AMD XILINX

Key Revocation Lab

XAPP1344 (v1.1) March 14, 2022

Summary

This application note describes the programming of security related eFUSEs in Zynq® UltraScale+™ MPSoCs to set up the secure boot for a ZCU102 board. It then demonstrates how to program eFUSEs for revoking public keys of applications and partitions running on a ZCU102 board.

The reference design files for this application note can be downloaded from the Xilinx website. For detailed information about the design files, see Reference Design.

Introduction

Secure boot ensures the system only runs intended firmware and is accomplished by using the hardware root of trust boot mechanism. This provides the required confidentiality, integrity, and authentication to host the most secure applications. The Zynq UltraScale+ MPSoC hardware root of trust is based on the Rivest-Shamir-Adleman (RSA)-4096 asymmetric authentication algorithm with the Secure Hash Algorithm (SHA)-3/384. The use and revocation of primary public keys and secondary public keys is demonstrated.

The secure boot process starts by determining which Primary Public Key (PPK) to use and then validating its integrity. The public key is stored in the boot image (BI) in external memory, therefore, it is assumed that an adversary could tamper with it. Consequently, the configuration security unit (CSU) reads the public key from external memory, calculates its cryptographic checksum using the SHA-3/384 engine, and compares it to the value stored in eFUSEs. If they match, the integrity of the public key has been validated and the boot can continue. The Secondary Public Key (SPK) and its associated ID are then read, stored in on-chip memory (OCM), and authenticated using the PPK. After the SPK and SPK ID have been authenticated, the CSU compares the ID that was bound to the SPK in the BI to the ID that is stored in the eFUSEs. If the IDs match, the SPK is valid and the boot continues. Lastly, the SPK verifies the authenticity of the entire BI. The CSU authenticates the first stage boot loader (FSBL), and optionally the PMU firmware (PMUFW) if enabled in the BI. If encrypted, the CSU also performs the decryption after authentication.

Refer to *Zynq UltraScale+ Device Technical Reference Manual* (UG1085) to better understand different boot modes and features available for secure, encrypted, and normal boot.

The Zynq UltraScale+ MPSoC can store the hash digest of both PPKs. Each PPK can only be revoked once (i.e., revoke the first PPK and use the second PPK). Since only two revocations is not sufficient in typical systems, the Zynq UltraScale+ MPSoC provides a secondary key mechanism that:

- Provides a second level of defense if the first authentication mechanism gets compromised.
- Allows the user to revoke SPK more than twice.
- Uses different keys to authenticate each application or group of partitions, enhancing the security posture of the end system.

Each SPK is associated with an ID called SPK_ID. The Zynq UltraScale+ MPSoC provides a 32-bit eFUSE register called SPK_ID to hold the SPK_ID associated with SPK, so the user can revoke the SPK a maximum of 32 times. In this document, this revocation method is referred to as Zynq UltraScale+ standard key revocation.

In addition, the Zynq UltraScale+ MPSoC also provides 256 user eFUSEs. These eFUSEs can be used optionally to indicate the revocation status of the SPK associated with SPK_IDs 0-256. With this, the user can revoke up to 256 SPKs. In this document, this revocation method is referred to as Zynq UltraScale+ enhanced key revocation.

The following table lists the key differences between standard and enhanced revocation modes.

Table 1: **Zynq UltraScale+ Key Revocation Modes**

Zynq UltraScale+ Standard Key Revocation	Zynq UltraScale+ Enhanced Key Revocation
Uses SPK ID eFUSEs	Uses user eFUSEs
32 Reserved eFUSEs	256 user-assigned eFUSEs ***Note***: Not all user eFUSEs are required to be used.
FSBL key must be revoked using the standard revocation format.	FSBL cannot be revoked using enhanced revocation format.
Non-FSBL partitions can be revoked using standard revocation format.	Non-FSBL partitions can be revoked using enhanced revocation format.
Only one standard SPK ID number is valid at a time.	Many enhanced SPK ID numbers can be valid at a time (up to 256).
Changing the SPKID eFUSEs impacts all partitions regardless of SPK ID number.	Changing the USER eFUSEs only impacts partitions using that specific SPK ID number.

Hardware and Software Requirements

The following hardware and software are required for this application note:

Note: This lab was developed on Windows 10. If using Linux the file paths will need to be updated with your system's path details.

- ZCU102 evaluation board
- AC to DC power adapter (12 VDC)
- USB type-A to micro-B USB cable for UART communication

- Secure Digital (SD) card ≤ 32 GB
- SD formatted using the FAT file system
- Vitis Unified Software Platform (2021.1 or later)

  ***Note*:** Versions other than 2021.1 have not been verified.
- Serial communications terminal application (i.e., Tera Term or PuTTY)
- Required design files, which can be downloaded from Reference Design

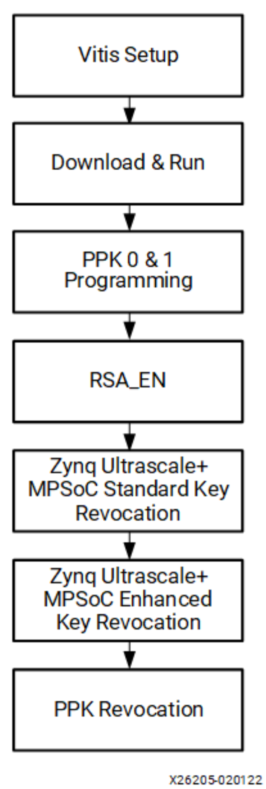
IMPORTANT! *Programming any of the noted eFUSE settings preclude Xilinx test access, therefore, Xilinx might not accept return material authorization (RMA) requests. Additionally, programming eFUSEs limits the usage of the board, because after provisioning the board for secure boot, only authenticated boot images can boot.*

***Note*:** For the simplicity of this application note, you are advised to extract the contents of the required design files to `C:\Xilinx`.

Reference Design Flow

The following figure shows a summary of the reference design flow.

Figure 1: **Reference Design Flow**



Vitis Setup

- Create an FSBL for the Arm Cortex-A53 Core
- Modify BSP to Include XilSkey Library

Vitis Setup

Create an FSBL for the Arm Cortex-A53 Core

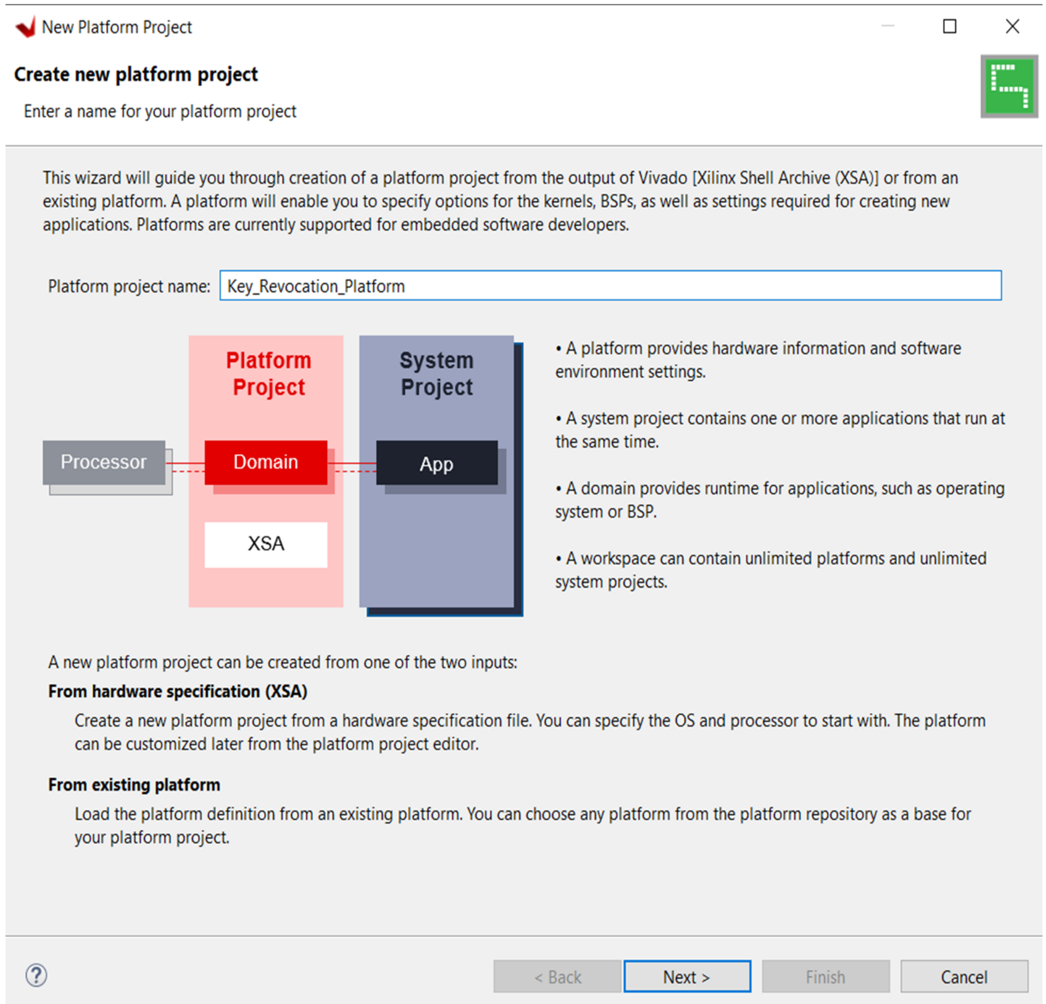
An FSBL for booting the lab application must be created within the Platform Project. The FSBL will be generated in a later section with the SD card boot mode. The FSBL loads on the Arm® Cortex®-A53 processor and subsequently, the FSBL loads the lab application on the Cortex-A53 core. In Vitis, the FSBL is created as part of the platform project.

1. Launch Vitis™.
2. Set the workspace path.

   Note: For this walk-through the workspace path is assumed to be `C:\Xilinx\Key_Revocation_Lab`.

3. Select **File > New > Platform Project**.
   - The Create a New Platform window opens.
   - Enter *Key_Revocation_Platform* for the platform project name as shown in Figure 2.

