

# **External Secure Storage Using the PUF**

Author: Nathan Menhorn

# Summary

To store data in non-volatile memory (NVM) using a Zynq<sup>®</sup> UltraScale+<sup>™</sup> device, data must be stored externally and should be encrypted if it is confidential. All Zynq UltraScale+ devices have a built-in physically unclonable function (PUF), which can generate a cryptographically strong, device-unique encryption key that can be used in combination with the built-in advanced encryption standard (AES) cryptographic core. This key cannot be read by a user, allowing for a heightened level of key security. Only if a Zynq UltraScale+ device is provisioned to store the PUF configuration information in eFUSEs and if Rivest-Shamir-Adleman (RSA) Authentication is registered and enabled in eFUSEs, then the PUF's device-unique encryption key can be used to encrypt and decrypt user data, which can then be stored and read from external non-volatile memory. Download the reference design files for this application note from the Xilinx website. For detailed information about the design files, see Reference Design.

# Introduction

The PUF takes advantage of silicon variations unique to Zynq UltraScale+ devices to generate a device-unique encryption key that cannot be read by anyone, including the user. Along with generating a unique encryption key, the PUF also generates the required helper data so that the PUF can exactly regenerate the encryption key later. The details of the PUF are described in the *Zynq UltraScale+ MPSoC: Technical Reference Manual* (UG1085) [Ref 2]. Normally, the PUF's encryption key, referred to as the Key Encryption Key (KEK), is used for encrypting a user's plain-text red key so that a user's red key can be stored encrypted in black key form in either eFUSES or the boot header. The black encryption key is then decrypted using the PUF's KEK to generate the red key, which in turn is used for decrypting the boot information during secure boot. This use of the PUF is shown in the following figure.

**IMPORTANT:** The PUF characterization results confirm that over the life of the device, the PUF is expected to reliably regenerate the KEK across all voltages and temperatures **assuming registration at a nominal voltage and temperature**.

 $\diamondsuit$ 

**IMPORTANT:** The **RSA\_EN eFUSE** must be programmed in order to use the PUF's device-unique encryption key to encrypt and decrypt user data. Once this is programmed, Boot Header based authentication (bh\_auth\_enable) can no longer be used.

Xilinx is creating an environment where employees, customers, and partners feel welcome and included. To that end, we're removing noninclusive language from our products and related collateral. We've launched an internal initiative to remove language that could exclude people or reinforce historical biases, including terms embedded in our software and IPs. You may still find examples of non-inclusive language in our older products as we work to make these changes and align with evolving industry standards. Follow this link for more information.

#### AMDA XILINX



Figure 1: Encrypting and Decrypting the Device Key Using PUF

When the PUF is registered in eFUSEs and RSA authentication is enabled in eFUSEs, documented in *Programming BBRAM and eFUSEs* (XAPP1319) [Ref 3], the PUF's device-unique encryption key can be used to encrypt and decrypt any user data. This encrypted data can then be stored externally to the Zynq UltraScale+ device, which is the focus of this application note. The RSA authentication settings cannot be stored in the boot header when using the PUF to encrypt and decrypt user data.

**IMPORTANT:** When the RSA\_ENABLE eFUSEs are programmed, boot header authentication is no longer permitted.

The process of using the PUF to encrypt user data is shown in Figure 2 and works as follows: a user generates data that must be encrypted and appends an optional ID. This optional ID can be used to validate that the correct version of data that is being used, such as when the data consists of encryption key information or configuration and is useful in preventing replay attacks. Even though the ID is optional, Xilinx highly recommends using it to ensure a more secure system. The optional ID enables key/data revocation as the user data packet can be revoked by burning one of the 256-bit user eFUSEs. Each of the 256-bit user eFUSEs can be mapped to 256 different 8-bit user IDs. Keep in mind that user eFUSEs are a shared resource as the fuses could be used for Enhanced Key Revocation software, a tamper log (see *Developing Tamper-Resistant Designs with Zyng UltraScale+ Devices* (XAPP1323) [Ref 4], or any other user function.

Next, the PUF is enabled to regenerate the PUF's device-unique encryption key, which is loaded into the AES cryptographic core to encrypt the data. Xilinx recommends minimizing the use of the PUF's key by keeping the user data small or implementing an advanced key-rolling architecture where the PUF's device-unique key is only used to encrypt the first portion of a larger sized data, thereby minimizing its exposure. This helps to avoid differential power analysis (DPA) attacks. After the encrypted data is written to external memory, the data is read





back and decrypted to verify the process using the GCM authentication tag. If the data is authenticated, the user selected ID is safe to use. Conversely, if the data verification fails, a revocation penalty can take place, such as burning an associated user eFUSE.



Figure 2: Normal Encryption Process Using PUF

Decrypting external data using the PUF is shown in Figure 3 and works as follows: the encrypted data packet is read from the external memory location followed by regeneration of the PUF decryption key. The data is then decrypted and authenticated via the GCM tag. If authentication passes and if the ID from the decrypted data has not been revoked in user eFUSEs, then the data is valid and can be used. Conversely, if the GCM tag authentication fails, then a penalty can be invoked and the decryption process could be stopped to avoid side channel attacks such as DPA. Furthermore, if the decryption process authenticates but the data's ID has been revoked in user eFUSEs, the data is invalid and should not be used.



**IMPORTANT:** The PUF KEK isn't a FIPS legal key for storing data outside a cryptographic boundary. However, you can create a FIPS-legal KEK, encrypt the FIPS-legal KEK with the PUF KEK, store the encrypted FIPS-legal KEK in eFUSEs, and subsequently use the FIPS-legal KEK to store data outside the cryptographic boundary.

www.xilinx.com

Send Feedback



Figure 3: Using the PUF for Decryption

#### **Hardware and Software Requirements**

The hardware and software requirements for the reference systems are as follows:

- ZCU102 Evaluation Board
- AC power adapter (12 VDC)
- USB type-A to USB mini-B cable x2
- Optional Platform JTAG hardware and associated cables
- Secure Digital (SD) card formatted using the FAT file system
- Xilinx Vitis™ Development Environment (Vitis IDE) 2021.2
- Required design files, which can be downloaded here.



**IMPORTANT:** Programming any of the eFUSE settings noted in Table 12-13 in Zynq UltraScale+ MPSoC: Technical Reference (UG1085) [Ref 2] precludes Xilinx test access. Consequently, Xilinx may not accept return material authorization (RMA) request.



#### AMD**.7** XILINX

# Create a New Embedded Project for the Zynq UltraScale+ MPSoC

Perform the following steps to create a new embedded project for the Zynq UltraScale+ MPSoC. A brief description is covered in this section. Step-by-step instructions can be found in Appendix A. For detailed elaboration on each step, refer to the *UltraScale+ MPSoC: Embedded Design Tutorial* (UG1209) [Ref 5] for further details.

- 1. Open up Vivado<sup>®</sup> Design Suite and create the hardware design required for the Zynq UltraScale+ ZCU102 Evaluation Board. The PL is not required for this lab so all the PS-PL interfaces are disabled and no bitstream is exported.
- 2. Export the hardware and launch Xilinx Vitis® IDE from within the Vivado Design Suite.
- 3. Create a platform project using the XSA file exported from Vivado. The platform projects will automatically create ZCU102\_XAPP1333 platform that includes standalone domain BSP, first stage boot loader projects called **zynqmp\_fsbl** and **zynqmp\_pmufw** along with their associated Board Support Packages named **zynqmp\_fsbl\_bsp** and **zynqmp\_pmufw\_bsp** running respectively on the ARM Cortex-A53 processor in the APU domain and TMR MicroBaze processor in the PMU domain.
- 4. Build the platform, including **zynqmp\_fsbl and zynqmp\_pmufw**, by right-clicking on the ZCU102\_XAPP1333 platform and select **Project -> Build Project** from the main menu.
- 5. Create a HelloWorld project to verify the hardware and software setup before proceeding.

# **Key Generation**

Key generation is covered in detail in the Secure Boot section of *UltraScale+ MPSoC: Embedded Design Tutorial* (UG1209) [Ref 5] so only a summary pertaining to this application note is documented here.

#### **AES Key Generation**

Create a new directory in the Xilinx Vitis workspace root directory (called Keys). The Vitis root directory can be found the same level as the HelloWorld folder. Generate a device key and its associated IV, an operational key, and one partition block key and its associated IV. Combine these keys and IVs into a file named multiple\_keys.nky. Alternatively, copy the Keys folder found in the reference design documents to use for this lab or, if desired, use them as a template and insert your own key and IV values.

Device	zcu9eg;
Key O	0123456789012345678901234567890123456789012345678901234567890123;
IV	01DBD60260A7EC34DE5F6A494;
Key Opt	E070C542B6680A855724793A75222391E663CBD35F45D070F22F703A5CA31B45;
Key 1	0000001000000100000010000001000000100000
IV 1	0000001000000100000001;



Encrypting the boot image is not required to use the PUF for encrypting user data. However, Xilinx highly recommends doing so, which is used throughout this application note.



**IMPORTANT:** Be sure to use your own AES keys and associated IVs for operational devices. The keys provided in this lab are for demonstration purposes and are not cryptographically strong. Per the NIST Special Publication (SP) 800-38D Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM), new IVs need to be used each time a key is used to encrypt new data. This means that if the boot image is updated, a new IV needs to be selected and provided to Bootgen.

#### **RSA Asymmetric Key Generation**

For this application note, generate a pair of RSA keys called **psk0.pem** and **ssk0.pem**. Alternatively, these keys are provided in the design documents in the Keys folder. RSA authentication is required to use the PUF for encrypting and decrypting user data. While this application note does not require the use of a secondary key set, Xilinx highly recommends doing so in an operational application.

#### **Generate SHA3 of Public RSA Asymmetric Key**

Generate the associated SHA3 hash of the RSA PPK and name the output file **sha3.txt**. Alternatively, this hash can be found in the design documents in the Keys folder.

# **PUF eFUSE Configuration**

 $\diamondsuit$ 

**IMPORTANT:** THESE INSTRUCTIONS MODIFY THE EFUSES ON THE ZCU102 DEVELOPMENT BOARD AND MAY LIMIT FUTURE USE OF THE DEVELOPMENT BOARD FOR NON-SECURE TESTING AND DEBUGGING!



**IMPORTANT:** Programming any of the noted eFUSE settings noted in Table 12-13 Zynq UltraScale+ MPSoC: Technical Reference Manual (UG1085) [Ref 2] preclude Xilinx test access. Consequently, Xilinx might not accept return material authorization (RMA) requests. See the important note below Table 12-13 of the Zynq UltraScale+ MPSoC: Technical Reference (UG1085) [Ref 2].

