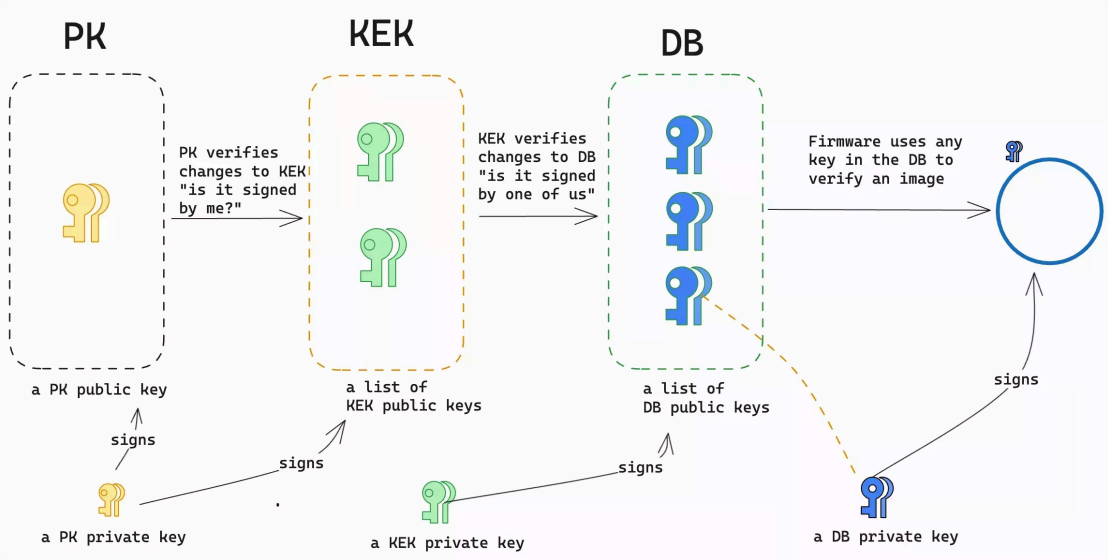
Trusted Boot Keys

All security features of Trusted Boot rely on cryptographic keys. Secure Boot relies on the chain of trust between Platform Key (PK), Key Exchange Keys (KEK), and Signature Database (DB) keys to achieve a tamper proof boot process. Full Disk Encryption also relies on a key pair. This section talks about the different keys used in Trusted Boot, their roles, and best practices to manage them securely.



Platform Key (PK)

The PK is at the top of the Secure Boot cryptographic key hierarchy. It's a key pair that has a private key and a public key.

The private PK signs updates to the Key Exchange Key (KEK) variable, which contains a list of certificates, public keys, or signatures in a EFI Signature List (ESL). The public PK is used to verify whether updates to the KEK are signed with the authentic private key and can be trusted. It establishes a relationship of trust between the platform owner and the platform firmware. A system can only have one PK.

Key Exchange Key (KEK)

The KEK is a set of certificates, public keys, or signatures in an ESL signed by a private PK key. Entries in the KEK set are used to update the Signature Database (DB) and the Forbidden Signature Database (DBX). Each entry in

the KEK set has a corresponding private key. It establishes a relationship of trust between the firmware and the operating system (OS).

Any private key corresponding to an entry in the KEK set can sign updates to the DB. When there are updates to the DB, the corresponding public key, certificate, or signature in the KEK set is used to verify that those updates are signed by the authentic private key.

Signature Database (DB) and Forbidden Signature Database (DBX)

The DB validates signed EFI (Extensible Firmware Interface) binaries. The DB may contain a mixed set of certificates, public keys, signatures or hashes of binary files. The signature stored in the EFI binary (or a hash of the binary if there is no signature) is compared against the entries in the database. The binary will be executed if one of the following conditions is met:

- The EFI binary is signed, and the signing key is in the DB.
- The EFI binary is signed, and the signature on the binary is in the DB.
- The EFI binary is unsigned and a SHA-256 hash of the image is in the DB.

Similarly, DBX may contain a mixed set of certificates, signatures or hashes. Any EFI whose signatures, hashes, or signing key matches the entries in DBX is forbidden from being executed.

Platform Configuration Registers (PCR) Policy Key

The PCR policy key pair is in charge of signing the pre-calculated measurements of the boot process and involved in disk encryption. The private PCR policy key signs the pre-calculated measurement during EdgeForge. The public key is embedded in the UKI image.

During EdgeForge, each boot component is hashed and these hash values, or measurements, are signed by the PCR private key. The signed measurements are embedded in the UKI image, along with the public key of the PCR policy key pair. For more information about EdgeForge, refer to EdgeForge with Trusted Boot.

During installation, encrypted partitions are setup using a Disk Encryption Key (DEK), which is itself encrypted by the TPM and stored in a secure blob. The PCR public key embedded in the ISO is used to form a binding policy. The binding policy states that in order to decrypt the secure blob containing the DEK, the PCR measurements must match a pre-calculated set of measurements signed by the corresponding PCR private key.