Figure 2: **Create a Platform Project**

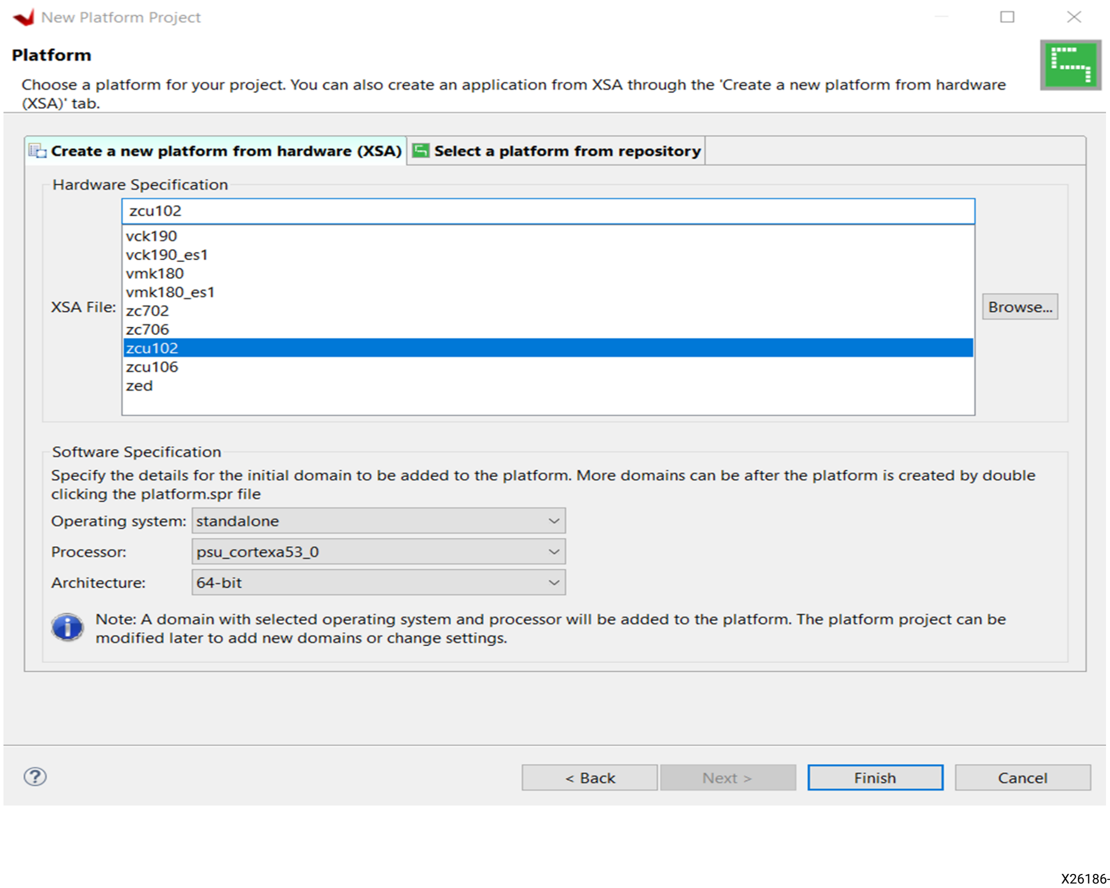


X26175-013122

4. Select **Next**.
5. Select the *Create a new platform from hardware (XSA)* tab as shown in the following figure.
6. Select the prebuilt zcu102 xsa file from the Hardware Specification drop-down menu.

   Note: If your system needs an xsa file that is not listed, these can be built within Vivado (see *Zynq UltraScale+ MPSoC: Embedded Design Tutorial* (UG1209)).
7. Make the following selections within the Software Specification section of the window:
   a. Operating system: standalone
   b. Processor: psu_cortexa53_0
   c. Architecture: 64-bit

Figure 3: **Select XSA File and Setup the Platform**



8. Select **Finish**.

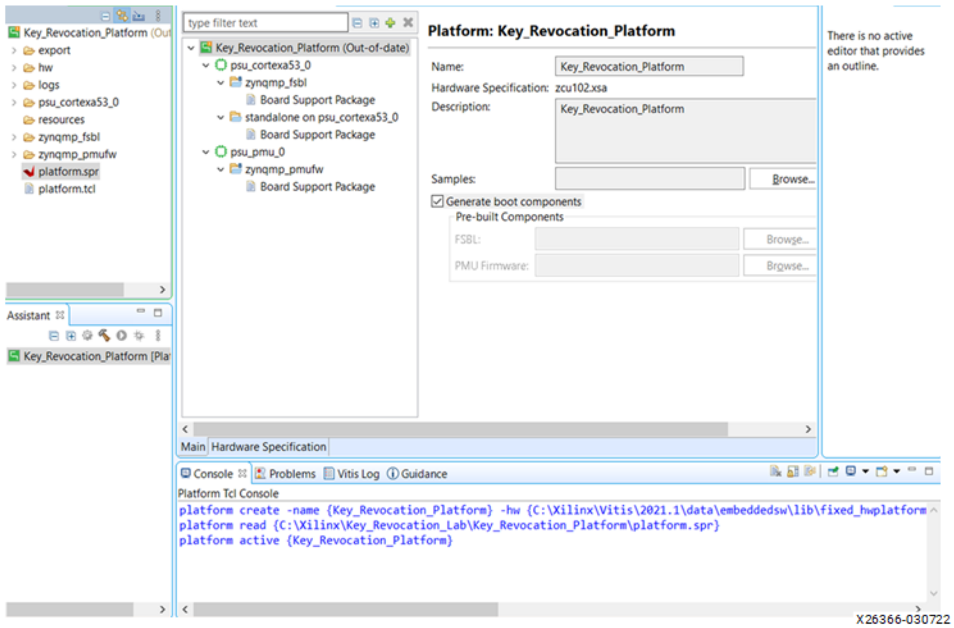
Vitis creates the board support package (BSP) and an FSBL application. It might take a moment for the Vitis to compile and create the FSBL and BSP.

***Note*:** In this example, the application name *fsbl_a53* is to identify that the FSBL is targeted for the application processing unit (APU) (the Arm® Cortex®-A53 Core).

Modify BSP to Include XilSkey Library

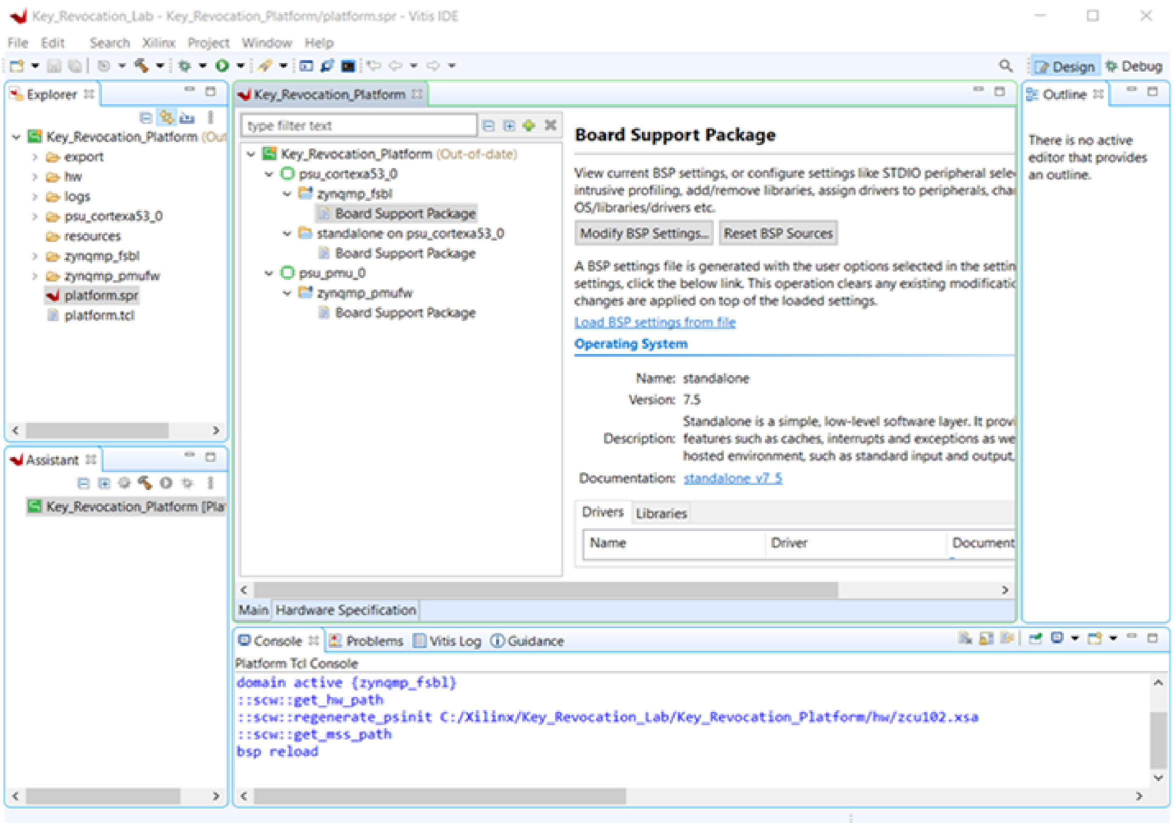
Programming eFUSEs requires the XilSkey library. Therefore, it is necessary to modify the BSP settings to include the XilSkey library.

Figure 4: **Platform Details**



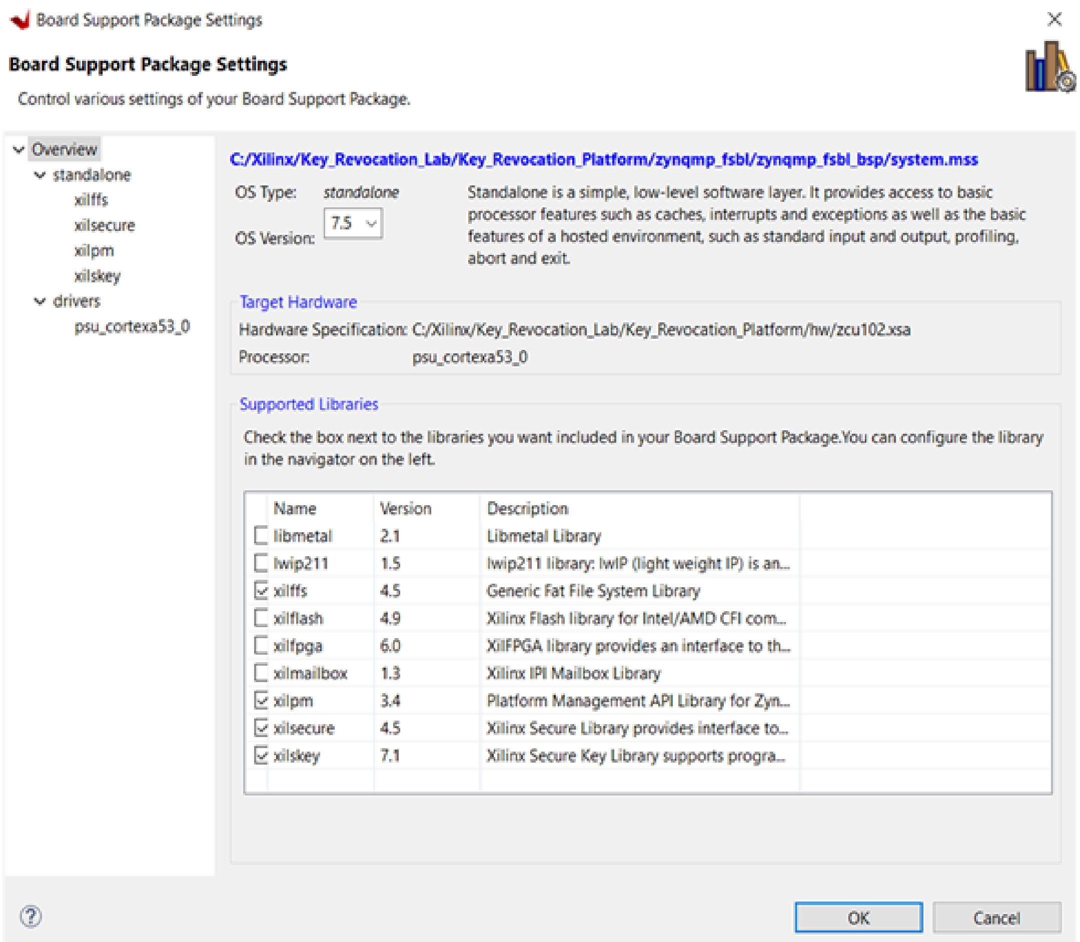
1. Select the *platform.spr* within the Explorer window as shown in the previous figure.
2. In the *Key_Revocation_Platform* tab, under the *zynqmp_fsbl* drop-down menu, select the *Board Support Package*.
3. Select *Modify BSP Settings* as shown in the following figure.