#### **PUF eFUSE Settings**

PUF registration is covered in detail in *Using the PUF* in the *UltraScale+ MPSoC: Embedded Design Tutorial* (UG1209) [Ref 5] so only a summary pertaining to this application note is documented here.

1. In the Vitis workspace for this application note, right-click on the **platform.spr** that is located under **ZCU102\_XAPP1333** platform in the **Explorer** view and click **Open**.



2. Select **Board Support Package** under **standalone on psu\_cortexa53\_0** in the **ZCU102\_XAPP1333** platform view and click **Modify BSP settings**.

vitis_2021.2_workspace_ZCU102 - ZCU102_XAPP1333/platformspr - Vitis IDE						
<u>File Edit</u> Search <u>Xilinx P</u> roject <u>W</u> indow <u>H</u> elp						
😁 • 🔛 🐚   🕲 • 🍕 •   🎋 • 🔕 •   🔗 • 🖾 💋 🖬 👳 🗇 • 🔿 •						
💽 Explorer 🛛 📄 🐨 🗖	✓ ZCU102_XAPP1333 ⋈					- 0
> 🔜 HelloWorld_system [ ZCU102_XAPP1333 ] 🗸 🔄 ZCU102_XAPP1333	type filter text	Board Support Packa	ige			
> 🚡 export > by hw > by log log > by resources > by synamp.fxbl > by synamp.fxbl > by synamp.fxbl > by synamp.fxbl > by synamp.fxbl > by synamp.fxbl	<ul> <li>✓ CUID2 (APP1333)</li> <li>✓ CP pru cortexa33,0</li> <li>✓ Syngmp, fibl</li> <li>✓ Board Support Package</li> <li>✓ Shand Support Package</li> <li>✓ Pru pru, D</li> <li>✓ Pru pru, D</li> <li>✓ Board Support Package</li> </ul>	View current BSP settings, or peripherals, change versions Modify BSP Settings. Rese ASSP settings from file Operating System Name: standalone Version: 7,6 Description: Standalone Documentation standalone Documentation standalone Documentation standalone Documentation standalone	configure settings like STDIO perip of OS/libraries/drivers etc. et BSP Sources with the subsquent changes are app and with the subsquent changes are app the subsquent changes are app the subscription of a source of a hosted envir ev 7 6	heral selection, compiler flags, SW intru- the settings dialog. To use exising setti- lied on top of the loaded settings. er. It provides access to basic processo noment, such as standard input and ou	sive profiling, add/remove librari ngs, click the below link. This oper click the below link. This oper rfeatures such as caches, interrup tput, profiling, abort and exit.	es, assign drivers to ration clears any rts and exceptions
		Drivers Libraries				
		Mana	Driver	Desumentation	Formation	•
Assistant ⊠ E ⊕ ♥ ♥ 0 株 8 □ □		psu_acpu_gic psu_adma_0	scugic zdma	Documentation Link Documentation Link	Import Examples Import Example Import Example Import Examples Import Examples Import Examples	_
→ 🐸 HelloWorld_system [System]		psu_adma_1 psu_adma_2	zdma	Documentation Link	Import Examples	
> ( HelloWorld [Application]		psu_adma_3	zdma	Documentation Link	Import Examples	
🔨 Debug		psu_adma_4	zdma	Documentation Link	Import Examples	
Release		psu_adma_5	zdma	Documentation Link	Import Examples	
ZCU102_XAPP1333 [Platform]		psu_adma_6	zdma	Documentation Link	Import Examples	
		psu_adma_7	zdma	Documentation Link	Import Examples	
						~
	Main Hardware Specification					
$\sim$	main Hardware specification					
	📮 Console 🖾 📳 Problems 📗 Vitis Log 🕦 Guid	dance				4.6
	Build Console [ZCU102_XAPP1333]					
	Nothing to build in platform 'ZCU182_XAPP:	1333.				

Figure 4: Standalone BSP for PUF eFUSE registration

- 3. In the **Supported Libraries**, select **xilsecure** and **xilskey**.
- 4. Click **OK** to close the window.
- 5. Right-click the **ZCU102\_XAPP1333** platform in the **Explorer** view, which is now marked out-of-date, and click **Build Project**.
- 6. Select **Board Support Package** under **standalone on psu\_cortexa53\_0** in the **platform view** that just opened and select **Libraries** tab in **Operating Systems** section.
- 7. Scroll to the bottom of the Libraries tab and click Import Examples for the xilskey library.
- 8. Check the **xilskey\_puf\_registration** example and click **OK**. This adds the associated project to your workspace.
- 9. Open the **xilskey\_puf\_registration.h** file in the **src** folder under the fully expanded **xilskey\_puf\_registration\_example\_1\_system** in the **Project Explorer** tab.
- 10. Change the definition of **XSK\_PUF\_INFO\_ON\_UART** to **TRUE**. This setting is extremely important to verify the PUF registration completed successfully.
- 11. Ensure the definition of **XSK\_PUF\_PROGRAM\_EFUSE** is set to **TRUE**.
- 12. Change the definition of **XSK\_PUF\_PROGRAM\_SECUREBITS** to **TRUE**.
- 13. Change the definition of **XSK\_PUF\_SYN\_WRLK** to **TRUE**.
- 14. Set the XSK\_PUF\_AES\_KEY to the Key 0 value in the aes\_key.nky file.





- 15. Set the **XSK\_PUF\_BLACK\_KEY\_IV** to a value that is user choice. This IV is not related to the IV created in **aes\_key.nky** and can be any user generated value. This IV is used by encryption when encrypting the red key with the PUF's KEK.
- 16. Create a file named **puf\_iv.txt** with the ASCII-HEX string of the PUF IV used in **XSK\_PUF\_BLACK\_KEY\_IV** as this is needed during boot. Alternatively, use the one provided in the design documents in the Keys folder.
- 17. Verify all the required changes are made before continuing as shown in the figure below. The **xilskey\_puf\_registration.h** file with the example keys, shown in the figure below, is included in the reference design in the puf\_registration folder.
- 18. To save changes to **xilskey\_puf\_registration.h** click File -> Save in the main toolbar.

kilskey_puf_registration.h ∞
/***************** Macros (Inline Functions) Definitions *******************/
<pre>/* Following parameters should be configured by user */</pre>
<pre>#define XSK_PUF_INF0_ON_UART TRUE #define XSK_PUF_PROGRAM_EFUSE TRUE #define XSK_PUF_IF_CONTRACT_MANUFATURER FALSE</pre>
/* For programming/reading secure bits of PUF */ #define XSK_PUF_READ_SECUREBITS FALSE #define XSK_PUF_PROGRAM_SECUREBITS TRUE
<pre>#if (XSK_PUF_PROGRAM_SECUREBITS == TRUE) #define XSK_PUF_SYN_INVALID FALSE #define XSK_PUF_SYN_WRLK TRUE #define XSK_PUF_EALSEP_DISABLE FALSE</pre>
#define XSK_PUF_RESERVED FALSE #endif
<pre>#define XSK_PUF_AES_KEY "012345678901234567890123456789012345678901234567890123456789012345678901234567890123" #define XSK_PUF_IV "012345678901234567890123"</pre>
<pre>#define XSK_PUF_REG_MODE XSK_PUF_MODE4K ⊖</pre>

Figure 5: PUF Registration File Required for eFUSE

#### **PUF Registration into eFUSEs**

**IMPORTANT:** THESE INSTRUCTIONS MODIFY THE EFUSES ON THE ZCU102 DEVELOPMENT BOARD AND MIGHT LIMIT FUTURE USE OF THE DEVELOPMENT BOARD FOR TESTING AND DEBUGGING!

To register the PUF into the eFuse, perform the following steps:

- 1. Right-click on the **platform.spr** that is located under ZCU102\_XAPP1333 platform in the **Explorer** view and click **Open**.
- 2. Select **Board Support Package** under **standalone** on psu\_cortexa53\_0 in the ZCU102\_XAPP1333 platform view and click **Modify BSP settings**.





3. In the **Board Support Package Settings** window, expand the Overview tree, then click **standalone** as shown in the figure below.

Board Support Package S Control various settings of yo	<b>ettings</b> our Board Support Package.				
✓ Overview ✓ standalone	Configuration for OS: star	ndalone			
xilffs	Name	Value	Default	Туре	Description
	hypervisor_guest	false	false	boolean	Enable hypervisor guest s
psu_cortexa53_0	stdin	psu_uart_0	none	peripheral	stdin peripheral
	stdout	psu_uart_0	none	peripheral	stdout peripheral
	zynqmp_fsbl_bsp	true	false	boolean	Disable or Enable Optimiz
	microblaze_exceptions	false	false	boolean	Enable MicroBlaze Excep
	enable_sw_intrusive_profi	false	false	boolean	Enable S/W Intrusive Profi
	4		111		
0					Cancel
0					

*Figure 6:* Setting Up the UART Output Using the BSP Settings

- 4. Ensure the **stdin** and **stdout** functions are mapped to psu\_uart\_0 and click **OK**.
- In Xilinx Vitis Explorer view, on the left, right-click xilskey\_puf\_registration\_example\_1\_system and select Build Project.
- 6. Turn power off to the ZCU102 board.
- 7. Connect either the USB JTAG connector J2 to the ZCU102 development board and then a computer or connect the Platform JTAG to the ZCU102 and the associated hardware to a computer.
- 8. Connect a USB cable from the USB Serial port connector J83 on the ZCU102 board to a computer and note which COM port was enumerated with the Silicon Labs Quad CP2108 USB to UART Bridge: Interface 0.
- 9. Open a terminal program such as PuTTY or Tera Term and connect to the COM port listed above at 115,200 baud. Enable terminal logging and select a file name and location.



10. On the ZCU102 development board, set the dip switch SW6 to configure the board for JTAG boot mode as shown in the figure below.



Figure 7: ZCU102 JTAG Boot Mode Switch

www.xilinx.com

11. Power on the ZCU102 board using switch SW1.



12. Right-click **xilskey\_puf\_registration\_example\_1 > Run As > Launch Hardware (Single Application Debug)** as shown in the figure below.

