]$
[admin@localhost ~]$ pkcs11-tool --module /opt/utimaco/lib/libcs_pkcs11_R3.so --slot 0 --login --keypairgen --key-type RSA:2048 --label zsk1 --id 04
Logging in to "Bind9 RSA".
Please enter User PIN:
Key pair generated:
Private Key Object; RSA
label:      zsk1
ID:         04
Usage:      decrypt, sign, unwrap
Access:     sensitive, always sensitive, never extractable, local
Unique ID:  i+`BEK+`-a`iXU
uri:        pkcs11:model=CryptoServer;manufacturer=Utimaco%20IS%20GmbH;serial=SI003001_0000;token=Bind9_RSA;id=%04;object=zsk1;type=private
Public Key Object; RSA 2048 bits
label:      zsk1
ID:         04
Usage:      encrypt, verify, wrap
Access:     local
Unique ID:  K`-`#L`FtO)`
uri:        pkcs11:model=CryptoServer;manufacturer=Utimaco%20IS%20GmbH;serial=SI003001_0000;token=Bind9_RSA;id=%04;object=zsk1;type=public
[admin@localhost ~]$
[admin@localhost ~]$
```

Figure 33 : New ZSK generated

2. Verify the generated keys in HSM.

```
p11tool2 slot=0 LoginUser=ask ListObjects
```

```
[admin@localhost Administration]$
[admin@localhost Administration]$ ./p11tool2 slot=0 LoginUser=ask ListObjects
Enter normal user PIN:

CKO_PUBLIC_KEY:

+ 1.1
  CKA_KEY_TYPE           = CKK_RSA
  CKA_UNIQUE_ID          = 4B8E2DBC-231F-4CFE-AE46-19744F7DBFB4
  CKA_LABEL              = zsk1
  CKA_ID                 = 0x04 ()

+ 1.2
  CKA_KEY_TYPE           = CKK_RSA
  CKA_UNIQUE_ID          = 47F3D5BF-A118-44DC-8B19-0DAE9EF76E98
  CKA_LABEL              = ksk
  CKA_ID                 = 0x01 ()

+ 1.3
  CKA_KEY_TYPE           = CKK_RSA
  CKA_UNIQUE_ID          = D20F49D3-F4A9-4FB4-8D5A-DAA846CD9E83
  CKA_LABEL              = zsk
  CKA_ID                 = 0x02 ()

+ 1.4
  CKA_KEY_TYPE           = CKK_RSA
  CKA_UNIQUE_ID          = F9A31E4E-F144-4E96-98C5-C11E0118FBE2
  CKA_LABEL              = ksk1
  CKA_ID                 = 0x03 ()

CKO_PRIVATE_KEY:

+ 2.1
  CKA_KEY_TYPE           = CKK_RSA
  CKA_UNIQUE_ID          = 80723E0B-EE2F-4348-B4FC-5B218ED76F40
  CKA_SENSITIVE          = CK_TRUE
  CKA_EXTRACTABLE        = CK_FALSE
  CKA_LABEL              = ksk1
  CKA_ID                 = 0x03 ()

+ 2.2
  CKA_KEY_TYPE           = CKK_RSA
  CKA_UNIQUE_ID          = CE692B60-4245-4B2B-A2A7-2D6194695855
  CKA_SENSITIVE          = CK_TRUE
  CKA_EXTRACTABLE        = CK_FALSE
  CKA_LABEL              = zsk1
  CKA_ID                 = 0x04 ()

+ 2.3
  CKA_KEY_TYPE           = CKK_RSA
  CKA_UNIQUE_ID          = 7158C7FD-CC7C-451E-9040-420F4C48B2A4
  CKA_SENSITIVE          = CK_TRUE
  CKA_EXTRACTABLE        = CK_FALSE
  CKA_LABEL              = zsk
  CKA_ID                 = 0x02 ()

+ 2.4
  CKA_KEY_TYPE           = CKK_RSA
  CKA_UNIQUE_ID          = 397AD08B-8E00-4D81-9EE2-CA97CC430AD6
  CKA_SENSITIVE          = CK_TRUE
  CKA_EXTRACTABLE        = CK_FALSE
  CKA_LABEL              = ksk
  CKA_ID                 = 0x01 ()