Figure 5: **Board Support Package Modification**



4. Select the *xilskey* check box as shown in the following figure.

Figure 6: **Adding Xilskey to the BSP**



5. Select **OK**.

6. Repeat steps 2-5 in this section for the *Board Support Package*, under the *standalone on psu_cortexa53_0* drop-down menu - Select the *xilsecure* and *xilskey* libraries.

Note: In the BSP settings window, the version of standalone OS type and XilSkey might be different based on the Vitis version being used. This application note was developed using Vitis 2021.1.

Create a Lab Application for the Arm Cortex-A53 based APU

Now that the platform is created and the BSP has been modified to support this exercise, the next section shows how to create an empty bare-metal application targeted for an Arm Cortex-A53 Core 0. This application will be modified using source files that get compiled and run to exercise the Zynq UltraScale+ Key Revocation features.

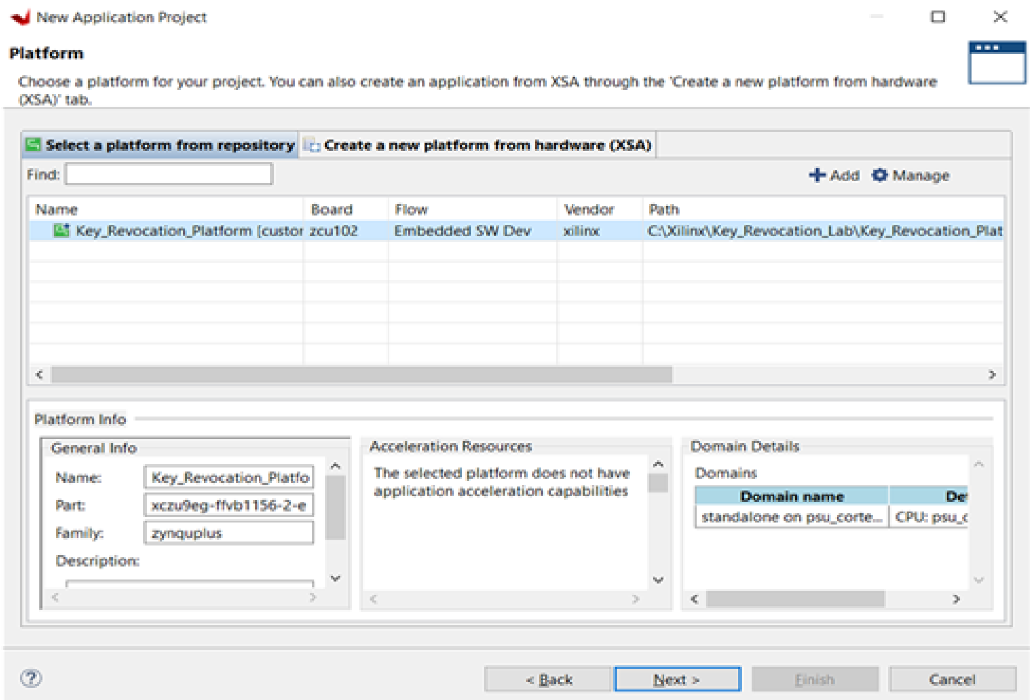
Note: This application is referred to as a lab application throughout the document.

1. Select **File→New→Application Project**.

   The New Project dialog box opens. Select **Next** if the welcome page opens.

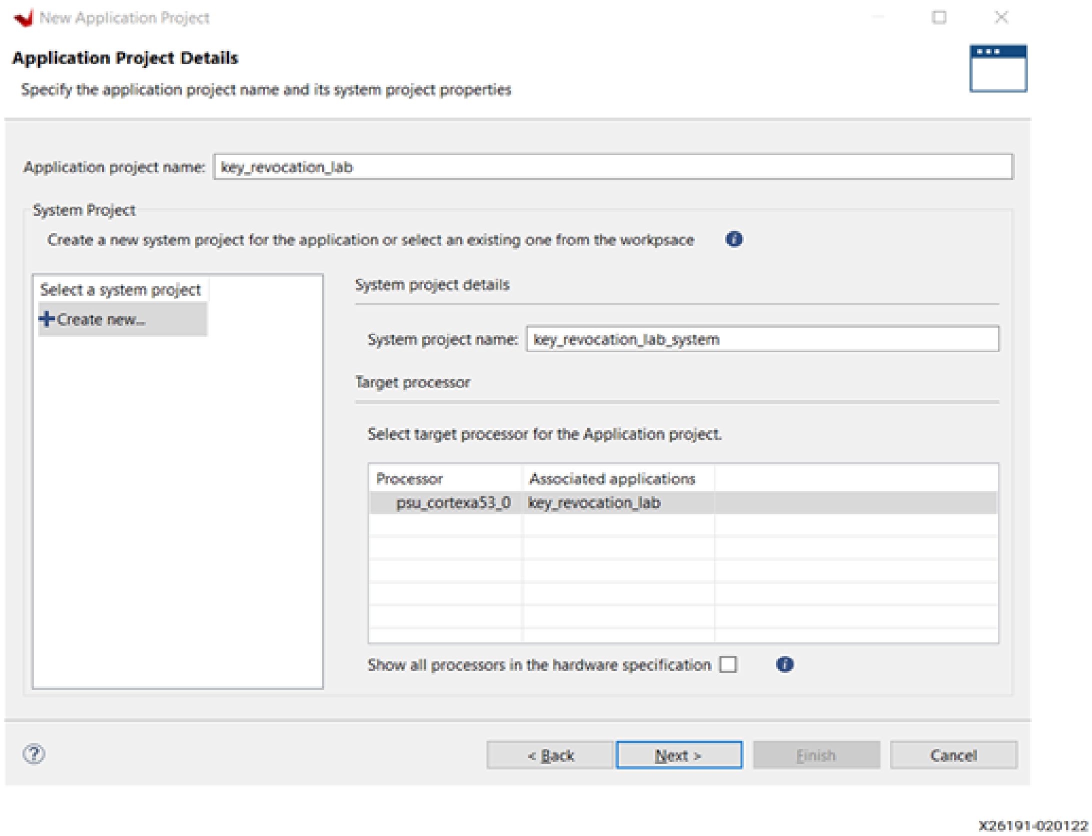
2. Select **Key_Revocation_Platform [custom]**, created earlier in the lab under the *Select a platform from repository* tab.

Figure 7: **Key_Revocation_Platform [custom]**



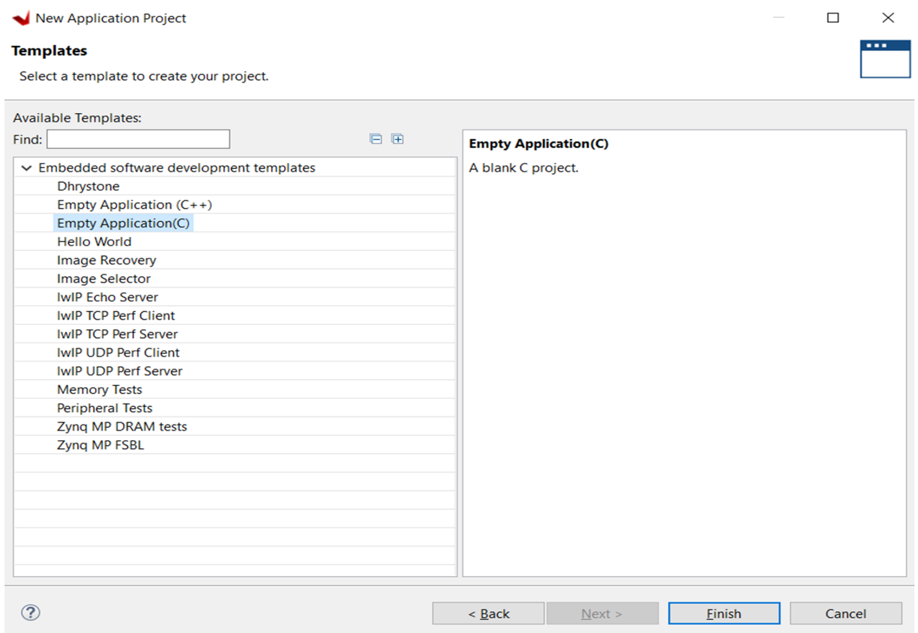
3. Select **Next**.
4. Enter `key_revocation_lab` in the *Application* project name field as shown in the following figure.

Figure 8: **Application Project Details**



5. Select **Next**.
6. Select **Next** on the *Domain* page and the *Template* window will open.
7. In the *Template* window, select *Empty Application* as shown in the following figure.

Figure 9: **Template Selection**



X26193-020122

8. Select **Finish**.

   Vitis creates the empty application named "`key_revocation_lab`". After a bare-metal application is generated, the following C source files must be imported to create the lab application for eFUSE programming:

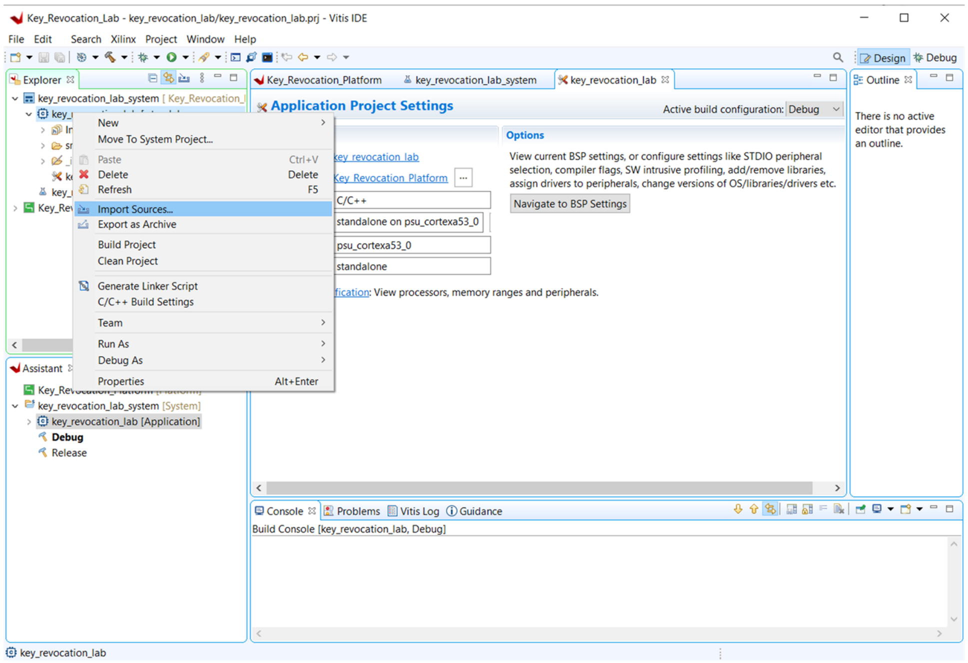
   - `key_revocation_lab_main.c`
   - `key_revocation_lab_utils.c`
   - `key_revocation_lab_main.h`
   - `key_revocation_lab_utils.h`

   These files can be found in C:\Xilinx\Key_Revocation_Lab \enhanced_key_revocation_lab_files.

   Note: Files should be extracted in C:\Xilinx. If they are extracted elsewhere, the extracted files can be found in that location.

9. Right-click the *key_revocation_lab* under the *key_revocation_lab_system* drop-down menu in the Explorer window as shown in the following figure.
10. Select *Import Sources*.