			0	
Explorer 🛛				
> 📰 HelloWorld	system [ ZCU102_XAPP1333 ]			
> 📰 xilskey efu	seps_zynqmp_example_1_system	[ZCU102_XAPP1333]	t.	
🗸 📰 xilskey_puf	_registration_example_1_system	[ZCU102_XAPP1333]		
> C xilskey	ouf registration example 11 star	idalone on psu-cortex	a53_0]	
> 📂 Deb	New	>		
👗 xilsl	Move To System Project			
~ <u>CU10</u>	Paste	Ctrl+V		
> 🗁 hw 💥	Delete	Delete		
> 🅞 logs 🔊	Refresh	E5		
> 🗁 psu	Kenesii			
📂 resc 🚵	Import Sources			
> 📂 tem 🛃	Export as Archive			
> 🔁 zynı	Ruild Droject			
> 🗁 zyni				
V plat	Clean Project			
📄 plat	Generate Linker Scrint			
	C/C + Build Sattings			
	C/C++ build Settings			
	Team	>		
1	Run As	>	E 1 Launch	Hardware (Single Application Debug)
Assistant	Debug As	>	E 2 Launch	SW Emulator (Single Application Debug)
V 🐸 HelloW	Properties	Alt+Enter	E 3 Launch	Hardware (Single Application Debug (GDB))
	* **		Run Conf	igurations
Release				

*Figure 8:* **Running the PUF Registration on the ZCU102 Board** 

The PUF registration application starts running and outputs information to the terminal as shown in the figure below. An example log of the PUF registration is included in the design files in the Logs folder called **puf\_registration\_log.log**.