[admin@localhost Administration]$
```

Figure 34 : Keys updated in HSM

7.5.2 KSK Rollover

1. Generate a key file for the new KSK.

_ Console

```
/usr/local/bin/dnssec-keyfromlabel -f KSK -a RSASHA256 -l  
"pkcs11:token=Bind9_RSA;object=ksk1" example.net
```

```
[admin@localhost RSA_keyfiles]$  
[admin@localhost RSA_keyfiles]$ ls -lh  
total 20K  
-rw-r--r-- 1 root root 96 Jul 1 10:53 dsset-example.net.  
-rw-r--r-- 1 root root 606 Jul 1 10:18 Kexample.net.+008+22449.key  
-rw----- 1 root root 554 Jul 1 10:18 Kexample.net.+008+22449.private  
-rw-r--r-- 1 root root 605 Jul 1 10:07 Kexample.net.+008+44141.key  
-rw----- 1 root root 554 Jul 1 10:07 Kexample.net.+008+44141.private  
[admin@localhost RSA_keyfiles]$  
[admin@localhost RSA_keyfiles]$ sudo -E /usr/local/bin/dnssec-keyfromlabel -f KSK -a RSASHA256 -l "pkcs11:token  
=Bind9_RSA;object=ksk1" example.net  
Enter PIN for PKCS#11 Token (Slot 0 - 3001@172.31.1.61 - CLUSTER_0000 - SLOT_0000):  
Kexample.net.+008+26066  
[admin@localhost RSA_keyfiles]$  
[admin@localhost RSA_keyfiles]$ ls -lh  
total 28K  
-rw-r--r-- 1 root root 96 Jul 1 10:53 dsset-example.net.  
-rw-r--r-- 1 root root 606 Jul 1 10:18 Kexample.net.+008+22449.key  
-rw----- 1 root root 554 Jul 1 10:18 Kexample.net.+008+22449.private  
-rw-r--r-- 1 root root 605 Jul 1 12:50 Kexample.net.+008+26066.key  
-rw----- 1 root root 554 Jul 1 12:50 Kexample.net.+008+26066.private  
-rw-r--r-- 1 root root 605 Jul 1 10:07 Kexample.net.+008+44141.key  
-rw----- 1 root root 554 Jul 1 10:07 Kexample.net.+008+44141.private  
[admin@localhost RSA_keyfiles]$
```

Figure 35 : Key file for new KSK

2. Sign the zone with the old and new KSK.

_ Console

```
/usr/local/bin/dnssec-signzone -S -o example.net /var/named/example.net
```

```
[admin@localhost RSA_keyfiles]$
[admin@localhost RSA_keyfiles]$ sudo -E /usr/local/bin/dnssec-signzone -S -o example.net /var/named/example.net
Enter PIN for PKCS#11 Token (Slot 0 - 3001@172.31.1.61 - CLUSTER_0000 - SLOT_0000):
Fetching example.net/RSASHA256/44141 (KSK) from key repository.
Fetching example.net/RSASHA256/22449 (ZSK) from key repository.
Fetching example.net/RSASHA256/26066 (KSK) from key repository.
Verifying the zone using the following algorithms:
- RSASHA256
Zone fully signed:
Algorithm: RSASHA256: KSKs: 2 active, 0 stand-by, 0 revoked
                        ZSKs: 1 active, 0 stand-by, 0 revoked
/var/named/example.net.signed
[admin@localhost RSA_keyfiles]$
[admin@localhost RSA_keyfiles]$ ls -lh
total 28K
-rw-r--r-- 1 root root 192 Jul  1 12:52 dsset-example.net.
-rw-r--r-- 1 root root 606 Jul  1 10:18 Kexample.net.+008+22449.key
-rw----- 1 root root 554 Jul  1 10:18 Kexample.net.+008+22449.private
-rw-r--r-- 1 root root 605 Jul  1 12:50 Kexample.net.+008+26066.key
-rw----- 1 root root 554 Jul  1 12:50 Kexample.net.+008+26066.private
-rw-r--r-- 1 root root 605 Jul  1 10:07 Kexample.net.+008+44141.key
-rw----- 1 root root 554 Jul  1 10:07 Kexample.net.+008+44141.private
[admin@localhost RSA_keyfiles]$
[admin@localhost RSA_keyfiles]$ ls -lh /var/named/example.net.signed
-rw-r--r-- 1 root root 8.9K Jul  1 12:52 /var/named/example.net.signed
[admin@localhost RSA_keyfiles]$
```

Figure 36 : Signing zone with old and new KSK

3. Wait for the zone transfer time, TTL of DNSKEY resource record set and TTL on the DS record set.
4. Sign the zone with the new KSK.

› _ Console

