

Alpaquita Linux

Using Custom Keys in Secure Boot



Alpaquita Linux
Revision 1.0
June 2024

be//soft

Copyright © BellSoft Corporation 2018-2024.

BellSoft software contains open source software. Additional information about third party code is available at https://bell-sw.com/third_party_licenses. You can also get more information on how to get a copy of source code by contacting info@bell-sw.com.

THIS INFORMATION MAY CHANGE WITHOUT NOTICE. TO THE EXTENT PERMITTED BY APPLICABLE LAW, BELLSOFT PROVIDES THIS DOCUMENTATION “AS IS” WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT WILL BELLSOFT BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST INVESTMENT, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF BELLSOFT IS EXPRESSLY ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The use of any software product referenced in this document is governed by the applicable license agreement, which is not modified in any way by the terms of this notice.

Alpaquita, Liberica and BellSoft are trademarks or registered trademarks of BellSoft Corporation. The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis. Java and OpenJDK are trademarks or registered trademarks of Oracle and/or its affiliates. Other trademarks are the property of their respective owners and are used only for identification purposes.

Contents

1. Overview 4

How Secure Boot works in Alpaquita Linux 4

2. Setting up Secure Boot 6

Setting up virtual machine 6

Setting up Alpaquita Linux 7

Switching Secure Boot to Setup mode 7

Create PK, KEK, db keys 9

 Installing required packages 9

 Creating the keys 10

Signing shim 11

Installing keys to Secure Boot database 11

Verifying Secure Boot 12

1. Overview

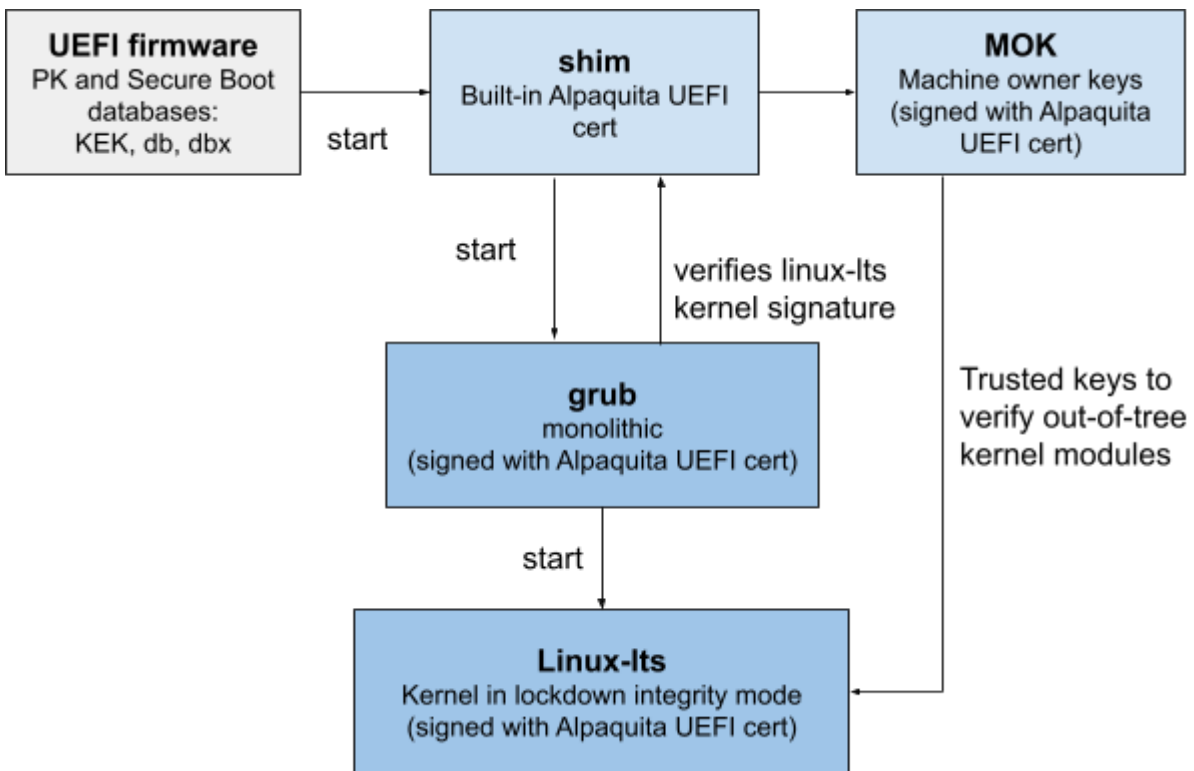
Secure Boot (SB) is a mechanism to boot only trusted software and protect the system from malicious code early in the boot process. Since the introduction of this feature in 2011 by Microsoft, most Unified Extensible Firmware Interface (UEFI) firmware now comes with preloaded Microsoft certificates that only boot software signed by them when SB is enabled. Moreover, many SB systems today allow users to add their own trusted keys to SB databases

Alpaquita Linux contains a Microsoft-signed shim that enables you to run Alpaquita in Secure Boot mode on a x86_64 UEFI system without any preliminary setup.