3	PMU-FW is not running, certain applications may not be supported.
4	App: Example is running on Silicon version 4.0
5	App:PUF Registration Completed:0x00000000
6	App:PUF Syndrome data Start!!!
7	13892C962D76FC55448479EDF35577F1C9DF378C9918374981F28C66D885C8D1C881D1479D49F71D874734982994963FF4F874C44
	RF244F5928F26866653FD3D84D388531749963688443859D888948237325D83469F8F945D48880927818658D81F2935489CF6F96
	1483F9235607168D824588F514FFC9F8D545454701C79181C18D58663C5CDFF8221DF2491C4F522D8817FD71F334358D898D55858
	D389F83F2F3F2F8875F948FC831C444878458F8FF256F46F23F88F39F2848R0FF5151487381D8086D9F2193874C88314FD842377D
	A34264D378648F513C579834885D548F61F8594FCD8346644D3884249C151588888D3D7F944D848F6669784992D444687F64387DC
	3/16/30/ER/241F49166F9(7)9041640/2075477007275F82/041F940868/DF744/24F7F88/21856972868F6F78/9518/70354636688478544
	587273177579845783F57444645756178587275691384786457483187855998460181572806754985678439568
	5/2/1/1/1/2/2/0/2/2/0/2/2/2/2/2/2/2/2/2/2
	30367AU835440005U6/344/20403L04171332E01LF0L37/F4311F006A3A101/0052LE0E/F/0E10E1010E2244/U311331330/03400
	B1<2500/04024054C34C34C34C44C4CC474050C434054C474C474040C4C4039540405354040535274055654544052240
	B12233C395362811252F080004C15CF5C3032EFF53A2340FFF74C444763704C495462F31333C000000000000000000000000000000000
	00000000000000000000000000000000000000
	20000000000000000000000000000000000000
	20020202020202020202020202020202020202
•	App:Pur Syndrome data Endiii
0	Ann. Ally, 00503400
9	App: AUX-006C3AB9
9 10	App: AUX-006C3AB9 App: CHASH -3C5351EF App: ShutterValue -01000055
9 10 11	App: AUX:006C3A89 App: CHASH -3C5351EF App: ShutterValue -0100005E App: Part For a 1:3455780012345578001234557800123455780012345578001234557800123
9 10 11 12 13	App: AUX-006C3A89 App: CHASH -3C5351EF Anp: ShutterValue -0100005E App: Red key - 01234567890123456789012345678901234567890123456789012345678901234567890123
9 10 11 12 13 14	App: AUX-006C3A89 App: CHASH -3C5351EF App: CHASH -3C5351EF App: Red key - 0123456789012345678901234567890123456789012345678901234567890123 App: Black key IV - 012345678901234567890123 App: Lack key IV - 012345678901234567890123
9 10 11 12 13 14	App: AUX-006C3A89 App: CHASH -3C5351EF Aoo: ShutterValue -0100005E App: Red key - 0123456789012345678901234567890123456789012345678901234567890123 App: Black key IV - 012345678901234567890123 App: ULacne olsabled App: DLacne olsabled App: DLacne olsabled
9 10 11 12 13 14 15 16	App: AUX-006C3A89 App: CHASH -3C5351EF Aoo: ShutterValue -0100005E App: Red key - 01234567890123456789012345678901234567890123456789012345678901234567890123 App: Black key IV - 012345678901234567890123 App: U.acne 015301e0 App: DMA config Apo: DMA config Apo: DMA config
9 10 11 12 13 14 15 16 17	App: AUX:006C3A89 App: CHASH -3C5351EF App: CHASH -3C5351EF App: Red key - 01234567890123456789012345678901234567890123456789012345678901234567890123 App: Black key IV - 012345678901234567890123 App: Dcacne clsabled App: DMA config App: DMA config initialize App: DMA config initialize App: App: DMA config initialize
9 10 11 12 13 14 15 16 17 18	App: AUX:006C3A89 App: CHASH -3C5351EF App: ShutterValue -0100005E App: Red key - 01234567890123456789012345678901234567890123456789012345678901234567890123 App: Black key IV - 012345678901234567890123 App: DLache disabled App: DMA config App: DMA config initialize App: AES initialize App: AES initialize App: AES encryption
9 10 11 12 13 14 15 16 17 18 19	App: AUX-006C3AB9 App: CHASH -3C5351EF Ano: ShutterValue -0100005E App: Red key - 01234567890123456789012345678901234567890123456789012345678901234567890123 App: Black key IV - 012345678901234567890123 App: Duacne oisableo App: DMA config App: DMA config App: DMA config initialize App: AES initialize App: AES initialize App: AES encryption App: AES generated
9 10 11 12 13 14 15 16 17 18 19 20	App: AUX-006C3A89 App: CHASH -3C5351EF Aoo: ShutterValue -0100095E App: Rd key - 01234567890123456789012345678901234567890123456789012345678901234567890123 App: Black key IV - 012345678901234567890123 App: Duacne o15abled App: DMA config App: DMA config App: DMA config App: AES initialize App: AES initialize App: AES encryption App: Encrypted key generated App: Black key - AASCEC8506D3959ACB5249A89B9A68D537F667EFD42341CB08647EAE4C3669BD
9 10 11 12 13 14 15 16 17 18 19 20 21	App: AUX-006C3A89 App: CHASH -3C5351EF App: ShutterValue -0100005E App: Red key - 01234567890123456789012345678901234567890123456789012345678901234567890123 App: Black key IV - 012345678901234567890123 App: Utache 01sa01ed App: DWA config App: DWA config initialize App: AES initialize App: AES encryption App: Encrypted key generated App: Black key - AASCEC850603959ACB5249A89B9A68D537F667EFD42341CB08647EAE4C3669BD App: Black key - BLEE
9 10 11 12 13 14 15 16 17 18 19 20 21 22	App: AUX-006C3A89 App: CHASH -3C5351EF App: ShutterValue -0100005E App: Red key - 0123456789012345678901234567890123456789012345678901234567890123 App: Black key IV - 012345678901234567890123 App: DCache disabled App: DMA config App: DMA config initialize App: DMA config initialize App: AES initialize App: AES encryption App: Black key - AASCEC8506D3959ACB5249A89B9A68D537F667EFD42341CB08647EAE4C3669BD App: Programming AFUES
9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	App: AUX-006C3A89 App: CHASH -3C5351EF App: ShutterValue -0100005E App: Red key - 0123456789012345678901234567890123456789012345678901234567890123 App: Red key - 0123456789012345678901234567890123 App: DLacne oisabled App: DLacne oisabled App: DMA config App: DMA config initialize App: AES initialize App: AES encryption App: Encrypted key generated App: Black key - AASCEC8506D3959ACB5249A8989A68D537F667EFD42341C808647EAE4C36698D App: Promatted syndrome data start!!! 962C091355EC762DD790AA4F17758C37DEC5A9371899660CF281D1C0847D181C81DE7A99D983A47873F969C474EBFAE54AF2AB06
9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	App: AUX-006C3A89 App: CHASH -3C5351EF Anp: ShutterValue -0100005E App: Red key - 0123456789012345678901234567890123456789012345678901234567890123 App: Red key - 0123456789012345678901234567890123456789012345678901234567890123 App: DNA config App: DMA config initialize App: DMA config initialize App: AES initialize App: AES initialize App: AES encrypted key generated App: Black key - AA5CEC8506D3959ACB5249A89B9A68D537F667EFD42341CB08647EAE4C3669BD Ann- Brognaming AEUSE App:Formatted syndrome data start!!! 962C091355EC762DED790AA4F17758C37DEC9A9371899660CF281D1C084701B1C81DE7A99D983A47873F969C474EBFAE54AF2AB66 260F923DFD55388D38A3696A917593B4ABB2940035D3237F9E09FA6BD8B0A456510793F281B05ECF8095AE9B31A9610075084592B
9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	App: AUX-006C3A89 App: CHASH -3C5351EF App: CHASH -3C5351EF App: Rd key - 01234567890123456789012345678901234567890123456789012345678901234567890123 App: Black key IV - 012345678901234567890123 App: Duacne 01sa01e0 App: DVA config App: DVA config initialize App: DVA config initialize App: AES initialize App: AES initialize App: AES encryption App: Encrypted key generated App: Black key - AASCEC850603959ACB5249A89B9A68D537F667EFD42341CB08647EAE4C3669BD App: Formatted syndrome data start!!! 962C091355EC762DED790A44F17758C37DEC9A9371899660CF281D1C0847D181C81DE7A99D983A47873F969C474EBFAE54AF2AB06 260F923DFD5538B038A3696A917593B4ABB82940035D3237F9E09FA6BD8DA456510793F2818D6ECFB95AE9831A9610075084502B
9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	App: AUX-006C3A89 App: CHASH -3C5351EF App: CHASH -3C5351EF App: ShutterValue -0100005E App: Red key - 0123456789012345678901234567890123456789012345678901234567890123 App: Black key IV - 012345678901234567890123 App: Ucacne disabled App: DVA config App: DVA config initialize App: DVA config initialize App: DVA config initialize App: AES initialize App: AES initialize App: AES encryption App: Encrypted key generated App: Black key - AASCEC8806D3959CB5249A89B9A68D537F667EFD42341C808647EAE4C3669BD Ann- Programming ASUSE App:Formatted syndrome data start!!! 962C091355EC762DED790AA4F17758C37DEC9A9371899660CF281D1C0847D1B1C81DE7A99D983A47873F969C474EBFAE54AF2AB66 206F923DFD553B8D38A3696A917593B4A8B82940035D327F969FA6BD8BA46551793F2818DE6CFB954E981A9610D75084592B DC9EF14F546A5D5EB91C7058B0C181DE5C3C06E21D22EB52AE1FD178B2D353AE371550099BD9E38BE2F3E2B3945E0FEBC431CB0F5 B8A06F725EF0083EF246A8D298F7246A8D298F2654B31A9617B0750815493C20834270376A7B0364573C51BF52B83F2661E5485C785B
9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	App: AUX-006C3A89 App: CHASH -3C5351EF Ano: ShutterValue -0100005E App: Red key - 01234567890123456789012345678901234567890123456789012345678901234567890123 App: Black key IV - 012345678901234567890123 App: DCacne disabled App: DCacne disabled App: DMA config App: DMA config initialize App: AES initialize App: AES encryption App: Encrypted key generated App: Black key - AASCEC8506D3959ACB5249A89B9A68D537F667EFD42341CB08647EAE4C3669BD App: Prognaming -&EUE App: ACC00000000000000000000000000000000000
9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	App: AUX-006C3A89 App: AUX-006C3A89 App: CHASH -3C5351EF Ano: ShutterValue -0100005E App: Red key - 01234567890123456789012345678901234567890123456789012345678901234567890123 App: Red key - 01234567890123456789012345678901234567890123456789012345678901234567890123 App: DLacne olsabled App: DLacne olsabled App: DLacne olsabled App: DMA config App: DMA config initialize App: AES initialize App: AES encryption App: Encrypted key generated App: Black key - AASCEC8506D3959ACB5249A89B9A68D537F667EFD42341C808647EAE4C3669BD App: Boognoming AFIKE App:Formatted syndrome data start!!! 962C091355EC762DED790AA4F17758C37DEC9A9371899660CF281D1C0847D181C81DE7A99D983A47873F969C474EBFAE54AF2AB06 266F923DFD5538B038A3696A917593B4ABB82940035D3237F9E09FA6BD8BDA456510793F2818D6ECFB95AE9B31A96100750845928 DC9EF14F546A5D5EB91C705B0DC1B1DE5C3C06E21D22EB52AE1FD178B2D353AE371550009BD9E38BE2F3E2B3945E07EBC431CB0F5 BBA86F25EEF0083EF246A8209FF35151F000B17340213F6E0DA8AA2AC37D304H32D537F669E01A9AC027A72F2082EF72DDD89A144E 5938AD6AA6159C28D888154D947E3D69660E0AD492AFA7E60A42AC37D304H32D537F669E01A9AC027A72F2082EF72DDD89A144E 7CD0086CEBF72472098521786E8D30975129846063545844A52B7FL0783F28A2964AAC4EE8E12F6521369CCAF835FC4
9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	App: AUX-006C3A89 App: CHASH -3C5351EF App: CHASH -3C5351EF App: ShutterValue -0100095E App: Black key - 0123456789012345678901234567890123456789012345678901234567890123 App: Black key TV - 012345678901234567890123 App: Dua config App: Dua config App: Outacne 015001e0 App: ACS initialize App: ACS initialize App: ACS initialize App: ACS encryption App: Encrypted key generated App: Black key - AASCEC850603959ACB5249A89B9A68D537F667EFD42341CB08647EAE4C3669BD App: Pomated syndrome data start!!! 962C091355EC762DED790A44F17758C37DEC5A9371899660CF281D1C0847D1B1C81DE7A99D983A47873F969C474EBFAE54AF2AB06 260F923DFD5538B038A3696A917593B4AB8B2940035D3237F9E09FA6BD8BDA456510793F2818D6ECFB95AE9B31A9610D750845028 BD406F25EEF0083EF246A8D269F735151FD0017340219F60D80BAC879342DB442A37D376A7BD364573C51BF5D883E048E54B82C44 5938AD6AA6159C2BD888B14D947E3D09666054585484A4A2C37D304182DE53F966001A9ACD27A72F28DE472DDB9A144E 7CD008CEBF7247209B521786E80307512984603531240044537D40D5780844A2C4EE8E12F65213696CA18528849CCCAF835FC81 BB3DDA68D628E7816C8F497FBCCA39BADFEF9A4A44940873D40D57680844AC4EE8E12F65213696CA18528849CCCAF835FC81
9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	App: AUX-006C3A89 App: CHASH -3C5351EF App: CHASH -3C5351EF App: Red key - 01234567890123456789012345678901234567890123456789012345678901234567890123 App: Black key IV - 0123456789012345678901234567890123 App: Utache 013801ed App: Otache 013801ed App: Otache 013801ed App: Otache 013801ed App: DMA config App: Otache 013801ed App: DMA config initialize App: AES initialize App: AES initialize App: AES encrypted key generated App: Black key - AA5CEC850603959AC85249A89B9A68D537F667EFD42341C808647EAE4C3669BD App: Black key - AA5CEC850603959AC85249A89B9A68D537F667EFD42341C808647EAE4C3669BD App: Bromatted syndrome data start!!! 962C091355EC7620ED790AA4F17758C37DEC9A9371899660CF281D1C0847D181C81DE7A99D983A47873F969C474EBFAE54AF2A806 206F923DFD553B8D38A3696A917593B4A8B82940035D237F9609FA68D8BA45651073F281806ECFB93AE9B31A9610075084592B DC9EF14F546A505EB91C705B00C181DE5C3C06E21D22EB52AE1FD178B20353AE37155009B09E38BE2F3E2B3945E07EC431C80F5 BBA06F25EEF00831EF246A8D209F735151FD00817340219F6DD80BAC873342D8442A37D376A7B03645735151EF5D883E06618E5485C48 5938A06A6159C28D888154D947E30569660E0AD492AFA7E60A42AC37D3041B2DE53F966901A9AC27A72F2082EF72DDB9A144E 7CD008CEBF7247209B521766E8D3075129846063545844A52B7F1D783F28A946AAC4EE8E12F6521369CA1B52B849CCCAF835FC41 BB3DDA6BDC38F736C8F309FDCEA39BADFEF9AAA990B8573D40D5FC80344C42894AA42E8E12F652169CA1B52B849CCCAF835FC41 BB3DDA6BDC38F736C8F309FDCEA39BADFEF9AAA4990B8573D40D5FC80344C428812F652169C78B5210F6017842208547621052C8054422C8772468952206F17652B8349CCCAF835FC41 BB3DDA6BDC38F736C8F39FBCEA39BADFEF9AAA4990B8573D40D5FC80344C42884426868A3C52B71768143D5CEB6C18521100EF7C422
9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	App: AUX-006C3A89 App: CHASH -3C5351EF App: CHASH -3C5351EF App: Red key - 0123456789012345678901234567890123456789012345678901234567890123 App: Black key TV - 012345678901234567890123 App: Ucache disabled App: DMA config App: DMA config initialize App: AES initialize App: AES initialize App: AES encryption App: Black key - AASCEC850603959AC85249A89B9A68D537F667EFD42341C808647EAE4C3669BD App: Black key - AASCEC850603959AC85249A89B9A68D537F667EFD42341C808647EAE4C3669BD App: Black key - AASCEC850603959AC85249A89B9A68D537F667EFD42341C808647EAE4C3669BD App: Black key - AASCEC850603959AC85249A89B9A68D537F667EFD42341C808647EAE4C3669BD App: Bnognamming ABUGE App: Formatted syndrome data start!!! 962C091355EC762DED790AA4F17758C37DEC9A9371899660CF281D1C0847D1B1C81DE7A99D983A47873F969C474EBFAE54AF2A806 260F923DFD553B8D38A3696A917593B4A8B82940035D327F9609FA6BD8BA456510793F2818D6ECFB954E981A9610D750845928 DC9EF14F546A5D5E91C70580B0C181DE5C3C06E21D22E52AE1FD17882D35A4551877360364573C51BF5D883E0618E54883CD4E 5938AD6AA6159C28D088B154D947E3D69660E0AD492AFA7E66A42AC37D3941B2DE53F9669A1A92C027A72F208D2EF32ED3945E07EBC431C80F5 BBA06F25EEF0083EF246AB2089F535151FD00875429F4AC4528F71D783F28A2964AAC4EE8E12F6521369CA18528849CCCAF835FC81 BB3DDA68D628EF72472098521786E8D307512984608354584445287F1D783F28A2964AAC4EE8E12F6521369CA18528849CCCAF835FC81 BB3DDA68D628E7816C8F497FBCEA398BADFEF9AA44908B573D40D5FC80344C6608083C53D9814D274689F2E08F1F6728D63C6060396 4011E043D760E7C9477975489A6DBFA589965D005E724C74160344012253F7FA65CCF86FD17681A1A5CEE0C1851196C72420 05115317DA0050733B53B5C68628D68CC5348C3C18B85485C1BFEC627DE07CAEE78EEE684087D0995DC83560D23295660999B85789