```
sudo /usr/local/bin/dnssec-signzone -S -o example.net /var/named/example.net
```

```
[admin@localhost RSA_keyfiles]$ sudo ls -lh
total 20K
-rw-r--r-- 1 root root 192 Jul  1 12:52 dsset-example.net.
-rw-r--r-- 1 root root 606 Jul  1 10:18 Kexample.net.+008+22449.key
-rw----- 1 root root 554 Jul  1 10:18 Kexample.net.+008+22449.private
-rw-r--r-- 1 root root 605 Jul  1 12:50 Kexample.net.+008+26066.key
-rw----- 1 root root 554 Jul  1 12:50 Kexample.net.+008+26066.private
[admin@localhost RSA_keyfiles]$
[admin@localhost RSA_keyfiles]$ sudo -E /usr/local/bin/dnssec-signzone -S -o example.net /var/named/example.net
Enter PIN for PKCS#11 Token (Slot 0 - 3001@172.31.1.61 - CLUSTER_0000 - SLOT_0000):
Fetching example.net/RSASHA256/22449 (ZSK) from key repository.
Fetching example.net/RSASHA256/26066 (KSK) from key repository.
Verifying the zone using the following algorithms:
- RSASHA256
Zone fully signed:
Algorithm: RSASHA256: KSKs: 1 active, 0 stand-by, 0 revoked
                  ZSKs: 1 active, 0 stand-by, 0 revoked
/var/named/example.net.signed
[admin@localhost RSA_keyfiles]$
[admin@localhost RSA_keyfiles]$ ls -lh
total 20K
-rw-r--r-- 1 root root  96 Jul  1 13:08 dsset-example.net.
-rw-r--r-- 1 root root 606 Jul  1 10:18 Kexample.net.+008+22449.key
-rw----- 1 root root 554 Jul  1 10:18 Kexample.net.+008+22449.private
-rw-r--r-- 1 root root 605 Jul  1 12:50 Kexample.net.+008+26066.key
-rw----- 1 root root 554 Jul  1 12:50 Kexample.net.+008+26066.private
[admin@localhost RSA_keyfiles]$
```

Figure 37 : Signing the zone with new KSK

7.5.3 ZSK Rollover

1. Generate key file for the new ZSK.

›_ Console

```
/usr/local/bin/dnssec-keyfromlabel -a RSASHA256 -l
"pkcs11:token=Bind9_RSA;object=zsk1" example.net
```