Figure 10: **Source Code Import Menu Location**



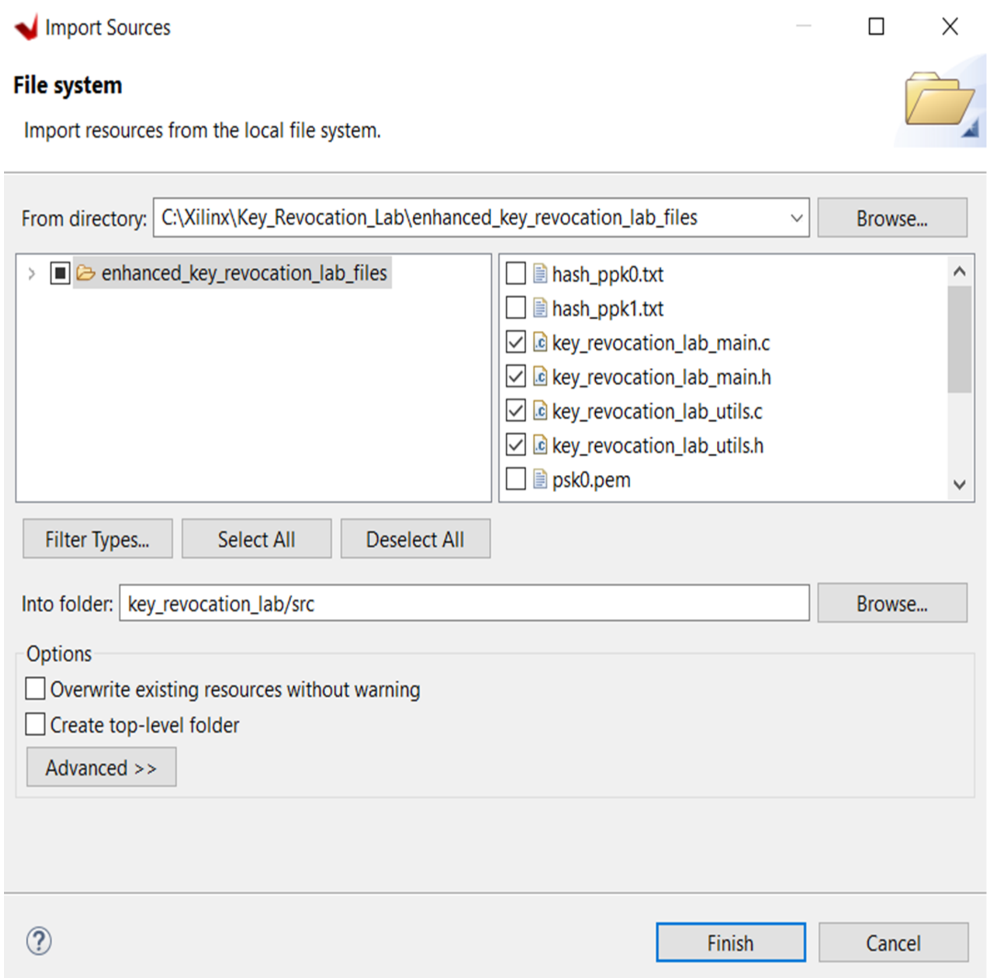
X26202-020122

11. Browse to C:\Xilinx\Key_Revocation_Lab\enhanced_key_revocation_lab_files.

12. Select the following files as shown in the following figure:

- `key_revocation_lab_main.c`
- `key_revocation_lab_main.h`
- `key_revocation_lab_utils.c`
- `key_revocation_lab_utils.h`

Figure 11: **Source Code Selection**



13. Select **Finish**.
14. Open *key_revocation_lab_main.h*.
15. On line 20, set the value of macro `WRITE_EFUSES` to TRUE.
16. Save the file.

    Note: The default value of the WRITE_EFUSES macro is FALSE. If the value of this macro is false, no eFUSE is programmed, however, you are still able to execute all the eFUSE programming steps listed in the later sections of this application note without modifying/programming them. You are encouraged to first have a basic understanding of the tutorial user interface (UI) by setting the value to FALSE (i.e., skip Step 6 above). This allows you to become familiar with the lab application UI without programming any of the eFUSEs, which is helpful because eFUSE programming is irreversible.

17. Right-click on *key_revocation_lab* project.
18. Select **Clean Project**.
19. Right-click on *key_revocation_lab* project.

20. Select **Build Project**. Vitis will build both the application and Platform. This process can take several minutes to complete.

Note: Ensure there are no build errors reported within the Console or Problems tabs.

With the lab application ready, the next step is to create a BI and load the application on the ZCU102 board.

Download and Run Lab Application

Generate Boot Image

The next step is to create a boot image (BI) to boot the FBSL and lab application generated in previous sections via SD card boot mode.

A BI is generated using a Boot Image Format (BIF) file and the Bootgen tool (see UG1283).

Note: BI uses the FSBL and lab application ELF files generated in previous sections.

1. Create a `non_secured.bif` file in `C:\Xilinx\enhanced_key_revocation_lab_files` with the following content:

   ```
   //arch = zynqmp; split = false; format = BIN
   the_ROM_image:
   {
    [bootloader, destination_cpu=a53-0] C:\Xilinx\Key_Revocation_Lab
   \Key_Revocation_Platform\export\Key_Revocation_Platform\sw
   \Key_Revocation_Platform\boot\fsbl.elf
    [destination_cpu=a53-0, exception_level=el-3] C:\Xilinx
   \Key_Revocation_Lab\key_revocation_lab\Debug\key_revocation_lab.elf
   }
   ```

   Note: If using Linux, the backslashes will need to be changed to forward slashes.

2. Save the BIF file.

   Note: In this application note `C:\Xilinx\Key_Revocation_Lab` has been used as a Vitis workspace location. If any other workspace location is used, the BIF file contents need to be modified accordingly.

   Note: Linux users must modify this file using the "/".

3. Generate a BI named `BOOT.bin` using the `non_secured.bif` file:

   a. Launch a Windows CMD prompt and change to the directory containing the BIF file, in this case: `cd C:\Xilinx\enhanced_key_revocation_lab_files`.

      Note: If using Linux, change directory to the location where the lab files were extracted.

   b. Run the bootgen command as `bootgen -image non_secured.bif -r -o BOOT.bin -arch zynqmp -w on`.

      Note: Refer to *Bootgen User Guide* (UG1283) for detailed information.

      Note: The Windows and/or Linux system PATH variable needs to point to the bootgen tool or xsct can be launched and used instead. See *Xilinx Software Command-Line Tool (XSCT) Reference Guide* (UG1208).

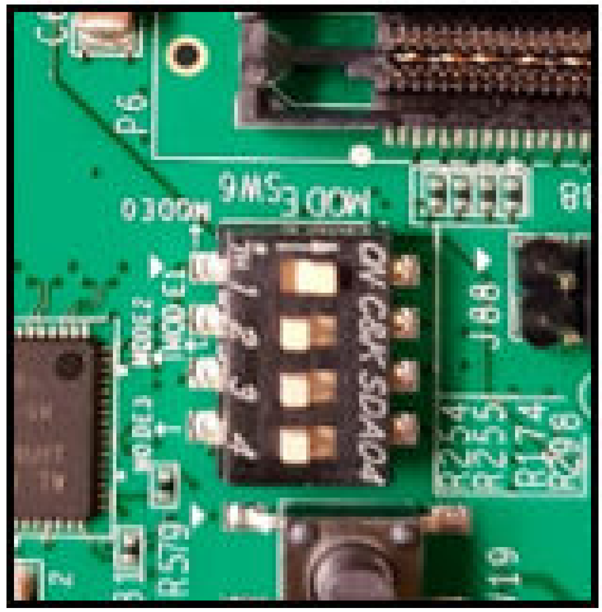
Note: If running on windows, **step a** is a CD command to the directory, and step b can be run as is.

A `BOOT.bin` file is created in `C:\Xilinx\enhanced_key_revocation_lab_files`. This is the BI that must be loaded onto the device.

Run Boot Image

1. Set dip-switch SW6 of the ZCU102 board for SD Card Boot Mode (1=ON; 2, 3, 4=OFF).

Figure 12: **SD Card Boot DIP Switches**



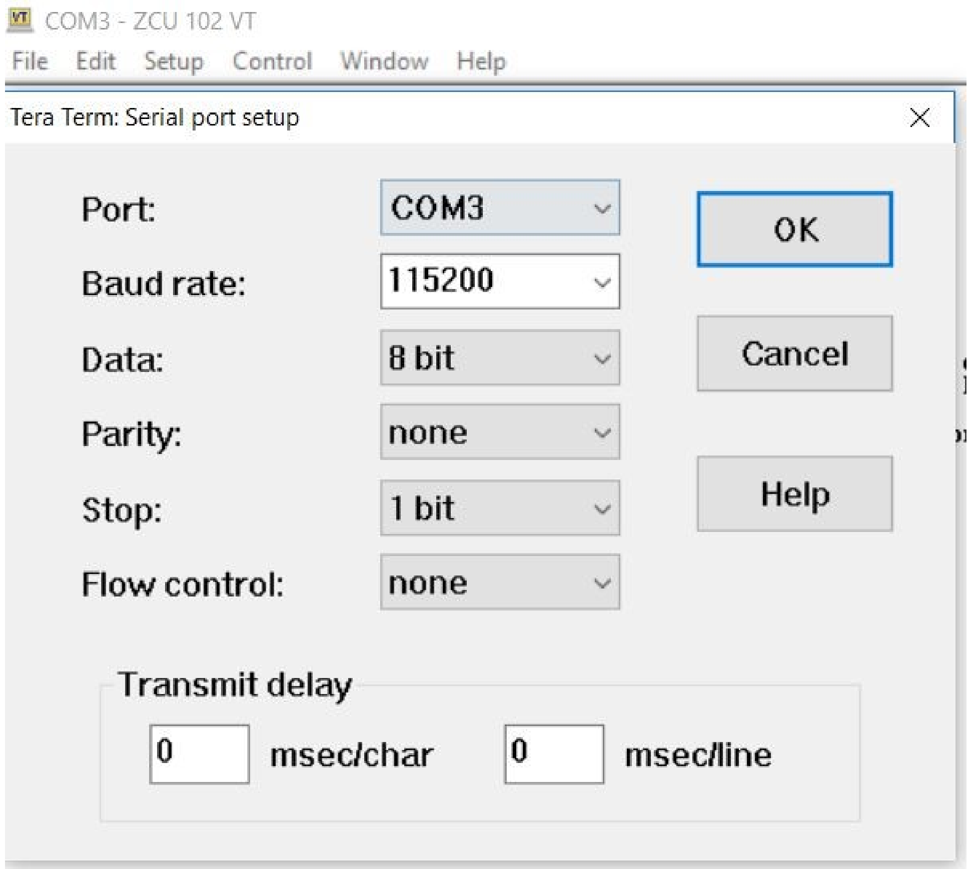
2. Copy the `BOOT.BIN` to the SD Card.
3. Insert the `BOOT.BIN` in the SD card slot.
4. Connect the UART cable.

   Note: The UART cable is connected between the ZCU102 board and computer.
5. Open any terminal window.

   Note: Tera Term is the terminal window used in this example.
6. Connect to the COM port (Baud rate:115200, Data Bits:8, Parity:none, Stop Bits:1).

   Note: In the following image the COM port has been assigned as COM3. It might be different depending on the setup. Use the Windows device manager utility to identify the correct the COM port to be connected for UART output of the ZCU102 board.