9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	App: AUX:006C3A89 App: CHASH -3C5351EF Ano: ShutterValue -0100005E App: Red key - 01234567890123456789012345678901234567890123456789012345678901234567890123 App: Black key I - 0123456789012345678901234567890123 App: Dcacne disabled App: DCacne disabled App: DCacne disabled App: DMA config App: DMA config initialize App: AES initialize App: AES encryption App: Black key - AASCEC850603959ACB5249A89B9A68D537F667EFD42341CB08647EAE4C3669BD App: Black key - AASCEC850603959ACB5249A89B9A68D537F667EFD42341CB08647EAE4C3669BD App: Brognamig - BEUE App: Formatted syndrome data start!!! 962C091355EC762DED790AA4F17758C37DEC9A9371899660CF281D1C0847D1B1C81DE7A99D983A47873F969C474EBFAE54AF2AB06 260F923DFD553B8D38A3696A917593B4ABB82940035D3237F9E09FA6BD8B0A456510793F2B18D6ECF895AE9B31A9610D75084502B DC9EF14F546A5D5EB1C705B0DC181DE5C3C66E21D22EB52AE1FD178B2D353AE37155000BD9E38BE2F3E2B3945E07EBC431CB08F5 BBA06F25EEF0083EF246AB209FF35151FD0017340219F6D0B8AC879342DB42A37D376A7BD364573C51BF5D8B3E0618E5483CC4E 5938AD6AA6159C2B0888B1540947E3D69660E0AD492AFA7E60A42AC37D3941B2DE53F96690A1A9CC27A72F20B2EF72DDD89A144E 7CD008CEBF7247209B5217B6E8D3075129B46063545844A52B7F1D783F28A2964AAC4EE8E12F6521369CA1B52B849CCCAF835FC81 BB3DDA6BD628E7816C8F497FBCEA39BADFEF9A4A490AB573D4DD5FC80344C608883C53D98140274689F2E0BF1F6728D63C606E394 4011E043D76DE7C94775795489A6DBFAS3965D004D5E724724741C6030440122317FA56CCF86FD176B1A1ASCEE0C185C1100EF7C422 05115917DA0050733053B5C666260A0BFAS3C31898513D4D5FC80344C668883C53D9814027468972E005159176B5C33866099BB5789 2B3A4BBB5C28E2783040CF83E1E8115ACD5605588233FFE673FD78427B427B42D784427829940
9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	App: AUX-006C3A89 App: CHASH -3C5351EF Aoo: ShutterValue -010005E App: Ref key - 01234567890123456789012345678901234567890123456789012345678901234567890123 App: DUacne 0123456789012345678901234567890123 App: DUacne 0123456789012345678901234567890123 App: DUacne 012345678901234567890123 App: AES initialize App: AES initialize App: AES encryption App: Encrypted key generated App: Boognameing AEUSE App: Formatted syndrome data start!!! 962669135556762DE0790A44F177586370EC9A937199660CF281D1C084701B1C81DE7A990983A47873F969C474EBFAE54AF2AB06 266F923DFD5538B038A3696A91759384AB88294003503237F9E09FA68D8B0A456510793F281B0ECF895AE9831A96100750845028 DC9EF14F546A505EB91C705B00C181DE5C3C06E21D22EB52AE1FD178820353AE371550098D9E38BE2F3E2B345607EBC431CB0F5 BBA06F25EEF0083EF246A8209FF35151FD0017340219F60D808AC879342D8442A370376A7BD364573C51BF50883E0618E5483C04E 5938AD6AA6159C2B08881540947E3D96660E40A042AFA7E60A42AC37D304A182DE53F966901A3AC027A72F2082EF720D0B9A144E 7CD008CEBF72472098521786E80307512984606354584445287F1D783F28A2964AAC4EE8E12F6521369CA18528849CCCAF835FC81 BB3DDA66D628E7816C8F497FBCEA39BADFEF9AA4490AB573D40D5FC80344C648883C53D98140274699F228B52169E1551160E7C422 05115917DA005673305385C68628068CC5348C3C18689458C18FE627DD7CAEE78EE68408709050C8536003239666099985789 283A48B85C288253940CF835E1E8115ACD68065898393239FE871FD794977044C7829940 App:Formatted syndrome data End!!!
9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	App: AUX-006C3A89 App: CHASH -3C5351EF App: ShutterValue -0100005E App: Black key - 01234567890123456789012345678901234567890123456789012345678901234567890123 App: Black key IV - 012345678901234567890123 App: Utache 01sa01e0 App: OtAc config App: OtAc config initialize App: AES initialize App: AES initialize App: AES encryption App: Encrypted key generated App: Black key - AASCEC850603959ACB5249A89B9A68D537F667EFD42341CB08647EAE4C3669BD App: Pormatted syndrome data start!!! 962C091355EC762DED790AA4F17758C37DEC9A9371B99660CF281D1C0847D1B1C81DE7A99D983A47873F969C474EBFAE54AF2AB06 206FF14F546A505EB01C705B00C1B1DE5C3C06E21D22EB52AE1FD178B2D353AE371550099BD928B8E275E2B3945607EBC431CB08 DC9EF14F546A505EB01C705B00C1B1DE5C3C06E21D22EB52AE1FD178B2D35AE371550099BD928B8E275E2B3945607EBC431CB0F5 BBA06F25EEF0083EF246AB2e9FF35151FD0017340219F6DD80BAC879342D8442A37D376A7BD364573C51BF5D8B3E0618E54B3CD4E 5938AD6AA6159C2BD888B1540947E3D69660E0AD492AFA7E60A42AC37D3041B2DE53F96690D1A9AC027A72F20B2EF72DDD99A144E 7C0008CEBF7247299B5217B6E8D3075129B46063545944A490B573D4D05FC80434D268B3C53D981402746B9F260B1F67280520660E396 A01E43D76DE7C9475795489AADEFF9AAA490AB573D4D05FC80442C837D3041B2DE53F966901A9AC027A72F20B2EF72DDD99A144E 7C0008CEBF7247299B5217B6E8D30F5129B46063545844452B71D783F28A2964AAC4EEBE12F6521B60CA1B52B49CCAF835FC81 BB3DDA6BD628E7816CE7975795489AADFFF9AAA490AB573D4D05FC80442C837D3041B2DE53F966901A9AC027A72F20B2EF72DDD99A144E 7C0008CEBF724779585116CB79547595489AADFFF9AAA490AB573D4D05FC80442C837D3041B2DE53F966901A9AC027A72F20B2EF72DDD99A144E 7C0008CEBF724772958516CB79547595489AADFFF9AAA490AB573D4D05FC80444C688083C53D981402746B9F2E08F1F6728D636066E396 A911E403D76DE7C9475795489AADFFF9AAA490AB573D4D05FC80434C688083C53D981402746B9F2E085160B76489586 App:Formatted syndrome data End!!! App: Syndrome write successful
9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26	App: AUX-006C3A89 App: AUX-006C3A89 App: CHASH -3CS351EF App: Red key - 01234567890123456789012345678901234567890123456789012345678901234567890123 App: Black key I - 0123456789012345678901234567890123456789012345678901234567890123 App: DCacne disabled App: AES initialize App: AES initialize App: AES encryption App: Encrypted key generated App: Black key - AASCEC8506D3959AC85249A89B9A68D537F667EFD42341C808647EAE4C3669BD App: Promatted syndrome data start!!! 962C091355EC762DED790AA4F17758C37DEC9A9371899660CF281D1C0847D181C81DE7A99D983A47873F969C474EBFAE54AF2AB06 266F923DFD5538B038A3666A917593B4A8B8294003503237F9609FA65D80BA456518793F281806ECF895A50981A9613755018F50883E691E6431C08F5 BBA06F25EEFP0083EF246A8D209FF35151FD0017340219F60D80BAC87342DB442A3703F6A7BD364573C51BF5D88B26643E54B3CD4E 5938AD6AA6159C28D8888154D947E3D69660E0AD492AFA7E60A42AC37D364182DE53F96690D1A9ACD27A72F2082EF72DDD89A144E 7CD008CEBF72472098521786E8D3075129846063545844A52B7F10783F28A2964AAC4EE8E12F5521396A1B528B49CCAF835FC81 BB3DDA68D628E7816GF497FBCEA39BADFEF9AAA49A8573D04D5F280444C48823579B14D274689F2E0BF1F6728D03236660999B5789 283A48B85C288253940CF83E1EB115AC06B605589A239FFEB71FFDF949770844C7829940 App: Formatted syndrome data End!!! App: Syndrome write successful App: Aux write successful
9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27	App: AUX-006C3A89 App: CHASH -3CS351FF Aoo: ShutterValue -0100005E App: Ref Vey - 012345678901285284561879372818065CF895849831A96180756845828 DC9EF14F546A5D5E891C705800C181DE5C3C06E21022EB52AE1F017882D353AE371550090BD9E38B22F3E2B3945E07E8C431CB0F5 BBA06F25EEF0083E7246A8209FF35151FD0017340219F6DD808AC879342D8442A37D376A78D364573C518F5D883E64385483C04E 5938AD6AA6159C2B0888154D947E3D59660E6AD422A77D96A42A237D36418D521395C1852B849CCCA7835FC81 BB3DA6BA6159C2B0888154D947E3D59660E6AD422A57D064A42C37D304182205376A7BD364573C518F5D8838466035884452B7F1078728A2964AAC4EE812F6521369C1852B849CCA7835FC81 BB3DA6BA628E7816C8F497FBCEA398ADFEF9A4A490A8573D4DD5FC80344C680803553098140274689F2E08F1F6728D63C6662596 4911E643D760E7C947575489A5D8FA589850D4D5E724C741c603e4012E33F17FA55CCF86FD17681A1A5CEE0C185C1180EF7C422 65115917DA0D5073865385C606228068C5348C318FE6371FD0754E78EE56840870905D5C8536D0323968609998B5789 283A48B85C2882583940CF83518E115AC0666558893251FE677F0744C7829940 App:Formatted syndrome data End!!! App: Max write successful App: Aux write successful App: Aux write successful App: CMA5H write successful
9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28	App: AUX-006C3AB9 App: CHASH -3C5351EF Acc: ShutterValue -0100005E App: Retkey - 01234567890123456789012345678901234567890123456789012345678901234567890123 App: Black key - 0123456789012345678901234567890123456789012345678901234567890123 App: DMa config App: DMa config App: DMa config App: Max config App: AES initialize App: AES initialize App: AES encryption App: Encrypted key generated App: Black key - ASCEC650603959ACB5249A89B9A68D537F667EFD42341CB08647EAE4C3669BD App: Black key - ASCEC650603959ACB5249A89B9A68D537F667EFD42341CB08647EAE4C3669BD App: Black key - ASCEC650603959ACB5249A89B9A68D537F667EFD42341CB08647EAE4C3669BD App: Start key - ASCEC650603959ACB5249A89B9A68D537F667EFD42341CB08647EAE4C3669BD App: Start key - ASCEC650603959ACB5249A89B9A68D537F9607EFD42341CB08647EAE4C3669BD App: Start key - ASCEC650603959ACB5249A89B9A68D537F9607EFD42341CB08647EAE4C3669BD App: Start key - ASCEC650603959ACB5249A89B9A68D537F9607EFD42341CB08647EAE4C3669BD CSCF14F546A5D5EB0127058B051B10E5C3C66E21D22EB52AE1FD178B2D353AE371550009B03E3BE2F3E2B3945E07EBC431CB0F5 BBA06F25EEF0083EF246AB209FF35151FD0017340219F60D808AC879342DB442A37D376A7BD364573C51BF508B3E0618E54B3CD4E S938A06A6159C28D88B81540947E3065660E4A0492AF71D783F28A2964AAC4EE8E12F6521369CA1B52B849CCAF835FC81 BB30DA6A6159C28D88B81540947E3065960E4A92AF71D783F28A2964AAC4EE8E12F6521369CA1B52B849CCAF835FC81 BB30DA6A6159C28D7838053B65082B030F3129B460635589A29314D5FC68344C680803C53093140274689F228B2136C2AF835FC81 BB30DA6B0528E7816C8F497F8CEA398ADFFF3AA499A83573D40D5FC68344C680883C5398140274689F228B5136CCAF835FC81 BB30DA6B0528E7816C8F497F8CEA398ADFF4574C741C603844012233F17FA56CCF86FD17681A1A5CEE0C185C1100EF7C422 e53148B85C288253940CF381E8115AC068605589A239FFEB71FD784277B44C7823940 App: Syndrome write successful App: Syndrome write successful App: CMASH write successful App: CMASH write successful App: CMASH write successful App: Black key writing is passed
9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29	App: AUX-006C3AB9 App: CHASH -3C5351EF Aco: ShutterValue -0100005E App: Red key - 0123456789012345678901234567890123456789012345678901234567890123 App: Dtacne oisableo App: Dtacne oisableo App: Dtacne oisableo App: MA config App: MA config App: MA config App: MA config App: AES initialize App: AES initialize App: AES encryption App: Encrypted key generated App: Black key - AISCEC8506D3959AC85249A8989A68D537F667EFD42341C808647EAE4C36698D App: Black key - AASCEC8506D3959AC85249A8989A68D537F667EFD42341C808647EAE4C36698D App: Formatted syndrome data start1!! 962C091355C762DE709AAAF17758C37DEC9A9371899660C7281D1C0847D181C81DE7A99D983A47873F969C474EBFAE54AF2A806 260F923DFD55388D38A3696A91759384A8B82940035D3237F9E09FA68D88DA456510793F2818D6ECF895AE9831A9610D750845028 DC9EF14F546A5D56891C70550B0C181DE5C3C06621D228E52AE1FD17882D353AE371550009B09E388E2F3E283945E07E64331C80F5 B8A06F25EEF0083EF246A8209FF35151FD0017340219F60D808Ac87934208442A37D376A78D3045373C518F5D883E6483C04E 5938A06AA6159C280B888154D947E305966064D492AFA7E60A42AC37D304182DE53F966901A9AC027A72F2082EF72D0D89A144E 7CD008CEBF72472098521786E8D3075129846063545844A5287F1D783F28A2964AAC4EE8E12F6521369CA18528849CCCAF835FC81 B8DD0A6B0628E7816C6F497FBCEA398AD05F72472609A52139E17FA56C4283F1D783F28A2964AAC4EE8E12F651369CA18528849CCCAF835FC81 B8DD0A6B0628E73816C8F497BECEA398AD9A5712069609E7247260945274274689F2268F116728D63C6662596 265115917DA0D50F7334F58A6D09D5F72472609A52127F28E33F17FA56CCF88F176F31A5CEE6C185C1100EF7C422 265115917DA0D50F3365385C6B628D68CC5348C318889458C18FE627DED7CAEE78EEE64887D09D50C8536D32396860998B5789 283A48B85C28823594DCF83E1E8115ACD68605589A239FFEB71FFDF94977044C7829940 App: Formatted syndrome data End!!! App: Aux write successful App: Aux write successful App: Aux write successful App: Aux write successful App: CHASH write successful App: Slack key writing is passed
9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 21 22 23 24 25 26 27 28 29 30	App: AUX-006C3A89 App: AUX-006C3A89 App: CHASH -3CS351EF App: Red key - 0123456789012345678001 App: ULacne orsabled App: MA config initialize App: AES initialize App: AES incrypted key generated App: Black key - AASCEC659603959ACB5249A8989A680537F667EFD42341CB08647EAE4C36698D App: Dengeneming AEUGE App: Formatted syndrome data start!!! 962C091355EC762DED790AA4F17758C37DEC9A3371899666CF281D1C0847D181C81DE7A990983A47873F969C474EBFAE54AF2A8B6 266F923DFD5538B03A369GA91753984A8B1294003503237F9E09FA6B08BDA45510793F2818D6ECFB95AE98149610753045282 DSC9F14F546A5D5EB91C7058B00E1BDE5C3C066E12022E552AEF1071882D353AE371550099B01384C73E283945687EBC431C80F5 B8A06F25EEF0083EF246A8209FF35151FD0017340219F60D80BAC879342D8442A37D376A7BD364573C51BF5D883E6618E5483CD4E 5938A06AA6159C28D88881540947E3D69660E0A0492AFA7E60A42AC37D3044182DE53F9669001A9ACD27A72P2082EF72DD0D89144E 7CD008CEBF72472098521786E8D3075129846063548445287F1D783F28A2964AAC4E8E12F6521369CA1822B449CCAF835FC81 B81D0A6B0528E7816C8F497FBCEA398ADEFF9AA4390A8573D4DD5FC8934FC888253D931402746B9F268E12F67215085C80682538 4011E043D76DE7C9475795489A6D87A58905004D5E724C7416G3944012E33F17FA56CF86F017681A1A5CEBC185C11046EF7C422 05115917DA00597385385C6B628068C5348C3C18089488C18FE67DD7DC7CAEF8EEE6840870905058356D032396860998B5789 283A488B5C288253940CF851EB115AC056805589A239FFEB71FFD7949770444C7829940 App: Formatted syndrome data End!!! App: Syndrome write successful App: Max write successful App: GC ccalculated on black key is = 25D1361C Black key writing is passed App
9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	App: AUX-006C3A89 App: CHASH -3CS351EF App: Red key - 012345678912434586980 App: OMA config initialize App: AES encrypted key generated App: Incrypted key generated App: Incrypted key generated App: Incrypted key generated App: Formatted syndrome data start!!! 962C091355EC762DED790AA4F17758C37DEC9A9371899660CF281D1C0847D181C81DE7A990983A47873F969C474EBFAE54AF2AB66 266F923DFD5538B038A3696A91759384A8B823400350327F969FA68D08DA4556139725818065CF895A56981396100750845928 DC9EF14F546A5D5EB91C709580DC1B1DESC3C06E21D22EB52AE1FD17882035A8E3715500908D983882F3E289345EF786431C80F5 88A66F25EFF0683EF7346A209F731515FD001734490A857300546739342D842A370376A7B0344573C51BF508838618E5485C4F 5938A06AA6159C2D088881549947E3D5966060A992AFA7E60A42C370384182DE53F966901A9AC027A72F2082EF72D0D9A144E 7CD080EC8F7247298521786E80307512984663545844A5287F1D783F28A2964AAC428E12F6521369CA18528849CCC4F385FC81 8830DA6B05C88F316C64206628058623468445287F1D783F28A2964AAC48E8E12F6521369CA18528849CCC4F385FC81 B830DA6B05C88F316C6429FAFCA5A989A5F304A49A8457305045FC8344668883C35D89149A027A692F2805360032396860998B5789 283A48B85C288253940CF8351E8115AC068605589A239FFE071FFDF94977044C7829940 App: Kux write successful App: Mark weite successful App: Rack key writing is passed App: Crc caclculated on black key is = 2501361C Black key writi
9 10 11 12 13 14 15 16 17 18 20 21 22 23 24 25 26 27 28 29 30 13 20 31 32	App: AUX-006C3A89 App: CHASH -3C5351EF App: Black key IV - 01234567890123456782000 App: DMA config App: AES initialize App: AES initialize App: Encrypted key generated App: Encrypted key generated App: Formatted syndrome data start!!! 962C091355EC762DD790AA4F17758C37DEC9A9371899660CF281D1C0847D181C81DE7A99D983A47873F969C474E8FAE54AF2A8B6 C96F214F546A5D5EB91C705B80C1B1DE5C3C66E21D22E852AE1FD17882D35AE371550098D9E38BE2F3E3B34A65187584592B DC9EF14F546A5D5EB91C705B80C1B1DE5C3C66E21D22E852AE1FD17882D353AE371550098D04573518F50883E0818548228 DC9EF14F546A5D5EB91C705B80C1B1DE5C3C66E21D22E852AE1FD17882D353AE371550098D04573518F50883E0818548228 DC9EF14F546A5D5EB91C705B80531575096660E0A0492AFA7E60A42A37D376A7BD36A573C51BF50883E0818548228 D5938AD6AA6159C2B088815409475153151P00017340219F60D808AC879342D8442A37D376A7BD36A573C51BF50883E08185483C04E 5938AD6AA6159C2B0888154D947515945605354844A5287F1D783728A2964AAC4E8E12F6521369CA18252849CCCAF835FC1 B8D0A6B0528E7816C8F497F8CEA3980DFF59804055274C741C60394412233F17FA56CCF88FD1658F1A65248870990508336003239686999885789 283A48885C288253940CF83E1E8115AC068605589A239FFE871FD794477829940 App: CMASH write successful App: CHASH write successful App: Writing eFUSE PUF secure bits SUCCess
9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 <b>32</b>	App: AUX-006C3A89 App: CHASH -3C5351EF App: Red key - 01234567890123456780012345678001234567800123456780012450000000000000000000000000000000000