During the boot process, before the encrypted disk partition is mounted, the TPM will perform the following:

- Verify the public key in the image is valid
- Verify the signature on the pre-calculated measurements using the public key
- Compare the pre-calculated measurements vs the actual PCR measurements

If all three verifications are successful, TPM will decrypt the secure blob, release DEK, and the OS can use it to decrypt the encrypted partitions of the disk.

Disk Encryption Key (DEK)

The disk encryption key (DEK) is generated during installation, encrypted by the TPM with an internal key, and sealed inside the TPM. You will never interact with the DEK itself.

During the boot process, the TPM will perform a series of verifications. If all of them are successful, the TPM will decrypt and release the DEK to the OS, so it can use it to decrypt the encrypted partitions. Refer to the PCR Policy Key section for details.

Factory Keys

Factory keys refer to the secure boot keys that are stored on the device in factory settings. In EdgeForge, these keys are stored in the folder **exported-keys**. They may include PK, KEK, and DB keys. Factory keys are often used to authenticate the firmware of a device. For more information, refer to Export Factory Keys.

Resources

- Export Factory Keys
- Generate Keys
- Key Management

Tags: edge

*Last updated on **Apr 28, 2025***

Generate Keys for Trusted Boot

Trusted Boot works by signing the Edge Installer image and provider images with cryptographic keys, and only allowing images signed with the trusted keys to operate during the Edge host boot process. This page guides you through the process of generating keys to be used when building Edge artifacts.

The key generation process produces three pairs of keys and a Platform Configuration Register (PCR) policy private key, and each pair of keys fulfills different purposes. The following table provides a brief overview of which keys are used in which Trusted Boot EdgeForge and deployment process. For more information, refer to EdgeForge with Trusted Boot.

Keys	Key Generation	Build Installer ISO	Building Provider Images	Installation
PK & KEK (private)	✅	Not needed	Not needed	Not needed
PK & KEK (public)		✅	Not needed	✅
DB (public)		✅	✅	Not needed
DB (private)		✅	✅	Not needed
PCR policy Key (private)		✅	Not needed	✅

> ⚠ WARNING
>
> All security provided by Trusted Boot assumes that the private keys are kept secure. We recommend that you perform key generation in an air-gapped environment and move the PK and KEK private

Feedback

keys to a secure location immediately after generating them. In addition, any build pipelines that are created for the purposes of building ISOs and provider images must be secured, as they must contain the PCR and DB private keys.

Prerequisites

- A physical or virtual Linux machine with *AMD64* (also known as x86_64) processor architecture to build the Edge artifacts. You can issue the following command in the terminal to check your processor architecture.

```
uname -m
```

- Minimum hardware configuration of the Linux machine:
  - 4 CPU
  - 8 GB memory
  - 50 GB storage
- You have exported the factory keys from the Edge device. For more information, refer to Export Factory Keys.
- Git. You can ensure git installation by issuing the git --version command.
- openssl must be installed on your Linux machine.

Instructions

Generate Keys Using Self-Signed Certificates | Generate Keys Using an Existing CA

If your environment does not require a Certificate Authority (CA), you can use self-signed certificates to generate the keys needed for Trusted Boot. Using self-signed certificates may make verifying the source of the certificate more difficult because there is no higher authority.

1. Clone the **CanvOS** repository.

```
git clone https://github.com/spectrocloud/CanvOS.git
```

2. Change to the **CanvOS/** directory.

```
cd CanvOS
```

3. View the available git tag.

```
git tag
```

4. Check out the latest available tag that is v4.4.0 or later. This guide uses the tag v4.4.0 as an example.

```
git checkout v4.4.0
```

5. Issue the following command to create the folders for Trusted Boot keys.

```
./earthly.sh +secure-boot-dirs
```

   This will create a folder named **secure-boot** and three subdirectories: **exported-keys**, **private-keys** and **public-keys**.

6. Copy the keys you exported from your Edge device in Export Factory Keys to the **exported-keys** directory.

> DANGER
>
> If this is not the first time you have generated keys, make sure that **secure-boot** folder has no existing keys except for the exported keys you just copied before proceeding to the next step. Issuing the key generation command will overwrite all existing keys silently. Ensure that you have backed up all your existing keys before generating new ones.

7. Issue the following command to generate keys. Replace `org-name` with the name of your organization, and replace 5475, the default expiration period in days, with the desired expiration period for your keys.

   > DANGER
   >
   > Specify a distant expiration date. If the keys expire before you can replace them, it can soft-brick the Edge host. Although the default is 15 years, you may choose to make this longer.

   ```
   ./earthly.sh +uki-genkey --MY_ORG="org-name" --
   EXPIRATION_IN_DAYS=5475
   ```

   All keys are generated to the **secure-boot** folder. Private keys are kept in a subdirectory called **private-keys**. Public keys are generated to a subdirectory called **public-keys**.

   The key generation script also produces a folder named **enrollment**. This folder contains public keys that will be built into the Edge installer ISO, and eventually enrolled in your Edge device when you install Palette Edge with the ISO.