```
[admin@localhost RSA_keyfiles]$
[admin@localhost RSA_keyfiles]$ ls -lh
total 20K
-rw-r--r-- 1 root root 96 Jul 1 13:08 dsset-example.net.
-rw-r--r-- 1 root root 606 Jul 1 10:18 Kexample.net.+008+22449.key
-rw----- 1 root root 554 Jul 1 10:18 Kexample.net.+008+22449.private
-rw-r--r-- 1 root root 605 Jul 1 12:50 Kexample.net.+008+26066.key
-rw----- 1 root root 554 Jul 1 12:50 Kexample.net.+008+26066.private
[admin@localhost RSA_keyfiles]$
[admin@localhost RSA_keyfiles]$ sudo -E /usr/local/bin/dnssec-keyfromlabel -a RSASHA256 -l "pkcs11:token=Bind9_
RSA;object=zsk1" example.net
Enter PIN for PKCS#11 Token (Slot 0 - 3001@172.31.1.61 - CLUSTER_0000 - SLOT_0000):
Kexample.net.+008+27769
[admin@localhost RSA_keyfiles]$
[admin@localhost RSA_keyfiles]$ ls -lh
total 28K
-rw-r--r-- 1 root root 96 Jul 1 13:08 dsset-example.net.
-rw-r--r-- 1 root root 606 Jul 1 10:18 Kexample.net.+008+22449.key
-rw----- 1 root root 554 Jul 1 10:18 Kexample.net.+008+22449.private
-rw-r--r-- 1 root root 605 Jul 1 12:50 Kexample.net.+008+26066.key
-rw----- 1 root root 554 Jul 1 12:50 Kexample.net.+008+26066.private
-rw-r--r-- 1 root root 606 Jul 1 13:16 Kexample.net.+008+27769.key
-rw----- 1 root root 554 Jul 1 13:16 Kexample.net.+008+27769.private
[admin@localhost RSA_keyfiles]$
[admin@localhost RSA_keyfiles]$
```

Figure 38 : Key file for new ZSK

2. Sign the zone with the new and old ZSK.

›_ Console

```
/usr/local/bin/dnssec-signzone -S -o example.net /var/named/example.net
```



```
[admin@localhost RSA_keyfiles]$ ls -lh
total 28K
-rw-r--r-- 1 root root 96 Jul 1 13:08 dsset-example.net.
-rw-r--r-- 1 root root 606 Jul 1 10:18 Kexample.net.+008+22449.key
-rw----- 1 root root 554 Jul 1 10:18 Kexample.net.+008+22449.private
-rw-r--r-- 1 root root 605 Jul 1 12:50 Kexample.net.+008+26066.key
-rw----- 1 root root 554 Jul 1 12:50 Kexample.net.+008+26066.private
-rw-r--r-- 1 root root 606 Jul 1 13:16 Kexample.net.+008+27769.key
-rw----- 1 root root 554 Jul 1 13:16 Kexample.net.+008+27769.private
[admin@localhost RSA_keyfiles]$
[admin@localhost RSA_keyfiles]$ sudo -E /usr/local/bin/dnssec-signzone -S -o example.net /var/named/example.net
Enter PIN for PKCS#11 Token (Slot 0 - 3001@172.31.1.61 - CLUSTER_0000 - SLOT_0000):
Fetching example.net/RSASHA256/22449 (ZSK) from key repository.
Fetching example.net/RSASHA256/26066 (KSK) from key repository.
Fetching example.net/RSASHA256/27769 (ZSK) from key repository.
Verifying the zone using the following algorithms:
- RSASHA256
Zone fully signed:
Algorithm: RSASHA256: KSKs: 1 active, 0 stand-by, 0 revoked
                ZSKs: 2 active, 0 stand-by, 0 revoked
/var/named/example.net.signed
[admin@localhost RSA_keyfiles]$
[admin@localhost RSA_keyfiles]$ ls -lh
total 28K
-rw-r--r-- 1 root root 96 Jul 1 13:22 dsset-example.net.
-rw-r--r-- 1 root root 606 Jul 1 10:18 Kexample.net.+008+22449.key
-rw----- 1 root root 554 Jul 1 10:18 Kexample.net.+008+22449.private
-rw-r--r-- 1 root root 605 Jul 1 12:50 Kexample.net.+008+26066.key
-rw----- 1 root root 554 Jul 1 12:50 Kexample.net.+008+26066.private
-rw-r--r-- 1 root root 606 Jul 1 13:16 Kexample.net.+008+27769.key
-rw----- 1 root root 554 Jul 1 13:16 Kexample.net.+008+27769.private
[admin@localhost RSA_keyfiles]$
[admin@localhost RSA_keyfiles]$ ls -lh /var/named/example.net.signed
-rw-r--r-- 1 root root 15K Jul 1 13:22 /var/named/example.net.signed
[admin@localhost RSA_keyfiles]$
```

Figure 39 : Signing zone with old and new ZSK

- Wait for the zone transfer time and maximum TTL used in the zone.
- Re-sign the zone with the new ZSK. Now we have only one ZSK in `example.net` so it will automatically pick this new ZSK for signing zone.

› Console