If you want to use Secure Boot with your own keys, you must sign the shim with your key in Alpaquita and add your trusted keys to the Secure Boot database (UEFI db). After that, you can enable secure boot in your system. For more information, see [Signing shim](#) and [Installing keys to Secure Boot database](#) in this document.

How Secure Boot works in Alpaquita Linux

Alpaquita Linux provides a trusted chain, which includes pre-bootloader "shim" with a built-in Alpaquita UEFI certificate, signed MOKManager utility, grub, and Linux kernel (automatically locked down if SB is enabled).



- **UEFI firmware** is a replacement for the Basic Input/Output System (BIOS) firmware interface. In the Secure Boot context, it has a platform key (PK) and several SB databases that include the signature database (db), revoked signatures database (dbx), and Key Enrollment Key database (KEK).
- **shim** is the first stage bootloader, a root of trust for all other booted components in Alpaquita Linux. It embeds the BellSoft Alpaquita UEFI certificate used to sign further booted software.
- **MOK** (Machine Owner Key) is an extra database of keys or hashes, similar to db, which can be easily managed by a user. There is a mockutil package that can be used to add or remove keys, but changes can only be confirmed and applied at boot time. This allows signing locally built kernel modules, bootloaders or kernels.

As demonstrated in the previous diagram, the shim, the first-stage bootloader, must be signed with the key you control from UEFI Signature Database (db), or the shim hash must be added to db to start Alpaquita Linux in Secure Boot mode. If you use the default key, you can enable Secure Boot without any setup. If you want to use some other key, sign the shim and load the key or hash to the UEFI db. After that, you can enable the secure boot, because the other components in the boot chain are trusted (signed by the trusted key).

To demonstrate shim signing and enrollment of new keys, we will use an x86 virtual machine using QEMU/KVM with UEFI support. Bare-metal machines have their own specific UEFI firmware implementations. Contact your hardware vendor (manufacturer) for details, such as the possibility of changing Secure Boot keys, assistance, and tools you may need.

2. Setting up Secure Boot

The following sections provide detailed information on how to set up Secure Boot for Alpaquita Linux using your own keys.

! Important:

Alpaquita Linux contains a Microsoft-signed shim that enables you to run Alpaquita in Secure Boot mode on a x86_64 UEFI system without any preliminary setup. If you are not sure about any procedure in the following steps, do not alter the default setup. Contact [BellSoft support](#) for more information.

Setting up virtual machine

To proceed, prepare an x86 Linux machine with QEMU/KVM installed and UEFI firmware for the demo VM.

OVMF is Open Virtual Machine Firmware that is a part of the Intel's EFI Development Kit II (edk2). It enables UEFI support for x86 Virtual Machines. To simplify the setup, use the edk2-ovmf or ovmf package that can be found in many distributions. Download Fedora edk2-ovmf package, which is available in official Fedora repositories and includes the already built UEFI firmware image with SB support.

1. Create a new directory for the new VM.

```
mkdir alpaquita-sb-vm
cd alpaquita-sb-vm
```

2. Unpack edk2-ovmf rpm to edk2-ovmf directory and copy two ovmf files from there.

```
mkdir edk2-ovmf
rpm2cpio edk2-ovmf-20210527gite1999b264f1f-3.fc36.noarch.rpm | cpio -i
--make-directories
cd ..
cp edk2-ovmf/usr/share/edk2/ovmf/OVMF_CODE.secboot.fd OVMF_CODE.fd
cp edk2-ovmf/usr/share/edk2/ovmf/OVMF_VARS.secboot.fd OVMF_VARS.fd
```