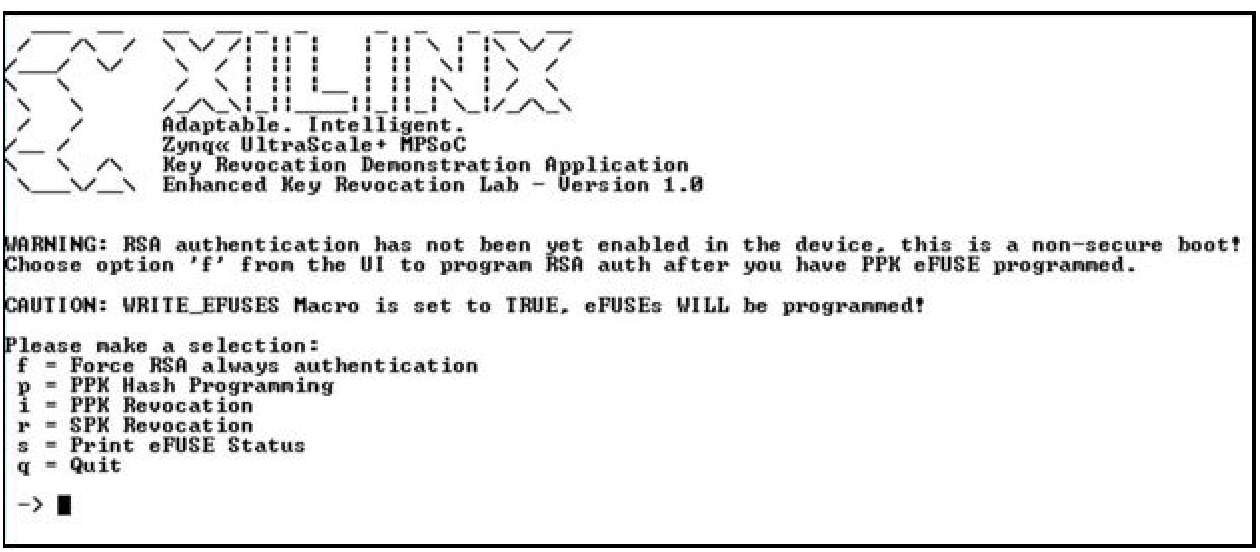
Figure 13: **Com Port Settings**



7. Power on the board using SW1.

   The UI of the lab application is displayed on the serial terminal, as shown in the following image.

Figure 14: **Main Menu**



Getting a display on the serial terminal means that the Vitis and lab application setup was correct. The selection menu in the following screen capture is referred to as main menu throughout this application note. Refer to Menu Options for more information on the UI.

Note: This is non-secured boot. The device has not been provisioned for secure boot. This is done in the upcoming sections, which involves programming eFUSEs using the lab application UI.

> **IMPORTANT!** *The lab application UI prints a WARNING message that the boot was non-secure. It also notifies you that the `WRITE_EFUSES` macro is set to TRUE (eFUSE programming enabled).*

Program the PPK0 and PPK1 Digest eFUSEs

Program the PPK0 eFUSE

Programming the PPK eFUSEs is the first step in securing the ZCU102 device (also referred to as device provisioning). In the Zynq UltraScale+ MPSoC, there are two PPK eFUSEs – PPK0 and PPK1. In this section both the PPK eFUSEs are programmed with SHA3-384 hashes of pre-generated Privacy Enhanced Mail (PEM) files. See Reference Design for the PEM file.

For this task, the non-secure BI generated are used in Generate Boot Image.

1. Power cycle the board.
2. Select **p = PPK Hash Programming** from the main menu.

   A summary of eFUSEs is printed for reference.
3. Enter **y** to confirm PPK programming.
4. Enter **0** to program PPK0.
5. Copy and paste the following PPK0 hash value into the prompt:

   ```
   79F08C4EB1AAF60CB5A655445657C03CF76022444364F490822E87474764FE892AD8FBB38
   CB486536CB3151C3D45B040
   ```

   Note: The corresponding pem file for the hash in step 4 is named `psk0.pem`. It is provided with this application note and required to generate the secure BI in later sections.

   Note: Bootgen can be used to create a PEM file using unique keys. Refer to the *Bootgen User Guide* (UG1283) for detailed information.

   Note: It is recommended to copy the provided PPK0 hash value to a text editor first to make sure there are no line breaks and ensure the value copied to clipboard is correct before pasting it to the application prompt.
6. Enter **y** to confirm PPK0 programming.

Figure 15: **PPK0 eFUSE Programming**

7. Enter any key to return to the main menu.

Program the PPK1 eFUSE

In this section, the PPK1 eFUSE is programmed with SHA3-384 hashes of pre-generated pem files. See Reference Design for the pem file.

After programming the PPK0 eFUSE, program the PPK1 eFuse:

1. Select **p = PPK Hash Programming** from the main menu.

   A summary of eFUSEs is printed by default for reference.

2. Enter **y** to confirm PPK programming.
3. Enter **1** to program PPK1.

4. Copy and paste the following PPK1 hash value into the prompt:

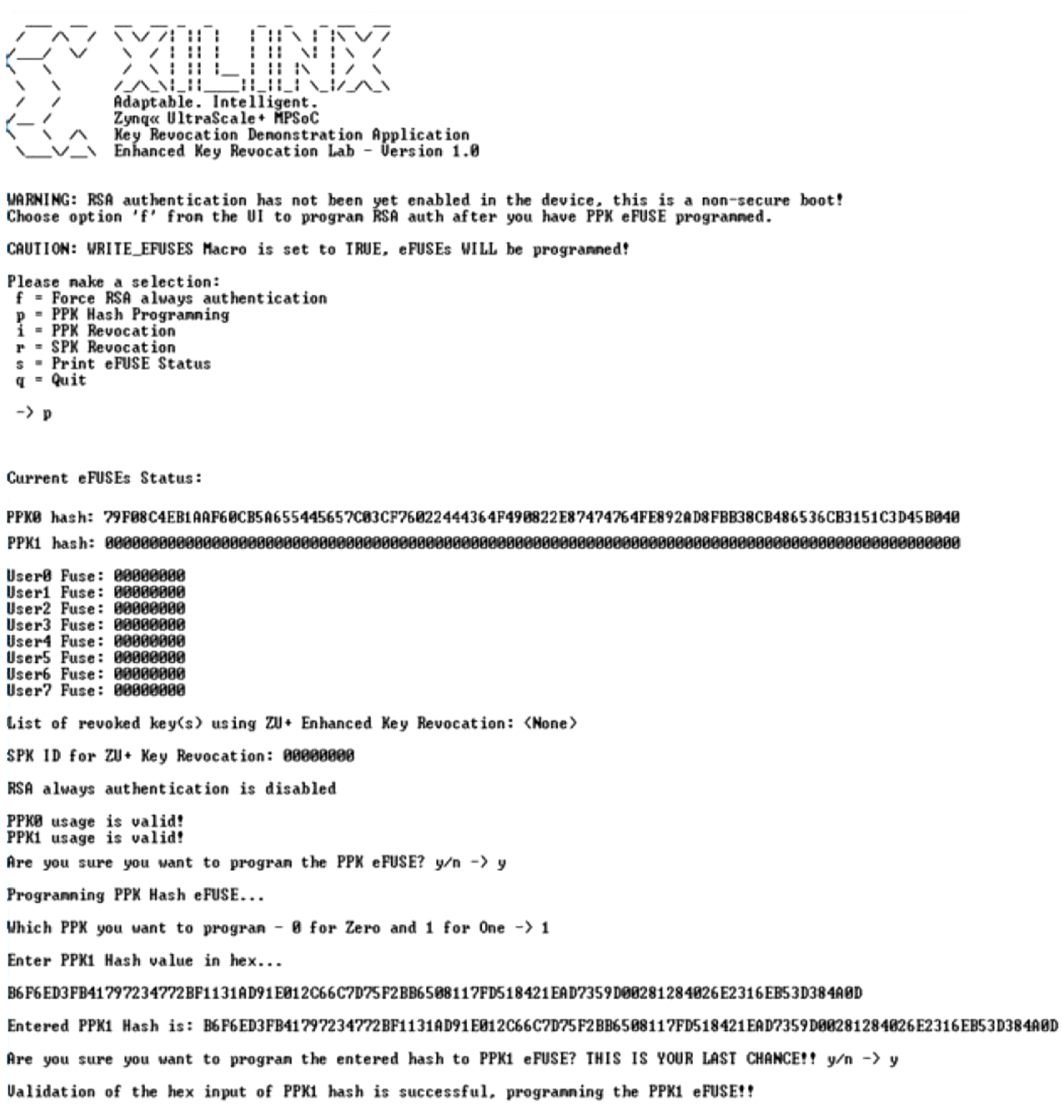
```
B6F6ED3FB41797234772BF1131AD91E012C66C7D75F2BB6508117FD518421EAD7359D0028
1284026E2316EB53D384A0D
```

Note: The corresponding pem file for the PPK1 hash is named `psk1.pem` (see Reference Design).

Note: It is recommended to copy the provided PPK1 hash value to a notepad first to make sure there are no line breaks and ensure the value copied to clipboard is correct before pasting it to the application prompt.

5. Enter **y** to confirm PPK1 programming.

Figure 16: **PPK1 eFUSE Programming**



```
Adaptable. Intelligent.
Zynq« UltraScale+ MPSoC
Key Revocation Demonstration Application
Enhanced Key Revocation Lab - Version 1.0

WARNING: RSA authentication has not been yet enabled in the device, this is a non-secure boot!
Choose option 'f' from the UI to program RSA auth after you have PPK eFUSE programmed.

CAUTION: WRITE_EFUSES Macro is set to TRUE, eFUSEs WILL be programmed!

Please make a selection:
 f = Force RSA always authentication
 p = PPK Hash Programming
 i = PPK Revocation
 r = SPK Revocation
 s = Print eFUSE Status
 q = Quit

 -> p


Current eFUSEs Status:

PPK0 hash: 79F08C4EB1AAF60CB5A655445657C03CF76022444364F490822E87474764FE892AD8FBB38CB486536CB3151C3D45B040
PPK1 hash: 000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

User0 Fuse: 00000000
User1 Fuse: 00000000
User2 Fuse: 00000000
User3 Fuse: 00000000
User4 Fuse: 00000000
User5 Fuse: 00000000
User6 Fuse: 00000000
User7 Fuse: 00000000

List of revoked key(s) using ZU+ Enhanced Key Revocation: <None>

SPK ID for ZU+ Key Revocation: 00000000

RSA always authentication is disabled

PPK0 usage is valid!
PPK1 usage is valid!
Are you sure you want to program the PPK eFUSE? y/n -> y

Programming PPK Hash eFUSE...

Which PPK you want to program - 0 for Zero and 1 for One -> 1

Enter PPK1 Hash value in hex...

B6F6ED3FB41797234772BF1131AD91E012C66C7D75F2BB6508117FD518421EAD7359D00281284026E2316EB53D384A0D

Entered PPK1 Hash is: B6F6ED3FB41797234772BF1131AD91E012C66C7D75F2BB6508117FD518421EAD7359D00281284026E2316EB53D384A0D

Are you sure you want to program the entered hash to PPK1 eFUSE? THIS IS YOUR LAST CHANCE!! y/n -> y

Validation of the hex input of PPK1 hash is successful, programming the PPK1 eFUSE!!
```

X26367-030722

Note: After successfully programming the eFUSE the status is printed by default. Because the eFUSEs programming is irreversible, this is for reference. The eFUSE status is also printed in case an error is encountered during programming eFUSEs.

6. Enter any key to return to the main menu.

Program the RSA_EN eFUSE

Forcing RSA Authentication

After successfully programming both PPK eFUSES, the device is ready for secure-only boot and the RSA_EN eFUSE needs to be programmed.