```
/usr/local/bin/dnssec-signzone -S -o example.net /var/named/example.net
```

```
[admin@localhost RSA_keyfiles]$
[admin@localhost RSA_keyfiles]$ ls -lh
total 20K
-rw-r--r-- 1 root root 96 Jul 1 13:22 dsset-example.net.
-rw-r--r-- 1 root root 605 Jul 1 12:50 Kexample.net.+008+26066.key
-rw----- 1 root root 554 Jul 1 12:50 Kexample.net.+008+26066.private
-rw-r--r-- 1 root root 606 Jul 1 13:16 Kexample.net.+008+27769.key
-rw----- 1 root root 554 Jul 1 13:16 Kexample.net.+008+27769.private
[admin@localhost RSA_keyfiles]$
[admin@localhost RSA_keyfiles]$ sudo -E /usr/local/bin/dnssec-signzone -S -o example.net /var/named/example.net
Enter PIN for PKCS#11 Token (Slot 0 - 3001@172.31.1.61 - CLUSTER_0000 - SLOT_0000):
Fetching example.net/RSASHA256/26066 (KSK) from key repository.
Fetching example.net/RSASHA256/27769 (ZSK) from key repository.
Verifying the zone using the following algorithms:
- RSASHA256
Zone fully signed:
Algorithm: RSASHA256: KSKs: 1 active, 0 stand-by, 0 revoked
                        ZSKs: 1 active, 0 stand-by, 0 revoked
/var/named/example.net.signed
Segmentation fault
[admin@localhost RSA_keyfiles]$
[admin@localhost RSA_keyfiles]$ ls -lh
total 20K
-rw-r--r-- 1 root root 96 Jul 1 13:36 dsset-example.net.
-rw-r--r-- 1 root root 605 Jul 1 12:50 Kexample.net.+008+26066.key
-rw----- 1 root root 554 Jul 1 12:50 Kexample.net.+008+26066.private
-rw-r--r-- 1 root root 606 Jul 1 13:16 Kexample.net.+008+27769.key
-rw----- 1 root root 554 Jul 1 13:16 Kexample.net.+008+27769.private
[admin@localhost RSA_keyfiles]$
[admin@localhost RSA_keyfiles]$ ls -lh /var/named/example.net.signed
-rw-r--r-- 1 root root 7.9K Jul 1 13:36 /var/named/example.net.signed
[admin@localhost RSA_keyfiles]$
```

Figure 40 : Signing the zone with new ZSK

5. Verify new signed zone.

```
/usr/local/bin/dnssec-verify -z -o example.net /var/named/example.net.signed
```

```
[admin@localhost RSA_keyfiles]$
[admin@localhost RSA_keyfiles]$ sudo -E /usr/local/bin/dnssec-verify -z -o example.net /var/named/example.net.signed
Loading zone 'example.net' from file '/var/named/example.net.signed'

Verifying the zone using the following algorithms:
- RSASHA256
Zone fully signed:
Algorithm: RSASHA256: KSKs: 1 active, 0 stand-by, 0 revoked
                        ZSKs: 1 active, 0 stand-by, 0 revoked
[admin@localhost RSA_keyfiles]$
[admin@localhost RSA_keyfiles]$
```

Figure 41 : New signed zone

6. Restart the named service using below command.

› Console