3. Create a QEMU disk image.

```
qemu-img create -f qcow2 -o preallocation=off alpaqa.qcow2 8G
```

4. Download official Alpaquita Linux ISO from the following location

<https://bell-sw.com/pages/downloads/alpaquita/>

5. Start a VM with the following options.

```
qemu-system-x86_64 -cpu host -enable-kvm -m 512 -smp 2 \
  -hda alpaquita.qcow2 \
  -machine q35,smm=on \
  -device e1000,netdev=net0 -netdev user,id=net0,hostfwd=tcp::5555-:22 \
  -global driver=cfi.pflash01,property=secure,value=on \
  -drive if=pflash,format=raw,unit=0,file=OVMF_CODE.fd,readonly=on \
  -drive if=pflash,format=raw,unit=1,file=OVMF_VARS.fd \
  -drive id=cd0,if=none,format=raw,readonly,file=alpaquita-virt-22.0-
x86_64.iso \
  -device ide-cd,bus=ide.1,drive=cd0,bootindex=1
```

6. The `qemu` options also forward ssh port, so you can access the VM via ssh after the installation is complete.

```
ssh -p 5555 alpaquita@localhost
```

Setting up Alpaquita Linux

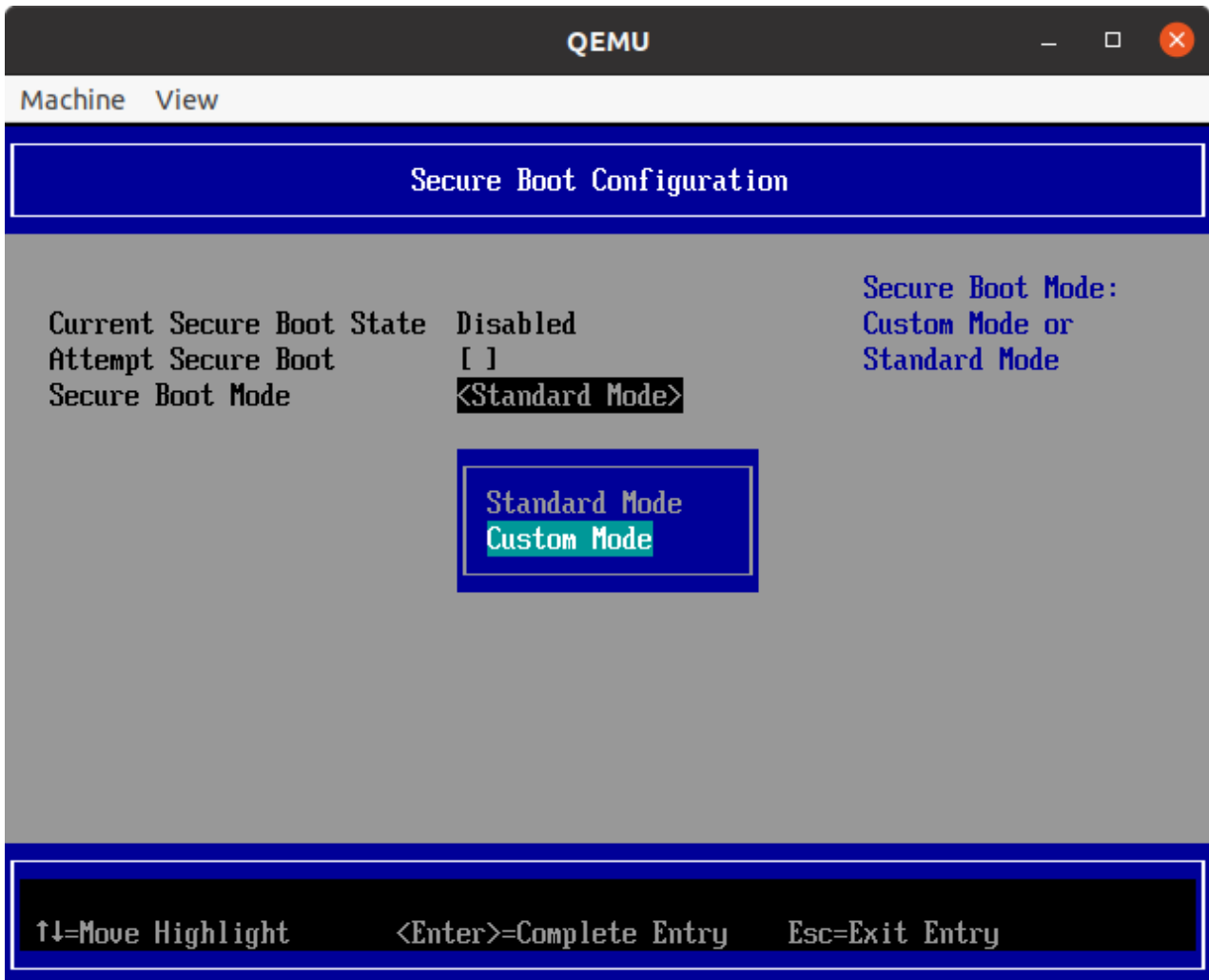
1. Boot the VM with Alpaquita Linux ISO. Alpaquita Linux ISO loads grub and starts the kernel. Now we can set up Alpaquita Linux.
2. On the "Secure Boot" screen of the Alpaquita Linux Installation, select "Install Secure Boot required packages to sign with own keys." Complete the setup and reboot.
3. Disable booting from ISO. You can remove the lines below from the options of the previous `qemu-system-x86_64` command:

```
-drive id=cd0,if=none,format=raw,readonly,file=alpaquita-virt-22.0-
x86_64.iso \
-device ide-cd,bus=ide.1,drive=cd0,bootindex=1 \
```

