Hard disk/SSD encryption is a strong defense but is not enough. You must secure your device against hardware attacks.

An **evil maid attack** is an attack on an unattended device, in which an attacker with physical access alters it in some undetectable way so that they can later access the device, or the data on it.

I will show you the way to enable secure boot in Arch linux installation (UEFI mode). I use a Thinkpad X230. This guide is working on Thinkpad laptops.

Pre-installation. Clear keys and enable UEFI mode in bios. Install arch linux/full disk encryption. Follow arch linux guide. Reboot your device. Open terminal and type

```
$ sudo -s
```

Install efitools

```
pacman -S efitools
```

Create a GUID for owner identification

```
uuidgen --random > GUID.txt
```

Platform key

```
openssl req -newkey rsa:4096 -nodes -keyout PK.key -new -x509 -sha256 -days 3650 -subj
"/CN=my Platform Key/" -out PK.crt
openssl x509 -outform DER -in PK.crt -out PK.cer
cert-to-efi-sig-list -g "$(< GUID.txt)" PK.crt PK.esl
sign-efi-sig-list -g "$(< GUID.txt)" -k PK.key -c PK.crt PK PK.esl PK.auth
```

Sign an empty file to allow removing PK when in "User Mode"

```
sign-efi-sig-list -g "$(< GUID.txt)" -c PK.crt -k PK.key PK /dev/null rm_PK.auth
```

Key Exchange Key

```
openssl req -newkey rsa:4096 -nodes -keyout KEK.key -new -x509 -sha256 -days 3650 -subj "/
CN=my Key Exchange Key/" -out KEK.crt
openssl x509 -outform DER -in KEK.crt -out KEK.cer
cert-to-efi-sig-list -g "$(< GUID.txt)" KEK.crt KEK.esl
sign-efi-sig-list -g "$(< GUID.txt)" -k PK.key -c PK.crt KEK KEK.esl KEK.auth
```

Signature Database key

```
openssl req -newkey rsa:4096 -nodes -keyout db.key -new -x509 -sha256 -days 3650 -subj
"/CN=my Signature Database key/" -out db.crt
openssl x509 -outform DER -in db.crt -out db.cer
cert-to-efi-sig-list -g "$(< GUID.txt)" db.crt db.esl
sign-efi-sig-list -g "$(< GUID.txt)" -k KEK.key -c KEK.crt db db.esl db.auth
```

Signing bootloader and kernel

Install sbsigntools

```
pacman -S sbsigntools
```

```
sbsign --key db.key --cert db.crt --output /boot/vmlinuz-linux /boot/vmlinuz-linux
sbsign --key db.key --cert db.crt --output /efi/EFI/arch/grubx64.efi
/efi/EFI/arch/grubx64.efi
```

Automatically sign bootloader and kernel on install and updates

Create the hooks directory

```
mkdir -p /etc/pacman.d/hooks
```

Create hooks for both the linux and grub packages

/etc/pacman.d/hooks/99-secureboot-linux.hook

```
[Trigger]
Operation = Install
Operation = Upgrade
Type = Package
Target = linux

[Action]
Description = Signing Kernel for SecureBoot
When = PostTransaction
Exec = /usr/bin/find /boot/ -maxdepth 1 -name 'vmlinuz-*' -exec /usr/bin/sh -c 'if ! /usr/
bin/sbverify --list {} 2>/dev/null | /usr/bin/grep -q "signature certificates"; then /usr/
bin/sbsign --key /root/db.key --cert /root/db.crt --output {} {}; fi' \ ;
Depends = sbsigntools
Depends = findutils
Depends = grep
```

/etc/pacman.d/hooks/98-secureboot-grub.hook

```
[Trigger]
Operation = Install
Operation = Upgrade
Type = Package
Target = grub

[Action]
Description = Signing GRUB for SecureBoot
When = PostTransaction
Exec = /usr/bin/find /efi/ -name 'grubx64*' -exec /usr/bin/sh -c 'if ! /usr/bin/sbverify
--list {} 2>/dev/null | /usr/bin/grep -q "signature certificates"; then /usr/bin/sbsign --
key /root/db.key --cert /root/db.crt --output {} {}; fi' \ ;
Depends = sbsigntools
Depends = findutils
Depends = grep
```

Enroll keys in firmware

Copy all *.cer, *.esl, *.auth to the EFI system partition

```
cp /root/*.cer /root/*.esl /root/*.auth /efi/
```

Boot into UEFI firmware setup utility

```
systemctl reboot –firmware
```

You have already clear keys, make sure that is in **SETUP MODE.**

Open terminal and type

```
$ sudo -s
```

The EFI variables may be immutable (i-flag in lsattr output) in recent kernels

```
chattr -i /sys/firmware/efi/efivars/{PK,KEK,db,dbx}-*
```

Install keys into EFI . Make sure to follow db==>KEK==>PK

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

Reboot into firmware

Now you see that **setup mode** changed to **user mode.** Enable **Secure boot** and create a strong supervisor password.

Open terminal and type

```
$ od --address-radix=n --format=u1 /sys/firmware/efi/efivars/SecureBoot*
```

If Secure Boot is enabled, this command returns 1 as the final integer in a list of five, for example:

```
6  0  0  0  1
```

Now you are protected against evil maid attacks :)

* https://wiki.archlinux.org/title/Unified_Extensible_Firmware_Interface/Secure_Boot

* https://ruderich.org/simon/notes/secure-boot-with-grub-and-signed-linux-and-initrd

* https://www.rodsbooks.com/efi-bootloaders/secureboot.html