```
/usr/local/sbin/named -f -4 -g -c /usr/local/etc/named.conf
```

```
[admin@localhost RSA_keyfiles]$
[admin@localhost RSA_keyfiles]$ sudo -E /usr/local/sbin/named -f -4 -g -c /usr/local/etc/named.conf
01-Jul-2026 13:41:00.075 starting BIND 9.20.23 (Stable Release) <id:>
01-Jul-2026 13:41:00.075 running on Linux x86_64 5.14.0-427.42.1.el9_4.x86_64 #1 SMP PREEMPT_DYNAMIC Fri Oct 18
 14:35:40 EDT 2024
01-Jul-2026 13:41:00.075 built with 'CC=gcc -m64' '--disable-doh'
01-Jul-2026 13:41:00.075 running as: named -f -4 -g -c /usr/local/etc/named.conf
01-Jul-2026 13:41:00.075 compiled by GCC 11.5.0 20240719 (Red Hat 11.5.0-5)
01-Jul-2026 13:41:00.075 compiled with OpenSSL version: OpenSSL 3.5.1 1 Jul 2025
01-Jul-2026 13:41:00.075 linked to OpenSSL version: OpenSSL 3.5.1 1 Jul 2025
01-Jul-2026 13:41:00.075 compiled with libuv version: 1.42.0
01-Jul-2026 13:41:00.075 linked to libuv version: 1.42.0
01-Jul-2026 13:41:00.075 compiled with liburcu version: 0.12.1
01-Jul-2026 13:41:00.075 compiled with libxml2 version: 2.9.13
01-Jul-2026 13:41:00.075 linked to libxml2 version: 20913
01-Jul-2026 13:41:00.075 compiled with zlib version: 1.2.11
01-Jul-2026 13:41:00.075 linked to zlib version: 1.2.11
01-Jul-2026 13:41:00.075 -----
01-Jul-2026 13:41:00.075 BIND 9 is maintained by Internet Systems Consortium,
01-Jul-2026 13:41:00.075 Inc. (ISC), a non-profit 501(c)(3) public-benefit
01-Jul-2026 13:41:00.075 corporation. Support and training for BIND 9 are
01-Jul-2026 13:41:00.075 available at https://www.isc.org/support
01-Jul-2026 13:41:00.075 -----
01-Jul-2026 13:41:00.075 adjusted limit on open files from 1024 to 524288
01-Jul-2026 13:41:00.075 found 2 CPUs, using 2 worker threads
01-Jul-2026 13:41:00.083 DNSSEC algorithms: RSASHA256 RSASHA512 ECDSAP256SHA256 ECDSAP384SHA384 ED25519 ED448
01-Jul-2026 13:41:00.083 DS algorithms: SHA-1 SHA-256 SHA-384
01-Jul-2026 13:41:00.083 HMAC algorithms: HMAC-MD5 HMAC-SHA1 HMAC-SHA224 HMAC-SHA256 HMAC-SHA384 HMAC-SHA512
01-Jul-2026 13:41:00.083 TKEY mode 2 support (Diffie-Hellman): no
01-Jul-2026 13:41:00.084 TKEY mode 3 support (GSS-API): no
01-Jul-2026 13:41:00.094 the initial working directory is '/home/admin/Bind9/RSA_keyfiles'
01-Jul-2026 13:41:00.095 loading configuration from '/usr/local/etc/named.conf'
01-Jul-2026 13:41:00.105 the working directory is now '/var/named'
01-Jul-2026 13:41:00.109 using default UDP/IPv4 port range: [9000, 65500]
01-Jul-2026 13:41:00.114 listening on IPv4 interface lo, 127.0.0.1#53
01-Jul-2026 13:41:00.115 listening on IPv4 interface ens192, 172.31.1.75#53
01-Jul-2026 13:41:00.116 Disabling periodic interface re-scans timer
01-Jul-2026 13:41:00.117 generating session key for dynamic DNS
01-Jul-2026 13:41:00.122 using built-in root key for view _default
01-Jul-2026 13:41:00.122 set up managed keys zone for view _default, file '/var/named/dynamic/managed-keys.bind
',
01-Jul-2026 13:41:00.124 not using config file logging statement for logging due to -g option
01-Jul-2026 13:41:00.124 configuring command channel from '/usr/local/etc/rndc.key'
01-Jul-2026 13:41:00.124 command channel listening on 127.0.0.1#953
01-Jul-2026 13:41:00.126 managed-keys-zone: loaded serial 322