1. Power cycle the board or ensure you are in the main menu.
2. Open the main menu.
3. Press **s** to select **s = Print eFUSE Status**.
4. Compare the PPK0 and PPK1 hash values displayed on the serial terminal along with the two hashes provided in Program the PPK0 and PPK1 Digest eFUSEs. The values should match.

   ***Note*:** The eFUSE information associated with this lab is displayed in the figure below. The PPK hash fuses are programmed. The User fuses are all zero indicating that nothing has been revoked using the enhanced revocation. The SPK revocation ID is zero indicating that no SPK's have been revoked using the standard revocation. PPK0 and 1 are showing that they are valid so neither have been revoked at this stage of the lab.

Figure 17: **PPK0 and PPK1 Verification**

X26198-020122

5. Power cycle the board.
6. Select **f = RSA always authentication**.
7. Enter **y** to confirm.
8. Verify the PPK hash values.
9. Enter **y** to program the RSA_EN eFUSE.

   ***Note*:** The eFUSE should be programmed successfully, as shown in the following figure.

Figure 18: **RSA Enable eFUSE Write**

```
Adaptable. Intelligent.
Zynq« UltraScale+ MPSoC
Key Revocation Demonstration Application
Enhanced Key Revocation Lab - Version 1.0

WARNING: RSA authentication has not been yet enabled in the device, this is a non-secure booting!
Choose option 'f' from the UI to program RSA auth after you have PPK eFUSE programmed.

CAUTION: WRITE_EFUSES Macro is set to TRUE, eFUSEs WILL be programmed!

Please make a selection:
 f = Force RSA always authentication
 p = PPK Hash Programming
 i = PPK Revocation
 r = SPK Revocation
 s = Print eFUSE Status
 q = Quit

 -> f


Current eFUSEs Status:

PPK0 hash: 79F08C4EB1AAF60CB5A655445657C03CF76022444364F490822E87474764FE892AD8FBB38CB486536CB3151C3D45B040

PPK1 hash: B6F6ED3FB41797234772BF1131AD91E012C66C7D75F2BB6508117FD518421EAD7359D00281284026E2316EB53D384A0D

User0 Fuse: 00000000
User1 Fuse: 00000000
User2 Fuse: 00000000
User3 Fuse: 00000000
User4 Fuse: 00000000
User5 Fuse: 00000000
User6 Fuse: 00000000
User7 Fuse: 00000000

List of revoked key(s) using ZU+ Enhanced Key Revocation: <None>

SPK ID for ZU+ Key Revocation: 00000000

RSA always authentication is disabled

PPK0 usage is valid!
PPK1 usage is valid!


Are you sure you want to enable RSA always authentication? y/n -> y

PPK eFUSE value(s):

PPK0 hash: 79F08C4EB1AAF60CB5A655445657C03CF76022444364F490822E87474764FE892AD8FBB38CB486536CB3151C3D45B040

PPK1 hash: B6F6ED3FB41797234772BF1131AD91E012C66C7D75F2BB6508117FD518421EAD7359D00281284026E2316EB53D384A0D

Do you have the PSK files for PPK hash eFUSE printed above? y/n-> y

Programming RSA ENABLE eFUSE to always force RSA authentication


Selected eFUSE has been written successfully
```

X26199-020122

Verification of Device Provisioning

After successfully programming the PPK eFUSEs and the RSA_EN eFUSE, verify that secure only boot and device provisioning have been enabled successfully, i.e., non-secured BI does not load on the programmed board.

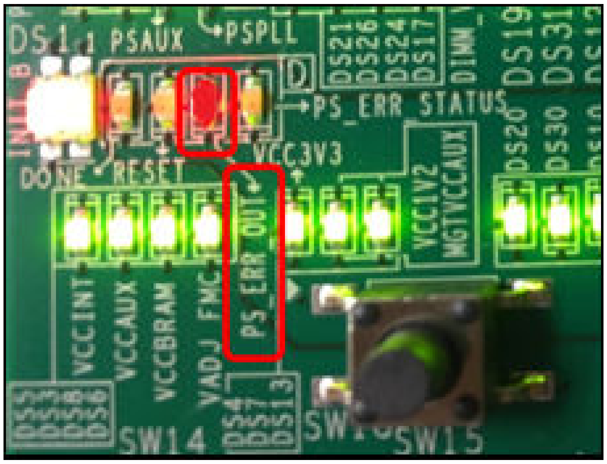
1. Push the **POR_B** button on the board or power cycle the board.

***Note*:** Pushing **POR_B** or power cycling resets the board, forcing a reload of the BI. However, it is expected that the FSBL and lab application in the BI will fail to load.

When the board comes online there is no output on the serial terminal, and both the FSBL and the lab application fail to load. In addition, the PS_ERR_OUT LED glows red, as shown in the following image.

***Note*:** It takes up to 30 seconds for the LED light to turn on.

Figure 19: **PS_ERR_OUT LED**



***Note*:** This change in boot behavior is permanent. Therefore, only the authenticated BI will boot on the ZCU102 device where the eFUSE programming was done. Generating a Secure Boot Image and Booting the Secured ZCU102 Device details how to generate a secured BI using the provided pem files.

Generating a Secure Boot Image and Booting the Secured ZCU102 Device

A new BI containing the pem files must be generated for booting the lab application on the secured ZCU102 device.

1. Create a new BIF file named `secured.bif` with the following content:

```
//arch = zynqmp; split = false; format = BIN
the_ROM_image:
{
 [pskfile]C:\Xilinx\enhanced_key_revocation_lab_files\psk0.pem
 [sskfile]C:\Xilinx\enhanced_key_revocation_lab_files\ssk0.pem
 [auth_params]spk_id = 0x00000000; ppk_select = 0; spk_select = spk-efuse
 [bootloader, destination_cpu=a53-0, authentication = rsa] C:\Xilinx
\Key_Revocation_Lab\Key_Revocation_Platform\export
\Key_Revocation_Platform\sw\Key_Revocation_Platform\boot\fsbl.elf
 [authentication = rsa, destination_cpu=a53-0, exception_level=el-3]
C:\Xilinx\Key_Revocation_Lab\key_revocation_lab\Debug
\key_revocation_lab.elf
}
```

2. Generate a secured `BOOT.BIN` using the following bootgen command:

```
bootgen -image secured.bif -r -o BOOT.bin -arch zynqmp -w on
```

Note: Refer to the *Bootgen User Guide* (UG1283) for detailed information.

3. Copy the new `BOOT.BIN` to the SD card.
4. Power on the board.

   Note: In the serial terminal output, the lab application UI appears, which indicates that with the new BI, the FSBL and lab application have loaded successfully, as shown in the following figure.

Figure 20: **Secure Boot Display**

```
Adaptable. Intelligent.
Zynq« UltraScale+ MPSoC
Key Revocation Demonstration Application
Enhanced Key Revocation Lab - Version 1.0

This device has been booted securely!

CAUTION: WRITE_EFUSES Macro is set to TRUE, eFUSEs WILL be programmed!

Please make a selection:
 p = PPK Hash Programming
 i = PPK Revocation
 r = SPK Revocation
 s = Print eFUSE Status
 q = Quit

 -> s


Current eFUSEs Status:

PPK0 hash: 79F08C4EB1AAF60CB5A655445657C03CF76022444364F490822E87474764FE892AD8FBB38CB486536CB3151C3D45B040

PPK1 hash: B6F6ED3FB41797234772BF1131AD91E012C66C7D75F2BB6508117FD518421EAD7359D00281284026E2316EB53D384A0D

User0 Fuse: 00000000
User1 Fuse: 00000000
User2 Fuse: 00000000
User3 Fuse: 00000000
User4 Fuse: 00000000
User5 Fuse: 00000000
User6 Fuse: 00000000
User7 Fuse: 00000000

List of revoked key(s) using ZU+ Enhanced Key Revocation: <None>

SPK ID for ZU+ Key Revocation: 00000000

RSA always authentication is enabled


Press any key to continue...
```

X26361-030622

Note: In the main menu there is a line of text that says “This device has been booted securely!” confirming secured boot of the ZCU102 device.

Note: If a device is securely booted, the option **f = force RSA always authentication** is not seen in the main menu.

Note: In this example the lab application uses the PPK0 eFUSE and the default SPK ID `0x00000000` (SPK eFUSE has not been programmed with any value) for authenticating both the FSBL and the lab application.

Zynq UltraScale+ Standard Key Revocation

Program SPK ID eFUSE

Generating a Secure Boot Image and Booting the Secured ZCU102 Device demonstrates how to generate a secure BI. The generated BI uses SPK ID as `0x00000000` (default) for the FBSL and the lab application. To make the device and booting process more secure this value must be changed. Perform the following steps to change the SPK ID to `0x00000001`.

Note: An SPK ID of `0x00000001` is used to minimize irreversible programming of the SPK eFUSE because it is the least significant bit. For more practical purposes, the SPK eFUSE can have any value between `0x00000001` and `0xFFFFFFFF`. To maximize SPK ID range the values it should be programmed as current value + next power of 2 (consuming only one bit per programming).

1. Power cycle the board to load the current BIF.
2. Select **r = SPK Revocation** from the main menu.
3. Select **s = Revoking keys by programming SPK eFUSE** from the sub-menu.

   The current SPK ID value is displayed.
4. Enter **00000001** for the new SPK ID.
5. Enter **y** to confirm SPK eFUSE programming, however any pattern of eFUSEs can be written to fulfill key revocation needs.

Figure 21: **Standard SPK Revocation**

```
Adaptable. Intelligent.
Zynq« UltraScale+ MPSoC
Key Revocation Demonstration Application
Enhanced Key Revocation Lab - Version 1.0

This device has been booted securely!

CAUTION: WRITE_EFUSES Macro is set to TRUE, eFUSEs WILL be programmed!

Please make a selection:
 p - PPK Hash Programming
 i - PPK Revocation
 r - SPK Revocation
 s - Print eFUSE Status
 q - Quit

 -> r
Current eFUSEs Status:

PPK0 hash: 79F08C4EB1AAF60CB5A6554456 57C03CF76022444364F490822E87474764FE892AD8FBB38CB486536CB3151C3D45B040

PPK1 hash: B6F6ED3FB41797234772BF1131AD91E012C66C7D75F2BB6508117FD518421EAD7359D00281284026E2316EB53D384A0D
User0 Fuse: 00000000
User1 Fuse: 00000000
User2 Fuse: 00000000
User3 Fuse: 00000000
User4 Fuse: 00000000
User5 Fuse: 00000000
User6 Fuse: 00000000
User7 Fuse: 00000000

List of revoked key(s) using ZU+ Enhanced Key Revocation: <None>

SPK ID for ZU+ Key Revocation: 00000000

RSA always authentication is disabled

PPK0 usage is valid!
PPK1 usage is valid!

Revoking Keys...
Please make a selection:

 u - Revoking keys by programming User eFUSEs (ZU+ Enhanced Key Revocation)
 s - Revoking keys by programming SPK eFUSE (ZU+ Key Revocation)
 u / s -> s

Programming SPK ID eFUSE for ZU+ Key Revocation...

Current SPK ID is: 00000000

Enter the new SPK ID in Hex value for the 32-bit SPK eFUSE to be programmed:
For example enter something like A23456F8 for 0xA23456F8

New SPK ID: 00000001

New SPK ID of ZU+ Key Revocation to be programmed is 00000001 , press y to continue -> y

Validation of hex input for SPK ID is successful, programming SPK eFUSE!!

Selected eFUSE has been written successfully
```

6. Select **s = Print eFUSE Status** from the main menu.
7. View the new SPK ID.

   Verify the correct SPK ID was programmed. The new SPK ID value should be `00000001`.
8. Power cycle the board. The FSBL and lab application fail to load and the PS_ERR_OUT LED glows red, as shown in Figure 19.