Figure 9: Terminal Output Registering PUF to eFUSEs

- 13. Verify line 12 of the UART output is the Red Key that was configured in **XSK\_PUF\_AES\_KEY** in **xilskey\_puf\_registration.h**.
- 14. Verify line 13 of the UART output is the Black Key IV that was configured in **XSK\_PUF\_BLACK\_KEY\_IV** in **xilskey\_puf\_registration.h**.
- Line 20 of the UART output is the **Black Key** generated by the AES encryption engine using the PUF as a KEK.



- Line 21 shows that the **Black Key** was burned into eFUSEs.
- Line 23 of the UART output is the required syndrome data that the PUF uses to regenerate its device-unique encryption key. It is the data that is being programmed into the eFUSEs.
- Lines 31 shows that the PUF information has been burned into eFUSEs.
- 15. Power off the **ZCU102** development board.

#### **RSA eFUSE Configuration**



**IMPORTANT:** THESE INSTRUCTIONS MODIFY THE EFUSES ON THE ZCU102 DEVELOPMENT BOARD AND MIGHT LIMIT FUTURE USE OF THE DEVELOPMENT BOARD FOR NON-SECURE TESTING AND DEBUGGING!



**IMPORTANT:** Programming any of the RSA\_EN eFUSE settings preclude Xilinx test access. Consequently, Xilinx might not accept return material authorization (RMA) requests.

#### **RSA eFUSE Settings**

RSA eFUSE registration is covered in detail in *Programming eFUSEs for AES and RSA Cryptographic Functions* in the *Programming BBRAM and RSA\_EN eFUSEs* [Ref 3], so only a summary pertaining to this application note is covered here.