01-Jul-2026 13:41:00.131 zone example.net/IN: loaded serial 0 (DNSSEC signed)
01-Jul-2026 13:41:00.132 all zones loaded
01-Jul-2026 13:41:00.132 FIPS mode is disabled
01-Jul-2026 13:41:00.132 running
```

Figure 42 : Starting named service



This completes the Integration for Bind9 with Utimaco SecurityServer.

8 Troubleshooting

Error	Diagnosis
CKR_USER_PIN_NOT_INITIALIZED	PKCS#11 Slot is not initialized.
CKR_PIN_INCORRECT	Check user PIN.
error: zone example.net/IN (unsigned): loading from master file http:// example.net failed: permission denied.	Provide permission to zone file as <code>chmod -R 750 http://example.net</code> and change the ownership of file to user that is running the named service.

Table 6: List of errors and their diagnoses

9 Further Information

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

All SecurityServer product documentation is also available at the Utimaco IS GmbH website:
<https://utimaco.com/>.

10 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.

11 Appendix A: References

Reference	Title/Company	Document No.
[CSTrSh]	CryptoServer Troubleshooting/Utlimaco IS GmbH	M011-0008-en
[CSP11Tool2]	CryptoServer_p11tool2_Manual.pdf	2012-0004
[CSPKCSM]	CryptoServer - PKCS#11 P11CAT Manual	M013-0001-en
[CSLAN5]	CryptoServerLAN_Manual_Systemadministrators.pdf	2018-0004
https://bind9.readthedocs.io/en/latest/dnssec-guide.html	BIND 9 DNSSEC Guide (ISC Official Documentation)	NA
https://integration-guides.utimaco.com/openssl-3.5-and-u.trust-gp-hsm-se-series/1.0.0/installing-pkcs11-provider	OpenSSL 3.5 and u.trust GP HSM SE-Series – Installing pkcs11-provider / Utimaco	NA

Table 7: References

12 Appendix B: Command Summary

Commands Used	Purpose
<code>mkdir -p /opt/utimaco/bin</code>	Create the Utimaco binary directory.
<code>mkdir -p /opt/utimaco/lib</code>	Create the Utimaco library directory.
<code>cp ~/path_to_application_folder/lib/libcs_pkcs11_R3.so /opt/utimaco/lib</code>	Copy the Utimaco PKCS#11 library to the target location.
<code>cd ~/path_to_application_folder</code>	Change to the SecurityServer installation directory.
<code>cp csadm p11tool2 /opt/utimaco/bin</code>	Copy Utimaco administration utilities.
<code>chmod +x /opt/utimaco/bin/csadm /opt/utimaco/bin/p11tool2</code>	Make the utilities executable.
<code>cd <install_directory>/Software/Linux/x86-64/Crypto_APIs/PKCS11_R3/sample</code>	Navigate to the PKCS#11 sample configuration directory.
<code>cp cs_pkcs11_R3.cfg /home/admin/Bind9</code>	Copy the PKCS#11 configuration file.
<code>cd /home/admin/Bind9</code>	Change to the Bind9 working directory.
<code>export CS_PKCS11_R3_CFG=/home/admin/Bind9/cs_pkcs11_R3.cfg</code>	Set the PKCS#11 configuration environment variable.
<code>p11tool2 slot=<slot_no> Label=<token_label> Login=ADMIN,ADMIN.key InitToken=ask</code>	Create and initialize an HSM token/slot.

Commands Used	Purpose
<code>p11tool2 slot=<slot_no> LoginS0=ask InitPin=ask</code>	Initialize the Security Officer PIN.
<code>openssl version -a</code>	Verify the installed OpenSSL version.
<code>openssl list -providers</code>	Verify the installed OpenSSL providers, including PKCS#11 provider.
<code>tar -xvf bind9-9.20.23.tar.gz</code>	Extract the Bind9 source package.