   ***Note*:** In this application note, failure to load the BI is purposefully done to show that our security mechanism is working.

   ***Note*:** A failure of the current BI to load on the device indicates that SPK ID revocation worked. Because the current BI uses the SPK ID of the eFUSE as `0x00000000` (default) and the new value of SPK ID in the device is `0x00000001`, the boot is expected to fail. A new BI with the SPK ID set to `0x00000001` must be generated for a successful boot.
9. Modify the `secured.bif` file generated in Generating a Secure Boot Image and Booting the Secured ZCU102 Device.

a. Change the **spk_id** value in the BIF file to `0x00000001` (hex value for 32-bit eFUSE).

b. Save the modified file and name it `secured_mod.bif`.

Note: In the demonstrations, the revocation steps are shown and the same keys are simply reloaded. In a normal scenario, new keys would be generated and incorporated.

```
//arch = zynqmp; split = false; format = BIN
the_ROM_image:
{
[pskfile]C:\Xilinx\enhanced_key_revocation_lab_files\psk0.pem
[auth_params]ppk_select = 0
[bootloader, destination_cpu=a53-0, authentication = rsa, spk_select
= spk-efuse, sskfile = C:\Xilinx\enhanced_key_revocation_lab_files
\ssk0.pem, spk_id = 0x00000001]C:\Xilinx\Key_Revocation_Lab
\Key_Revocation_Platform\export\Key_Revocation_Platform\sw
\Key_Revocation_Platform\boot\fsbl.elf
[authentication = rsa, destination_cpu = a53-0, spk_select = user-
efuse, sskfile = C:\Xilinx\enhanced_key_revocation_lab_files
\ssk1.pem, spk_id = 1]C:\Xilinx\Key_Revocation_Lab\key_revocation_lab
\Debug\key_revocation_lab.elf
}
```

10. Generate a new secured `BOOT.bin` using the bootgen command:

```
bootgen -image secured_mod.bif -r -o BOOT.bin -archzynqmp - w on
```

11. Copy the new `BOOT.BIN` to the SD card.

12. Power on the board.

    Both the FSBL and the lab application should load successfully. The lab UI main menu displays on the serial terminal.

    Note: SPK ID`0x00000001` should be used for BI generation targeted on the programmed ZCU102 device (unless changed to something else).

Zynq UltraScale+ Enhanced Key Revocation

Program User eFUSE

The SPK eFUSE is 32-bits, therefore there are only 32 possible revocations when using the Zynq UltraScale+ MPSoC standard key revocations. Another limitation is that all user partitions must share the same SPK ID with the FSBL. In the current example, the lab application and FSBL both have the SPK ID at `00000001`. To overcome this, there is Zynq UltraScale+ MPSoC Enhanced Key Revocation, which allows different SPK IDs across multiple user partitions using User eFUSES.

Note: The FSBL must always use SPK eFUSE for SPK ID. Zynq UltraScale+ MPSoC enhanced key revocation can only be used for user applications/partitions.

In this section, a new BI is first generated, which uses User eFUSE SPK ID 1 for the lab application. With the new BI successfully loaded, SPK ID 1 is revoked using lab UI, which leads to failure in loading of the lab application when the board is re-booted. A new SPK ID is then assigned to the lab application (in the BIF file) and a new BI is generated and loaded into the device to verify successful loading of the user application (lab application) with the new SPK ID.

Note: Zynq UltraScale+ MPSoC standard key revocation uses hexadecimal value of 32-bit SPK eFUSE. However, Zynq UltraScale+ MPSoC enhanced key revocation needs key decimal numbers between 1–255.

1. Create a new BIF file.
2. Enter file name `secured_eKeyR.bif` with the following contents:

   ```
   //arch = zynqmp; split = false; format = BIN
   the_ROM_image:
   {
   [pskfile]C:\Xilinx\enhanced_key_revocation_lab_files\psk0.pem
   [auth_params]ppk_select = 0
   [bootloader, destination_cpu=a53-0, authentication = rsa, spk_select =
   spk-efuse, sskfile = C:\Xilinx\enhanced_key_revocation_lab_files
   \ssk0.pem, spk_id = 0x00000001]C:\Xilinx\Key_Revocation_Lab
   \Key_Revocation_Platform\export\Key_Revocation_Platform\sw
   \Key_Revocation_Platform\boot\fsbl.elf
   [authentication = rsa, destination_cpu = a53-0, spk_select = user-efuse,
   sskfile = C:\Xilinx\enhanced_key_revocation_lab_files\ssk1.pem, spk_id =
   1]C:\Xilinx\Key_Revocation_Lab\key_revocation_lab\Debug
   \key_revocation_lab.elf
   }
   ```

   Note: The SPK-select field of the BIF file determines which revocation method will be used. The values for this field can be **spk-efuse** (Zynq UltraScale+ MPSoC standard key revocation) or **user-efuse** (Zynq UltraScale+ MPSoC enhanced key revocation). In the `secured_eKeyR.bif` file, the SPK-eFUSE value is used for the FSBL SPK-select field and the user-eFUSE value is used for the corresponding lab application field.

   Note: In the `secured_eKeyR.bif` file, `ssk0.pem` file is used for the FSBL and `ssk1.pem` file is used for the lab application.

3. Generate a new secured BI `BOOT.bin` using the bootgen command:

   ```
   bootgen -image secured_eKeyR.bif -r -o BOOT.bin -arch zynqmp -w on
   ```

4. Select a new BI and copy it to the SD card.
5. Power on the board.

   The lab application loads successfully and the main menu displays.

   Note: SPK ID 1 for the lab application was successful because it has not been revoked yet.

6. Select **r = SPK Revocation** from the main menu.
7. Select **u = Revoking keys** by **programming User eFUSEs** from the sub-menu.
8. Set **001** as the SPK ID to be revoked.
9. Set **y** to confirm.

   Note: The tool expects an integer value between 1 – 256. The SPK ID must be entered as three digits (i.e., for `1` enter `001`, for `32` enter `032`, and for `150` enter `150`.

Figure 22: **Enhanced SPK Revocation**

```
Adaptable. Intelligent.
Zynq« UltraScale+ MPSoC
Key Revocation Demonstration Application
Enhanced Key Revocation Lab - Version 1.0

This device has been booted securely!

CAUTION: WRITE_EFUSES Macro is set to TRUE, eFUSEs WILL be programmed!

Please make a selection:
 p - PPK Hash Programming
 i - PPK Revocation
 r - SPK Revocation
 s - Print eFUSE Status
 q - Quit

 -> r

Current eFUSEs Status:

PPK0 hash: 79F08C4EB1AAF60CB5A655445657C03CF76022444364F490822E87474764FE892AD8FBB38CB486536CB3151C3D45B040

PPK1 hash: B6F6ED3FB41797234772BF1131AD91E012C66C7D75F2BB6508117FD518421EAD7359D00281284026E2316EB53D384A0D

User0 Fuse: 00000000
User1 Fuse: 00000000
User2 Fuse: 00000000
User3 Fuse: 00000000
User4 Fuse: 00000000
User5 Fuse: 00000000
User6 Fuse: 00000000
User7 Fuse: 00000000

List of revoked key(s) using ZU+ Enhanced Key Revocation: <None>

SPK ID for ZU+ Key Revocation: 00000001

RSA always authentication is disabled

PPK0 usage is valid!
PPK1 usage is valid!

Revoking Keys...
Please make a selection:

 u - Revoking keys by programming User eFUSEs (ZU+ Enhanced Key Revocation)
 s - Revoking keys by programming SPK eFUSE (ZU+ Key Revocation)
 u / s -> u

Revoking using ZU+ Enhanced Key Revocation...

Enter User-eFUSE SPK ID between 1 - 256 to be revoked
For example to enter 1 enter as 001, to enter 32 enter as 032 and to enter 160 enter as 160

User SPK ID: 001
User-eFUSE SPK ID to be revoked is 1 , enter y to continue -> y

Current status of User-eFUSE0 is 00000000
After programming User-eFUSE0 value will be 00000001, do you want to continue? y/n -> y

Validation of hex input for programming eFUSE is successful, programming User-eFUSE0!!

Selected eFUSE has been written successfully
```

The UI prints out the current SPK ID and the one it will be changed to.

10. Enter **y** to reconfirm the eFUSE programming.

    Verify that the user eFUSE was successfully programmed.

11. Select **s = Print eFUSE Status** from the main menu.

    ***Note*:** In the status for **User0 eFUSE**, the new value should be printed (i.e., **00000001**) and in the list of revoked keys, 1 will be listed among the revoked keys.

12. Power cycle the board.

    The serial terminal shows that the FBSL loads, but the lab application fails to load. In addition, the PS_ERR_OUT LED glows red. This verifies that revocation of SPK ID 1 worked because in the current BI, the lab application uses SPK ID 1 (User eFUSE) which has been successfully revoked restricting its further usage.

    ***Note*:** It takes up to 30 seconds for the LED light to light up.

Note: In the programmed ZCU102 board, no user application can use the revoked user-eFUSE SPK ID 1.

With SPK ID 1 revoked, the lab application now must use a different SPK ID between 2-256 for a successful boot. In the following steps, the BIF file is modified with a new value for the lab application SPK ID, which will be 2.

13. Select `secured_eKeyR.bif`.

    a. Set **spk_id** field value from `1` to `2`.

    b. Save the file as `secured_eKeyR_mod.bif`.

```
//arch = zynqmp; split = false; format = BIN
the_ROM_image:
{
[pskfile]C:\Xilinx\enhanced_key_revocation_lab_files\psk0.pem
[auth_params]ppk_select = 0
[bootloader, destination_cpu=a53-0, authentication = rsa, spk_select
= spk-efuse, sskfile = C:\Xilinx\enhanced_key_revocation_lab_files
\ssk0.pem, spk_id = 0x00000001]C:\Xilinx\Key_Revocation_Lab
\Key_Revocation_Platform\export\Key_Revocation_Platform\sw
\Key_Revocation_Platform\boot\fsbl.elf
[authentication = rsa, destination_cpu = a53-0, spk_select = user-
efuse, sskfile = C:\Xilinx\enhanced_key_revocation_lab_files
\ssk1.pem, spk_id = 2]C:\Xilinx\Key_Revocation_Lab\key_revocation_lab
\Debug\key_revocation_lab.elf
}
```

14. Generate a new secured `BOOT.bin` BI using the bootgen command:

```
bootgen -image secured_eKeyR_mod.bif -r -o BOOT.bin -arch zynqmp -w on
```

15. Copy the new `BOOT.BIN` BI to the SD card.

16. Power on the board.

    Both the FSBL and lab application load successfully. User eFUSE SPK ID 2 for the lab application works because that key has not been revoked.

PPK0 Revocation

Program PPK0_INVLD eFUSE

Due to security concerns such as losing or compromising the Primary Secret Key (PSK), there might be a situation where usage of a PPK eFUSE must be revoked. This is a one-time operation (i.e., after a PPK – 0 or 1 is revoked it cannot be undone). Therefore, exercise caution while using this feature. Revoking both the PPKs or having an un-revoked/programmed PPK and not having the corresponding key/pem file leads to bricking of the board (provided the RSA always enable eFUSE is already programmed).

IMPORTANT! *DO NOT revoke a PPK unless the other one is programmed or there will be no way to boot the device.*

Because both PPK0 and PPK1 have been programmed, this section demonstrates how to invalidate the use of PPK0 as a PPK revocation example. After successful revocation, booting fails if the BI attempts to use PPK0. Changing the BIF file to use PPK1 successfully boots the device.