- 1. Right-click on the **platform.spr** that is located under **ZCU102\_XAPP1333** platform in the **Explorer** view and click **Open**.
- 2. Select **Board Support Package** under **standalone on psu\_cortexa53\_0** in the **platform view** that just opened and select **Libraries** tab in **Operating Systems** section.
- 3. Scroll to the bottom of the libraries tab and click Import Examples for the **xilskey** library.
- 4. Check the **xilskey\_efuseps\_zynqmp\_example** project and click **OK**. This adds the associated project to your workspace.
- 5. Open the **xilskey\_efuseps\_zynqmp\_input.h** file in the src folder under the fully expanded xilskey\_efuseps\_zynqmp\_example\_1\_system in the **Project Explorer** tab.
- 6. Change the definition of **XSK\_EFUSEPS\_RSA\_ENABLE** to **TRUE**. This permanently forces the use of RSA authentication.
- 7. Change the definition of **XSK\_EFUSEPS\_PPK0\_WR\_LOCK** to **TRUE**. This prevents any modifications to the PPK0 hash stored in eFUSEs.

www.xilinx.com

The first set of settings are shown in the figure below:



6	)/**		
	* Follo	owing is the define to select if	the user wants to program
	* Secur	re control bits	
	*/		
	#define	XSK_EFUSEPS_AES_RD_LOCK	FALSE
	#define	XSK_EFUSEPS_AES_WR_LOCK	FALSE
	#define	XSK_EFUSEPS_ENC_ONLY	FALSE
	#define	XSK_EFUSEPS_BBRAM_DISABLE	FALSE
	#define	XSK_EFUSEPS_ERR_DISABLE	FALSE
	#define	XSK_EFUSEPS_JTAG_DISABLE	FALSE
	#define	XSK_EFUSEPS_DFT_DISABLE	FALSE
	#define	XSK_EFUSEPS_PR0G_GATE_DISABLE	FALSE
	#define	XSK_EFUSEPS_SECURE_LOCK	FALSE
	#define	XSK_EFUSEPS_RSA_ENABLE	TRUE
	#define	XSK_EFUSEPS_PPK0_WR_LOCK	TRUE
	#define	XSK_EFUSEPS_PPK0_INVLD	FALSE
	#define	XSK_EFUSEPS_PPK1_WR_LOCK	FALSE
	#define	XSK_EFUSEPS_PPK1_INVLD	FALSE
	#define	XSK_EFUSEPS_LBIST_EN	FALSE
	#define	XSK_EFUSEPS_LPD_SC_EN	FALSE
	#define	XSK_EFUSEPS_FPD_SC_EN	FALSE
	#define	XSK EFUSEPS PBR BOOT ERR	FALSE

Figure 10: Settings for RSA Authentication When Using eFUSEs - 1

- In the next section of the configuration, change the definition of XSK\_EFUSEPS\_WRITE\_PPK0\_HASH to TRUE.
- 9. Change the definition of **XSK\_EFUSEPS\_PPK0\_HASH** to the value stored in **sha3.txt** that was created by bootgen (or copied form the Keys directory) from the previous section.

The second set of settings are shown in Figure 11. These settings using the examples keys are included in the design files in the **xilskey\_efuseps\_zynqmp\_input.h** file in the rsa\_registration folder. The second RSA authentication key (PPK1) is not written for this application note but it can be done by changing the value of XSK\_EFUSEPS\_PPK1\_WR\_LOCK and XSK\_EFUSEPS\_PPK1\_HASH.

10. To save changes to **xilskey\_efuseps\_zynqmp\_input.h** click **File -> Save** in the main toolbar.



⊖/** * Following is the define to select if t * User Fuses, PPK0 Sha3 hash, PPK1 Sha3	he user wants to select AES key, hash and SPKID for Zyng MP
*/ /* For writing into eFuse */ #define XSK_EFUSEPS_WRITE_AFS_KEYF	ALSE
#define XSK_EFUSEPS_WRITE_PPK0_HASH #define XSK_EFUSEPS_WRITE_PPK1_HASHF #define XSK_EFUSEPS_WRITE_SPKIDF	ALSE
<pre>#define XSK_EFUSEPS_WRITE_USER0_FUSE #define XSK_EFUSEPS_WRITE_USER1_FUSE #define XSK_EFUSEPS_WRITE_USER3_FUSE #define XSK_EFUSEPS_WRITE_USER4_FUSE #define XSK_EFUSEPS_WRITE_USER5_FUSE #define XSK_EFUSEPS_WRITE_USER6_FUSE #define XSK_EFUSEPS_WRITE_USER7_FUSE</pre>	FALSE FALSE FALSE FALSE FALSE FALSE FALSE
<pre>&gt;/** * Following defines should be given in t * The length of AES_KEY string must be 6 * SHA3/SHA2 selection and and for USER EU </pre>	the form of hex string. 4, PPK hash should be 96/64 based on ISES SPK ID must be 32
<pre>#define XSK_EFUSEPS_AES_KEY "00000000 #define XSK_EFUSEPS_PEKA_TS_SHA3TPUE</pre>	00000000000000000000000000000000000000
#define XSK EFUSEPS PPK0 HASH "38B29D67	0EECD2678105A9C51B203F50A0557B0614B33

Figure 11: Settings for RSA Authentication When Using eFUSEs - 2

#### **Programming RSA eFUSEs**

Program the RSA eFUSEs by performing the following steps:

- 1. Right-click on the **platform.spr** that is located under **ZCU102\_XAPP1333** platform in the **Explorer** view and click **Open**.
- 2. Select **Board Support Package** under **standalone on psu\_cortexa53\_0** in the **ZCU102\_XAPP1333** platform view and click **Modify BSP settings**.
- 3. In the **Board Support Package Settings** window, expand the **Overview** tree and then click **standalone**, as shown in Figure 4 in step 2 of PUF Registration into eFUSEs.
- 4. Ensure the stdin and stdout functions are still mapped to psu\_uart\_0 and click OK.

- In Xilinx Vitis Explorer view, on the left, right-click xilskey\_efuseps\_zynqmp\_example\_1\_system and select the Build Project option. This may have already been completed if your SDK environment is set up to build automatically.
- 6. Power off the **ZCU102** board.



- 7. Connect either the USB JTAG connector J2 to the **ZCU102** development board and then a computer or connect the **Platform JTAG** to the **ZCU102** and the associated hardware to a computer.
- 8. Connect a USB cable from the USB Serial port connector J83 on the ZCU102 board to a computer and make note of which COM port was enumerated with the *Silicon Labs Quad CP2108 USB to UART Bridge: Interface 0.*
- 9. Open a terminal program such as PuTTY or Tera Term and connect to the COM port listed above at 115,200 baud. Enable terminal logging and select a file name and location.
- 10. On the **ZCU102** development board, set the dip switch **SW6** to configure the board for **JTAG** boot mode as shown in Figure 7.
- 11. Power on the **ZCU102** board using switch **SW1**.
- 12. Right-click xilskey\_efuseps\_zynqmp\_example\_1 > Run As > Launch on Hardware (Single Application Debug).
- 13. The RSA eFUSE application starts running and outputs information to the terminal as shown in Figure 12. An example log of the writing the RSA eFUSEs is included in the design files in the *Logs* folder called **write\_rsa\_enable\_log.log**.
- 14. Verify line 15 from the output terminal matches the SHA3 output that was generated and stored in the **sha3.txt** file.
- 15. Notice that line 32 from the terminal matches the SHA3 output that was generated and stored in **sha.txt** file.
- Line 32 confirms that RSA authentication is enabled and now required for use because this was burned into the eFUSEs.
- Line 33 shows that the PPK0 eFUSE has been programmed and the PPK0 SHA3 value cannot be changed.

www.xilinx.com

16. Power off the **ZCU102** development board.



	Viling Tune MD First Stage Reat Landon
	Allinx 2010 MF FIRST Stage boot Loader
	Release 20112 Dec 14 2021 - 19:19:20
	Pho-PW is not running, tertain applications may not be supported.
	Usen Tuse 20000000
é	
10	
11	
12	
13	
14	
15	PPK8-388290678FFCD267818549C518283558485578861483358F688469D31C8C867FF7C38D287F678D2F4FFF6D2D8F49F7DF
16	
17	PPX 1 - 0000000000000000000000000000000000
18	
19	Sokid 0000000
20	
21	Secure and Control bits of eFuse:
22	
23	AFS key CRC check is enabled
24	Programming AES key is enabled
25	All boots must be encrypted with eFuseAFS key is disabled
26	BBRAM key is not disabled
27	Error output from PMU is enabled
28	Jtag is enabled
29	DFT is enabled
30	PROG GATE feature is enabled
31	Rebot from JTAG mode is enabled
32	RSA authentication is enabled
33	Locks writing to PPK0 efuse
34	Revoking PPK0 is disabled
35	writing to PPK1 efuses is not locked
	Revoking PPK1 is disabled
37	LBIST is in disabled state
	PBR boot error halt is disabled
	Zeroization of registers in Low Power Domain (LPD) during boot is disabled
40	Zeroization of registers in Full Power Domain (FPD) during boot is disabled
41	
42	User control bits of eFuse:
43	Programming USER_0 fuses is enabled
44	Programming USER_1 fuses is enabled
	Programming USER_2 fuses is enabled
	Programming USER_3 fuses is enabled
47	Programming USER_4 fuses is enabled
	Programming USER_5 fuses is enabled
	Programming USER_6 fuses is enabled
50	Programming USER_7 fuses is enabled
51	Reserved 1 bits are not programmed on eFUSE
52	Reserved 2 bits are programmed on eFUSE
53	
54	Successfully ran ZynqMP eFuse example

Figure 12: Terminal Output While Writing the RSA Settings to eFUSEs

## **PUF Encryption and Decryption**

#### **PUF Encryption Decryption Demo Application**

The PUF can now be used for encrypting and decrypting user data because the ZCU102 development board has been provisioned. Specifically, this section uses a reference design to show how to encrypt and decrypt user generated AES keys that are stored on an SD card.



- To support the SD card storage the **xilffs** library has to be added to application BSP. Right-click on the **platform.spr** that is located under **ZCU102\_XAPP1333** platform in the **Explorer** view and click **Open**.
- 2. Select **Board Support Package** under **standalone on psu\_cortexa53\_0** in the **ZCU102\_XAPP1333** platform view and click **Modify BSP settings**.
- Select xilffs library in the Board Support Package Settings window. Click on the xilffs library that appears on the left in Overview -> standalone and set the enable\_exfat configuration parameter to true. Click OK.