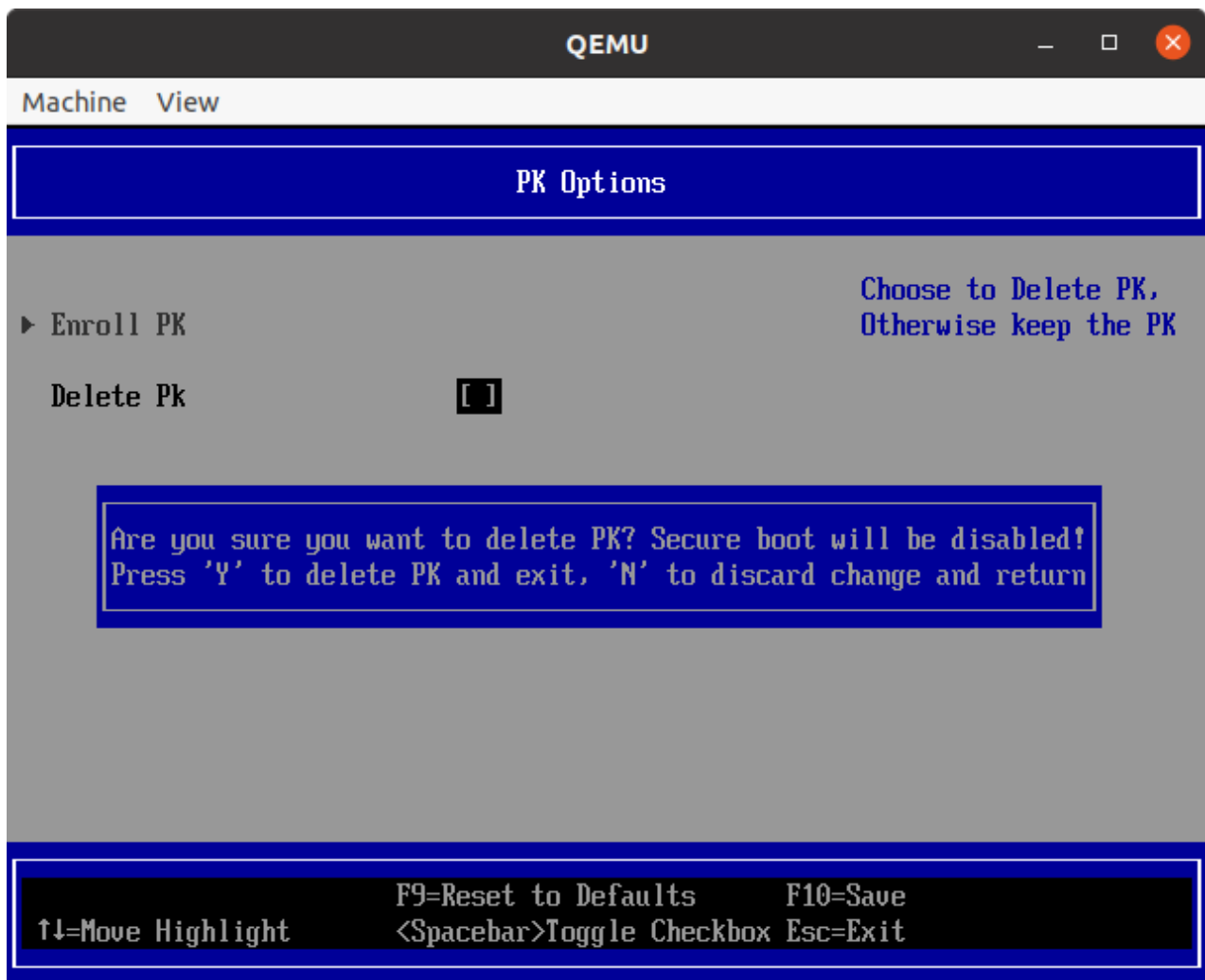
Switching Secure Boot to Setup mode

Reboot the machine, interrupt the normal booting process by pressing the ESC button. Navigate to

"Secure Boot Mode" and select "Custom Mode," then "Custom Secure Boot Options." Delete db, KEK, and PK keys.