8. Remove **PK.key** and **KEK.key** from the **private-keys** folder and keep them offline in a safe location.

Validate

Check the content of the **secure-boot/enrollment** directory. You should observe the following nine files.

```
$ ls secure-boot/enrollment/
KEK.auth  KEK.der  KEK.esl  PK.auth  PK.der  PK.esl  db.auth  db.der
db.esl
```

Tags: edge

*Last updated on **Apr 28, 2025***

Trusted Boot Key Management

Several key pairs are used in Trusted Boot during installer ISO generation, upgrade image generation, as well as installation. Each key pair serves a different purpose and is used during different stages of Edge artifact building and deployment. Each key pair needs to be secured differently. This page discusses the different key pairs used by Trusted Boot and how to secure them.

Careful key management is the foundation of all security benefits provided by Trusted Boot. All security provided by Trusted Boot assumes that your keys are handled and stored securely. Ensure that you follow our recommendations to avoid compromising the security of your systems.

Platform Key (PK)

The private PK must be stowed away in a secure location *immediately* after being generated. You do not need the PK private key during EdgeForge operations, installation, upgrades or deployments of your Edge hosts. The public PK key is required during the EdgeForge build process so that it can be embedded into the Edge Installer ISO and thereafter installed on Edge hosts. For more information, refer to EdgeForge with Trusted Boot.

> DANGER
>
> Ensure that the private PK is kept securely with strictly limited access. Someone in possession of the private PK key can make changes to the KEK and gain access to your devices and their data.

The following files are all part of the PK key.

Filename	Description	Key Management Recommendation
PK.pem	The public PK key in Privacy Enhanced Mail (PEM) format.	Store in the build pipeline for EdgeForge.

Filename	Description	Key Management Recommendation
PK.key	The private PK key.	Store offline in a secure location.
PK.esl	The EFI Signature List for the PK key.	Store in the build pipeline for EdgeForge.
PK.der	The public PK key in DER (Distinguished Encoding Rules) format, a binary form of the PEM file.	Store in the build pipeline for EdgeForge.
PK.auth	This file contains signed data used for updating the Secure Boot variables in the Unified Extensible Firmware Interface (UEFI) firmware.	Store in the build pipeline for EdgeForge.

Key Exchange Key (KEK)

The private KEK must be stowed away in a secure location **immediately** after being generated. You do not need the KEK private key during EdgeForge operations, installation, upgrades or deployments of your Edge hosts. The public KEK is required during the EdgeForge build process so that it can be embedded into the Edge Installer ISO and thereafter installed on Edge hosts.

Filename	Description	Key Management Recommendation
KEK.pem	The public KEK key in Privacy Enhanced Mail (PEM) format.	Store in the build pipeline for EdgeForge.
KEK.key	The private KEK key.	Store offline in a secure location.
KEK.esl	The EFI Signature List for the KEK key.	Store in the build pipeline for EdgeForge.

Filename	Description	Key Management Recommendation
KEK.der	The public KEK key in DER (Distinguished Encoding Rules) format, a binary form of the PEM file.	Store in the build pipeline for EdgeForge.
KEK.auth	This file contains signed data used for updating the Secure Boot variables in the UEFI firmware.	Store in the build pipeline for EdgeForge.

Signature Database (DB) and Forbidden Signature Database (DBX)

Both the public and private DB keys should be stored securely in the build pipeline of your Edge artifacts, as they are needed during EdgeForge both during initial deployment and upgrades. The build pipeline itself should be heavily secured with limited access. The DB private key must not be stored in repositories that are exposed publicly. Ideally, Edge host artifacts should be generated in an air-gapped environment to reduce potential exposure of the DB private key. If possible, the build pipeline should utilize an Hardware Security Module (HSM).

Filename	Description	Key Management Recommendation
db.pem	The public DB key in Privacy Enhanced Mail (PEM) format.	Store in the build pipeline for EdgeForge.
db.key	The private DB key.	Store in the build pipeline for EdgeForge
db.esl	The EFI Signature List for the DB key.	Store in the build pipeline for EdgeForge.
db.der	The public DB key in DER (Distinguished Encoding Rules) format, a binary form of the PEM file.	Store in the build pipeline for EdgeForge.

Filename	Description	Key Management Recommendation
db.auth	This file contains signed data used for updating the Secure Boot variables in the UEFI firmware.	Store in the build pipeline for EdgeForge.

Platform Configuration Registers (PCR) Policy Key

The PCR policy key pair is in charge of signing the pre-calculated measurements of the boot process and involved in disk encryption. The private PCR policy key needs to be stored securely in the build pipeline so it can sign the pre-calculated measurement during EdgeForge. The public key is generated and embedded in the UKI image automatically, and you do not need to handle the public key.

Filename	Description	Key Management Recommendation
tpm2-pcr-private.pem	The private PCR policy key.	Store in the build pipeline for EdgeForge.

Tags: edge

*Last updated on **Jun 16, 2024***