<code>autoreconf -fi</code>	Generate the configure scripts for Bind9 build.
<code>dnf install openssl-devel perl libicu- devel pkgconf userspace-rcu-devel libcap-devel libuv-devel</code>	Install Bind9 build dependencies.
<code>configure CC="gcc -m64" -enable-threads -disable-doh</code>	Configure the Bind9 build environment.
<code>make</code>	Build Bind9 from source.
<code>make install</code>	Install the compiled Bind9 software.
<code>/usr/local/sbin/named -version</code>	Check the installed Bind9 version.
<code>/usr/local/sbin/named -f -c /usr/local/ etc/named.conf</code>	Start the Bind9 DNS service.

Commands Used	Purpose
<code>pkcs11-tool --module /opt/utimaco/lib/libcs_pkcs11_R3.so --slot 0 --login --keypairgen --key-type RSA:2048 --label ksk --id 01</code>	Generate an RSA KSK on the HSM.
<code>pkcs11-tool --module /opt/utimaco/lib/libcs_pkcs11_R3.so --slot 0 --login --keypairgen --key-type RSA:2048 --label zsk --id 02</code>	Generate an RSA ZSK on the HSM.
<code>p11tool2 slot=0 LoginUser=ask ListObjects</code>	List objects stored in RSA slot.
<code>pkcs11-tool --module /opt/utimaco/lib/libcs_pkcs11_R3.so --slot 1 --login --keypairgen --key-type EC:prime256v1 --label ksk --id 01</code>	Generate an ECC KSK on the HSM.
<code>pkcs11-tool --module /opt/utimaco/lib/libcs_pkcs11_R3.so --slot 1 --login --keypairgen --key-type EC:prime256v1 --label zsk --id 02</code>	Generate an ECC ZSK on the HSM.
<code>p11tool2 slot=1 LoginUser=ask ListObjects</code>	List objects stored in ECC slot.
<code>/usr/local/bin/dnssec-keyfromlabel -f KSK -a RSASHA256 -l "pkcs11:token=Bind9_RSA;object=ksk" example.net</code>	Generate a Bind9 reference key file for RSA KSK.

Commands Used	Purpose
<pre>/usr/local/bin/dnssec-keyfromlabel -a RSASHA256 -l "pkcs11:token=Bind9_RSA;object=zsk" example.net</pre>	Generate a Bind9 reference key file for RSA ZSK.
<pre>ls -lh</pre>	List generated key files.
<pre>/usr/local/bin/dnssec-keyfromlabel -f KSK -a ECDSAP256SHA256 -l "pkcs11:token=Bind9_EC;object=ksk" exampleecc.net</pre>	Generate a Bind9 reference key file for ECC KSK.
<pre>/usr/local/bin/dnssec-keyfromlabel -a ECDSAP256SHA256 -l "pkcs11:token=Bind9_EC;object=zsk" exampleecc.net</pre>	Generate a Bind9 reference key file for ECC ZSK.
<pre>/usr/local/bin/dnssec-signzone -S -o example.net /var/named/example.net</pre>	Sign a DNS zone using RSA keys.
<pre>/usr/local/bin/dnssec-verify -z -o example.net /var/named/ example.net.signed Kexample.net. +008+44141.key Kexample.net. +008+22449.key</pre>	Verify a signed RSA zone file.
<pre>/usr/local/bin/dnssec-signzone -S -z -o exampleecc.net /var/named/ exampleecc.net</pre>	Sign a DNS zone using ECC keys.

Commands Used	Purpose
<pre>/usr/local/bin/dnssec-verify -z -o exampleecc.net /var/named/ exampleecc.net.signed Kexampleecc.net. +013+15983.key Kexampleecc.net. +013+23348.key</pre>	Verify a signed ECC zone file.
<pre>vi /usr/local/etc/named.conf</pre>	Edit the Bind9 configuration file.
<pre>/usr/local/sbin/named -f -4 -g -c /usr/ local/etc/named.conf</pre>	Start Bind9 with DNSSEC-enabled configuration.
<pre>/usr/local/bin/dnssec-verify -z -o example.net /var/named/ example.net.signed</pre>	Verify the re-signed DNS zone after key rollover.

Table 8: CLI Commands