This task demonstrates how to invalidate the use of PPK0 as a PPK revocation example.

1. Select **i = PPK Revocation** from the main menu.

   The status of eFUSEs is displayed for reference. Verify in the printed status that both PPK0 and PPK1 are valid.

2. Enter **y** to proceed with PPK revocation.

   The status of PPK0 and PPK1 is printed for reference.

3. Enter **0** for revoking PPK0.
4. Enter **y** to confirm.

   Confirmation of successful eFUSE programming is printed in the UI, as shown in the following figure.

Figure 23: **PPK0 Revocation**

```
Adaptable. Intelligent.
Zynq« UltraScale+ MPSoC
Key Revocation Demonstration Application
Enhanced Key Revocation Lab - Version 1.0

This device has been booted securely!

CAUTION: WRITE_EFUSES Macro is set to TRUE, eFUSEs WILL be programmed!

Please make a selection:
 p = PPK Hash Programming
 i = PPK Revocation
 r = SPK Revocation
 s = Print eFUSE Status
 q = Quit

 -> i


Current eFUSEs Status:

PPK0 hash: 79F08C4EB1AAF60CB5A655445657C03CF76022444364F490822E87474764FE892AD8FBB38CB486536CB3151C3D45B040

PPK1 hash: B6F6ED3FB41797234772BF1131AD91E012C66C7D75F2BB6508117FD518421EAD7359D00281284026E2316EB53D384A0D

User0 Fuse: 00000001
User1 Fuse: 00000000
User2 Fuse: 00000000
User3 Fuse: 00000000
User4 Fuse: 00000000
User5 Fuse: 00000000
User6 Fuse: 00000000
User7 Fuse: 00000000

List of revoked key(s) using ZU+ Enhanced Key Revocation: 1

SPK ID for ZU+ Key Revocation: 00000001

RSA always authentication is enabled

PPK0 usage is valid!
PPK1 usage is valid!

Are you sure you want to revoke PPK 0/1 eFUSE? y/n -> y

Revoking PPK eFUSE...

The PPK eFUSE values are:

PPK0 hash: 79F08C4EB1AAF60CB5A655445657C03CF76022444364F490822E87474764FE892AD8FBB38CB486536CB3151C3D45B040

PPK1 hash: B6F6ED3FB41797234772BF1131AD91E012C66C7D75F2BB6508117FD518421EAD7359D00281284026E2316EB53D384A0D

Which PPK you want to revoke - 0 for Zero and 1 for One -> 0

Are you sure you want to permanently invalidate usage of PPK0? y/n -> y
Revoking PPK0 ...


Selected eFUSE has been written successfully
```

5. Power cycle the board.

   ***Note*:** Both the FSBL and lab application do not load because the BI is still using a revoked PPK (i.e., PPK0). Booting failure can also be confirmed by observing the LED color of `PS_ERR_OUT`, which is red. It takes up to 30 seconds for the LED to light up.

   ***Note*:** After PPK0 has been revoked it can no longer be used in the BI, therefore, a new BI needs to be generated using PPK1.

6. Select `secured_eKeyR_mod.bif`.

   a. Set the **pskfile** field to use **psk1.pem**.

      Use the correct location of the file. In this case:

      `C:\Xilinx\enhanced_key_revocation_lab_files\psk1.pem`

   b. Set **ppk_select** to **1** to use **PPK1 eFUSE** (see the following code).

c. Save the file as `secured_eKeyR_PPKr.bif`.

```
//arch = zynqmp; split = false; format = BIN
the_ROM_image:
{
[pskfile]C:\Xilinx\enhanced_key_revocation_lab_files\psk1.pem
[auth_params]ppk_select = 1
[bootloader, destination_cpu=a53-0, authentication = rsa, spk_select
= spk-efuse, sskfile = C:\Xilinx\enhanced_key_revocation_lab_files
\ssk0.pem, spk_id = 0x00000001]C:\Xilinx\Key_Revocation_Lab
\Key_Revocation_Platform\export\Key_Revocation_Platform\sw
\Key_Revocation_Platform\boot\fsbl.elf
[authentication = rsa, destination_cpu = a53-0, spk_select = user-
efuse, sskfile = C:\Xilinx\enhanced_key_revocation_lab_files
\ssk1.pem, spk_id = 2]C:\Xilinx\Key_Revocation_Lab\key_revocation_lab
\Debug\key_revocation_lab.elf
}
```

Note: The corresponding pem file for PPK0 is `psk0.pem` and for PPK1 it is `psk1.pem`.

7. Generate a new secured `BOOT.bin` BI using the bootgen command:

```
bootgen -image secured_eKeyR_PPKr.bif -r -o BOOT.bin -arch zynqmp -w on
```

8. Copy the new BI to the SD card.
9. Power on the board.

   Both the FSBL and the lab application load successfully. The current BI is using a valid PPK1.

After successful execution of all the steps in this tutorial, the device state is as follows:

- PPK1 is the only valid PPK eFUSE and the corresponding pem file is `psk1.pem`.
  - PSK1 is programmed with hash for `psk1.pem`.
- RSA always authentication is enabled.
- SPK ID, with regards to Standard Zynq UltraScale+ key revocation is `0x00000001`.
- SPK ID 1 is invalid, with regards to Zynq UltraScale+ enhanced key revocation.

After PPK0 has been revoked it can no longer be used in the BI, therefore, a new BI needs to be generated using PPK1.

Key Revocation Lab Results

1. Select **POR_B**.
2. Select **s = Print eFUSE Status** from the main menu.

   Verify the final status of all the eFUSEs, as shown below.

Figure 24: **Success with Updated Boot Image**

```
Adaptable. Intelligent.
Zynq« UltraScale+ MPSoC
Key Revocation Demonstration Application
Enhanced Key Revocation Lab - Version 1.0

This device has been booted securely!

CAUTION: WRITE_EFUSES Macro is set to FALSE, eFUSEs WILL NOT be programmed!

Please make a selection:
 p - PPK Hash Programming
 i - PPK Revocation
 r - SPK Revocation
 s - Print eFUSE Status
 q - Quit

 -> s


Current eFUSEs Status:

PPK0 hash: 79F08C4EB1AAF60CB5A655445657C03CF76022444364F490822E87474764FE892AD8FBB38CB486536CB3151C3D45B040

PPK1 hash: B6F6ED3FB41797234772BF1131AD91E012C66C7D75F2BB6508117FD518421EAD7359D00281284026E2316EB53D384A0D

User0 Fuse: 00000001
User1 Fuse: 00000000
User2 Fuse: 00000000
User3 Fuse: 00000000
User4 Fuse: 00000000
User5 Fuse: 00000000
User6 Fuse: 00000000
User7 Fuse: 00000000

List of revoked key(s) using ZU+ Enhanced Key Revocation: 1

SPK ID for ZU+ Key Revocation: 00000001

RSA always authentication is enabled

PPK0 has been revoked, usage invalid!
PPK1 usage is valid!
```

***Note*:** The status of the eFUSEs in the figure above are provided assuming that the lab application was run on a ZCU102 board where none of the eFUSEs used in the tutorial were previously programmed.

Reference Design

Download the reference design files for this application note from the Xilinx website.

Reference Design Matrix

The following checklist indicates the procedures used for the provided reference design.

Table 2: **Reference Design Matrix**

Parameter	Description
General	
Developer name	Xilinx
Target devices	Zynq Zynq UltraScale+ MPSoCs
Source code provided?	Y
Source code format (if provided)	C
Design uses code or IP from existing reference design, application note, third party or Vivado® software? If yes, list.	N/A
Simulation	
Functional simulation performed	N
Timing simulation performed?	N
Test bench provided for functional and timing simulation?	N
Test bench format	N/A

Table 2: **Reference Design Matrix** *(cont'd)*

Parameter	Description
Simulator software and version	N/A
SPICE/IBIS simulations	N
Implementation	
Synthesis software tools/versions used	N/A
Implementation software tool(s) and version	N/A
Static timing analysis performed?	N
Hardware Verification	
Hardware verified?	Y
Platform used for verification	ZCU102

Reference Design Contents

The contents of the reference design downloaded are as follows:

Pre-generated public keys:

- `psk0.pem` (primary secret key)
- `psk1.pem` (primary secret key)
- `ssk0.pem` (secondary secret key)
- `ssk1.pem` (secondary secret key)

Pre-generated hash for two primary keys:

- `hash_ppk0`
- `hash_ppk1`

Source files needed to build lab application:

- `key_revocation_lab_main.c`
- `key_revocation_lab_main.h`
- `key_revocation_lab_utils.c`
- `key_revocation_lab_utils.h`

BIF_files sub-directory which contains the BIF files:

- `non_secured.bif`– Used to first boot the lab application for device provisioning.
- `secured.bif`– Used to boot after device is provisioned (PPK0, PPK1, and RSA_EN have been programmed).
- `secured_mod`– Used to boot when the default SPK_ID is modified for Zynq UltraScale+ MPSoC Standard Key Revocation.
- `secured_eKeyR.bif`– Used to boot to demonstrate Zynq UltraScale+ MPSoC Enhanced Key Revocation.

- `secured_eKeyR_mod`– Used to boot when User-eFUSE SPK ID is changed.
- `secured_eKeyR_PPKr`– Used to boot when PPK0 is revoked.

Menu Options

The main menu options are as follows:

- f = Force RSA always authentication – Select this to program the RSA_EN eFUSE.
- p = PPK Hash Programming – Select this to program PPK eFUSEs.
- i = PPK Revocation – Select this to revoke PPK eFUSEs.
- r = SPK Revocation – Select this to enter sub-menu for programming either SPK or User eFUSE (Secondary Key Revocation).
- s = Print eFUSE Status – Select this to print status of the eFUSEs.
- q = Quit – Select this to exit the lab application.

Conclusion

This application note details on how to use the security-related eFUSEs to enable secure boot on a ZCU102 device (i.e., device provisioning). It also demonstrates how to perform key revocations for partitions/applications using SPK eFUSE (Zynq UltraScale+ standard key revocation) and User eFUSEs (Zynq UltraScale+ enhanced key revocation). Lastly, it demonstrates PPK revocation and the importance of caution while using this feature. The source code of this lab example can be studied to understand which APIs to use for security-related eFUSE programming, and users can modify the given example code according to their needs.

IMPORTANT! *Exercise extreme caution while using this lab exercise. eFUSE programming is permanent and can lead to the board being unusable if done carelessly.*

References

These documents provide supplemental material useful with this guide:

1. *Programming BBRAM and eFUSEs* (XAPP1319)
2. *Zynq UltraScale+ MPSoC: Embedded Design Tutorial* (UG1209)
3. Vitis Library Xilskey
4. *Zynq UltraScale+ Device Technical Reference Manual* (UG1085)
5. *Bootgen User Guide* (UG1283)

Revision History

The following table shows the revision history for this document.

Section	Revision Summary
03/14/2022 Version 1.1	
Throughout document	Removed SDK and updated to Vitis Added new screen shots and steps. Updated BIF code content. Updated figure titles
Section: Create an FSBL for the Arm Cortex-A53 Core	Added new section (after step 3)
Section: Modify BSP to Include XilSkey Library	Added new section
Section: Create a Lab Application for the Arm Cortex-A53 based APU	Added 3 new sections
Section: Verification of Device Provisioning	Added new figure
06/26/2020 Version 1.0	
Initial Release.	N/A

Please Read: Important Legal Notices

AUTOMOTIVE APPLICATIONS DISCLAIMER

Copyright