Confirm the deletion of PK.



After deleting the PK key, Secure Boot turns on "Setup Mode."

Reboot the machine to complete this step.

Create PK, KEK, db keys

If you already have the required keys for signing shim and adding to the UEFI db, skip this step and go directly to the [Signing shim](#) part.

Installing required packages

Install the following packages.

```
sudo apk add openssl efitools sbsigntool uuidgen util-linux-misc
```

Creating the keys

Use the following script to create the keys.

```
#!/bin/sh

create_key()
{
    local name="$1"
    local subj="$2"

    openssl req -new -x509 -newkey rsa:2048 -sha256 -days 3650 \
        -keyout $name.key -subj "/CN=$subj" -out $name.pem
}

gen_keys()
{
    create_key PK "My Platform Key"
    create_key KEK "My Key Exchange Key"
    create_key SSK "My Shim Signing Key"

    local owner="$(uuidgen)"

    cert-to-efi-sig-list -g "$owner" PK.pem PK.esl
    cert-to-efi-sig-list -g "$owner" KEK.pem KEK.esl
    cert-to-efi-sig-list -g "$owner" SSK.pem SSK.esl

    cp SSK.esl db.esl

    # Uncomment to add Microsoft certificates to db
    # Dual boot, for Windows:
    #wget https://www.microsoft.com/pkiops/certs/MicWinProPCA2011_2011-10-19.crt
    # Third-party binaries signed by Microsoft (boot other Linux distros):
    #wget https://www.microsoft.com/pkiops/certs/MicCorUEFCA2011_2011-06-27.crt
    #openssl x509 -in MicWinProPCA2011_2011-10-19.crt -inform DER -out MicWin.pem
-outform PEM
    #openssl x509 -in MicCorUEFCA2011_2011-06-27.crt -inform DER -out MicCor.pem
-outform PEM
    #local mic="77fa9abd-0359-4d32-bd60-28f4e78f784b"
    #cert-to-efi-sig-list -g "$mic" MicCor.pem MicCor.esl
    #cert-to-efi-sig-list -g "$mic" MicWin.pem MicWin.esl
```

```
#cat SSK.esl MicWin.esl MicCor.esl > db.esl

sign-efi-sig-list -k PK.key -c PK.pem PK PK.esl PK.auth
sign-efi-sig-list -k PK.key -c PK.pem KEK KEK.esl KEK.auth
sign-efi-sig-list -k KEK.key -c KEK.pem db db.esl db.auth
}

gen_keys
```

After completion, the script should produce the following keys: SSK.key, SSK.pem, PK.auth, KEK.auth, db.auth. Note that "SSK" key is used to sign shim; and it is included in db.auth.

 **Important:**

Keep the private keys (*.key) secure and store them on an encrypted external device.

Signing shim

Perform the following commands with the keys to sign shim.

```
sudo su
sbsign --key SSK.key --cert SSK.pem /boot/efi/EFI/alpaquita/shimx64.efi
cd /boot/efi/EFI/alpaquita/
mv shimx64.efi.signed shimx64.efi
cp shimx64.efi ../BOOT/BOOTX64.EFI
exit
```

Installing keys to Secure Boot database

You can now Install new keys to UEFI Firmware using `KeyTool.efi` from the `efitools` package or `efi-updatevar` as follows.

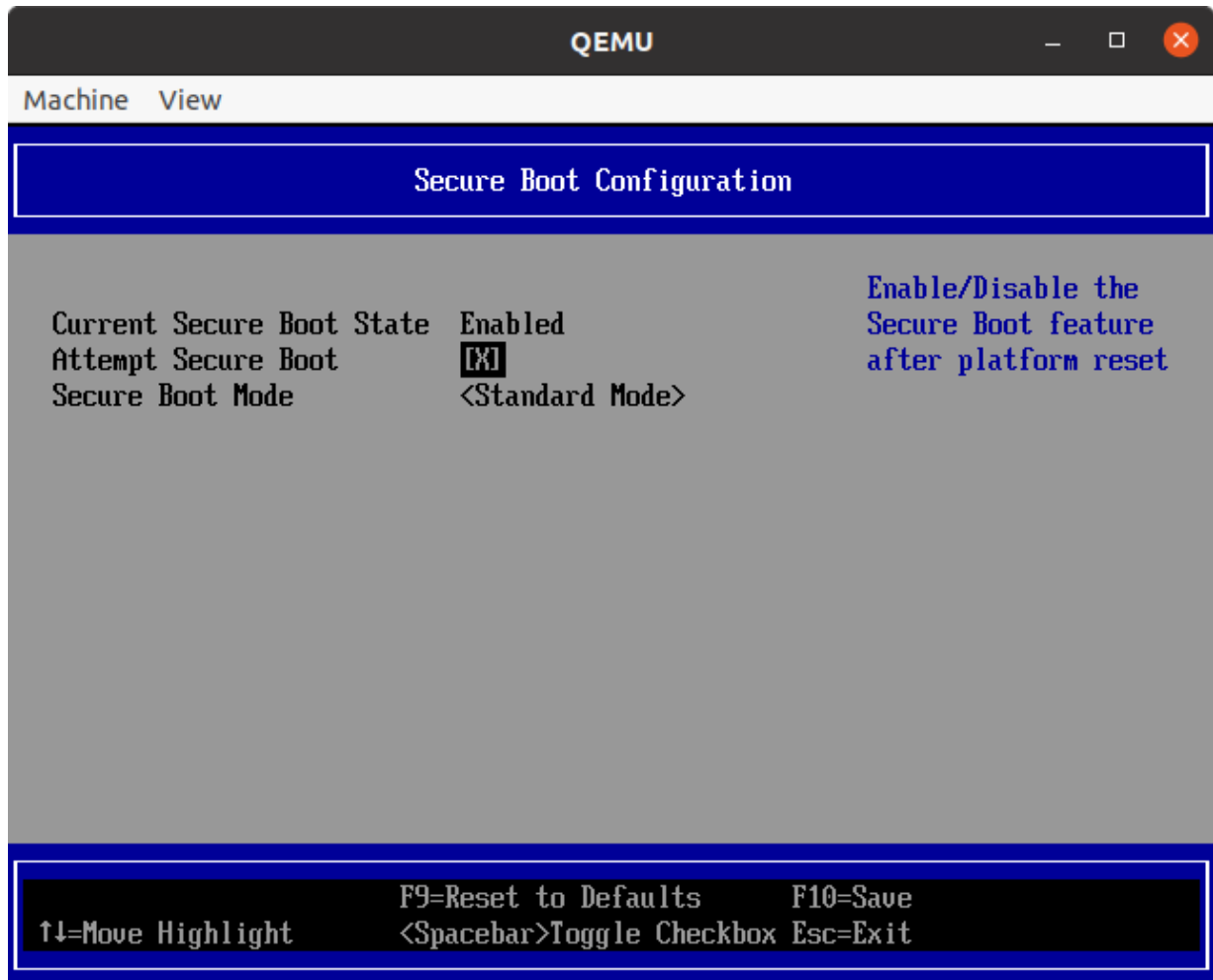
1. We recommend backing up UEFI variables before changing them.

```
efi-readvar -v PK -o PK_old.auth
efi-readvar -v KEK -o KEK_old.auth
efi-readvar -v db -o db_old.auth
```

2. Update with new ones.

```
sudo efi-updatevar -f db.auth db
sudo efi-updatevar -f KEK.auth KEK
sudo efi-updatevar -f PK.auth PK
```

3. Reboot the machine and enable Secure Boot in the UEFI firmware.



At this step, the Secure Boot is activated and Alpaquita Linux boots in lockdown mode.

Verifying Secure Boot

Enter the following command to verify that Secure Boot is enabled.

```
$ mokutil --sb-state
SecureBoot enabled
```

`dmesg` should contain the following lines indicating that secure boot is enabled and the kernel is locked

down.

```
$ sudo dmesg | grep Secure
[ 0.000000] Kernel is locked down from EFI Secure Boot; see man
kernel_lockdown.7
[ 0.000000] secureboot: Secure boot enabled
[ 0.207058] integrity: Loaded X.509 cert 'BellSoft: Alpaquita Secure Boot
CA: 3a793ac1906e33e9e12f28ca97c0f5b9652eae24'
```



Alpaquita Linux
Using Custom Keys in
Secure Boot

be//soft