These settings are shown in the following figure:

┥ Board Support Package Se	ttings					×
Board Support Package Se Control various settings of yo	<b>ttings</b> ur Board Support Package.					
<ul><li>✓ Overview</li><li>✓ standalone</li></ul>	Configuration for library: xilffs					
xilsecure	Name	Value		Default	Туре	Description
xilskey	enable_exfat	true	~	false	boolean	0:Disable exFAT, 1:Enable exFAT()
XIIITS	enable_multi_partition	false		false	boolean	0:Single partition, 1:Enable multi
v unvers	fs_interface	1		1	integer	Enables file system with selected
psu_contexabs_0	num_logical_vol	2		2	integer	Number of volumes (logical driv
	read_only	false		false	boolean	Enables the file system in Read_(
	set_fs_rpath	0		0	integer	Configures relative path feature
	use_chmod	false		false	boolean	Enables use of CHMOD function
	use_lfn	0		0	integer	Enables the Long File Name(LFN
	use_mkfs	true		true	boolean	Disable(0) or Enable(1) f_mkfs fu
	use_strfunc	0		0	integer	Enables the string functions (vali
	word_access	true		true	boolean	Enables word access for misaligr
	> ramfs_size	3145728		3145728	integer	RAM FS size

*Figure 13:* Configuring Board Support Package Settings in Xilinx Vitis – Standalone Library Configuration

- 4. Right-click the ZCU102\_XAPP1333 platform in the **Explorer** view, which is now marked as out-of-date, and click **Build Project**.
- 5. In Xilinx Vitis, click **File > New > Application Project**. If **Create a New Application Project** window appear click **Next**.
- 6. Select ZCU102\_XAPP1333 platform in the Platform Window and click Next.
- 7. Type in **ExternalKeyStorage** in the Application project name:
- 8. Leave remaining parameters at their default value and click Next.
- 9. Leave the domain as **standalone on psu\_cortexa53\_0**. These settings are shown in Figure 14, Figure 15, and Figure 16.
- 10. Select Next.
- 11. Select Empty Application (C).
- 12. Click Finish.
- 13. Expand the src folder in **ExternalKeyStorage** of the Project explorer window.
- 14. Right-click src and select Import Sources.





- 15. Click **Browse** in the File system window.
- 16. Navigate to the ExternalKeyStorage/src folder in the reference design file directory and check all ".c" and ".h" files and then click **Finish** as shown in Figure 17.

Select a platform from repos	ory 📋 Create a new	platform from hardware (XS	(A)		
nd:					🕂 Add  🏘 Manage
lame	Board	Flow	Vendor	Path	
TCU102_XAPP1333 [custor	] zcu102	Embedded SW Dev	xilinx	C:\Us	ers\kkepa\projects\vitis_2021.2_workspace_ZCU102\Z
latform lofo					
latform Info General Info		Acceleration Resources			Domain Details
latform Info General Info Name: ZCU102 XAPP	33	Acceleration Resources The selected platform doe	is not have applicat	ion ^	Domain Details Domains
latform Info General Info Name: ZCU102_XAPP Part: xczu9eg-ffvb11	33 5-2-e	Acceleration Resources The selected platform doe acceleration capabilities	is not have applicat	ion ^	Domain Details Domains Domain name Details
latform Info General Info Name: ZCU102_XAPP Part: xczu9eg-ffvb11 Family: zynquplus	33 5-2-e	Acceleration Resources The selected platform doe acceleration capabilities	is not have applicat	ion	Domain Details Domains Domain name Details standalone on psu_corte CPU: psu_cortexa53_
latform Info General Info Name: ZCU102_XAPP Part: xczu9eg-ffvb1 Family: zynquplus Description:	33 5-2-e	Acceleration Resources The selected platform doe acceleration capabilities	is not have applicat	ion	Domain Details Domains Domain name Details standalone on psu_corte CPU: psu_cortexa53_
latform Info General Info Name: ZCU102_XAPP Part: xczu9eg-ffvb11 Family: zynquplus Description: ZCU102_XAPP1333	33 6-2-e	Acceleration Resources The selected platform doe acceleration capabilities	is not have applicat	ion	Domain Details Domains Domain name Details standalone on psu_corte CPU: psu_cortexa53_

Figure 14: Creating the ExternalKeyStorage Project in Xilinx Vitis – Platform Selection



olication Project Details					••••
ecify the application project name and its system pro	ject properties				
oplication project name: ExternalKeyStorage					
Create a new system project for the application or s	select an existing one from the System project details	workspace 🚺			
HelloWorld_system xilskey_efuseps_zynqmp_example_1_system xilskey_puf_registration_example_1_system	System project name: Target processor	ExternalKeyStorage_system			
Create new	Select target processor	r for the Application project.			
	Processor	Associated applications			
	psu_cortexa53_0	ExternalKeyStorage			
	Channell and a second				
	Snow all processors in	the hardware specification	U		

Figure 15: Creating the ExternalKeyStorage Project in Xilinx Vitis – System Selection



	-		
ect the domain that the application would link to or te: New domain created by this wizard will have all th	create a new domain	late selected in the next step	
elect a domain	Domain details		
standalone on psu_cortexa53_0			
Create new	Name:	standalone_domain	
	Display Name:	standalone on psu_cortexa53_0	
	Operating System:	standalone 🗸 🗸	
	Processor	psu_cortexa53_0	

Figure 16: Creating the ExternalKeyStorage Project in Xilinx Vitis – Domain Selection





X26396-031422

Figure 17: Importing Files from the Reference Design into the ExternalKeyStorage Project

- 17. Create a new file called **ExternalKeyStorage.bif** in the **ExternalKeyStorage** folder. This file is also included with the design files and can be copied into the project folder but the paths must be updated to point to the correct folders. Manual creation of the BIF file is necessary to use the Black Key during boot as the Create Boot Image tool within Xilinx Vitis does not currently support this feature. Future revisions of Xilinx Vitis may support this feature.
- 18. Update the contents of the file to the contents shown in the following figure using the correct paths.

#### AMDA XILINX

1	<pre>//arch = zynqmp; split = false; format = BIN</pre>
	the_ROM_image:
	[pskfile]/Keys/psk0.pem
	[sskfile]/Keys/ssk0.pem
	[auth_params] spk_id = 0; ppk_select = 0
	[keysrc_encryption] efuse_blk_key
	[bh_key_iv]/Keys/puf_iv.txt
	[fsbl_config] puf4kmode, shutter = 0x0100005E, opt_key
	[bootloader, destination_cpu=a53-0, encryption = aes, authentication = rsa, aeskeyfile =/Keys/multiple_keys.nky]
11	/ZCU104/zynqmp_fsbl/fsbl_a53.elf
12	[destination_cpu=pmu, authentication = rsa]
13	/ZCU104/zynqmp_pmufw/pmufw.elf
14	[destination_cpu = a53-0, encryption = aes, authentication = rsa, aeskeyfile =/Keys/multiple_keys_app.nky]
15	./Debug/ExternalKeyStorage.elf
16	

Figure 18: ExternalKeyStorage.bif File

- 19. Build the ExternalKeyStorage project in Xilinx Vitis.
- 20. From the command prompt in the ExternalKeyStorage folder run the following command: bootgen -p zcu9eg -arch zynqmp -image ExternalKeyStorage.bif -w -o BOOT.bin
- 21. Power off the **ZCU102** board.
- 22. Copy BOOT.bin to a blank SD card.
- 23. Load the SD card into the J100 SD slot on the ZCU102 development board.
- 24. Connect a USB cable from the USB Serial port J83 on the ZCU102 board to a computer and make note of which COM port was enumerated with the Silicon Labs Quad CP2108 USB to UART Bridge: Interface 0.
- 25. Open a terminal program such as PuTTY or Tera Term and connect to the COM port listed above at 115,200 baud. Enable terminal logging and select a file name and location.

#### AMD**7** XILINX



X26403-031522

Figure 19: ZCU102 SD Boot Mode Switch Setting

- 26. On the **ZCU102** development board, set the dip switch **SW6** to configure the board for SD boot mode as shown in the previous figure.
- 27. Load the SD card into the **J100 SD** slot on the **ZCU102** development board.
- 28. Power on the **ZCU102** board using switch **SW1**.

In the terminal program, a menu appears as shown in the following figure:



Figure 20: Main Menu of External Key Storage Demo

29. Press **1** to encrypt a user key and to save the encrypted key to the external SD card and follow the prompts, as illustrated in Figure 21.





- a. Enter a 96-bit IV. **Please note: Do not reuse IV.** Per the AES-GCM standard the IV should be a new one per every use.
- Enter an 8-bit key ID. Use an ID of 42 for this key. An ID of 0 is mapped to user eFUSE 0 bit 0, an ID of 1 is mapped to user eFUSE 0 bit 1, ..., an ID of 255 is mapped to user eFUSE 7 bit 31.
- c. Enter a 256-bit AES key.
- d. Enter a file name including a file extension (for example, Key1.key) for the key up to 16 characters long and then press enter when complete.
- 30. After entering the file name, the program displays the unencrypted key blob which consists of the IV, Key's ID, and the key itself. Afterwards, the ID and AES key are encrypted using the PUF's device-unique KEK, the entire 61 byte encrypted key blob is displayed, and the entire encrypted key blob is written to the SD card.
- 31. Repeat the entire encryption process and encrypt another key and new IV (as per AES-GCM standard), using step 29. However, select an ID that is equal to  $0 \times FF$  and create a unique key file name (e.g., Key2.key).
- 32. Power off the **ZCU102** board.
- 33. Remove the SD card and insert the card into a SD card reader on a computer.
- 34. Using a browser or the command line, display the contents of the SD card.
- 35. Make sure both key files generated in step 29 and step 31 appear on the SD card as shown in Figure 22.



#### AMDA XILINX

PuTTY × Xilinx Zyng MP First Stage Boot Loader Release 2021.2 Dec 14 2021 - 19:15:26 PMU Firmware 2021.2 Dec 14 2021 19:16:17 PMU\_ROM Version: xpbr-v8.1.0-0 KKDBG: Build Dec 15 2021, 19:51:42 XILINX PUF-BASED USER KEY ENCRYPTION DEMO Encrypt key (2) Decrypt key Enter a 96 bit IV as 24 hex digits : 012345678901234567890123 Enter an 8 bit Key ID as 2 hex digits : 42 Enter the AES key: 0123456789012345678901234567890123456789012345678901234567890123 Enter a file name up to 16 characters : keyl.key Unencrypted Key Blob IV: 012345678901234567890123 KEY ID: 42 KEY: 0123456789012345678901234567890123456789012345678901234567890123 Encrypting the ID and AES key with the PUF's device-unique key... ENCRYPTED KEY BLOB: 012345678901234567890123E97E8AA7E85BB7FCE9BCC18AFDB8865D159045015C0127E9E6E C5CC86ED8E19F3CA1A650B09CEB410D32E987D36428C363A8062E Writing the key blob to the SD card... Key successfully written to the SD card! (1) Encrypt key (2) Decrypt key

X26405-031522

#### Figure 21: External Key Storage Encryption

Name	Date modified	Туре	~	Size	
BOOT.bin	12/15/2021 7:54 PM	BIN File		411 KB	;
📄 key1.key	1/1/2010 12:00 AM	KEY File		1 KB	;
Key2.key	1/1/2010 12:00 AM	KEY File		1 KB	;

Figure 22: Directory Contents of the SD Card after Writing the Encrypted Key

36. Open both keys in a hex editor and confirm that they match the encrypted key blobs displayed in the user application. KEY1.KEY is shown in the following figure and matches the output generated in Figure 21.



🔝 key1.key

^	Decoded text	0E	0C	0A	08	06	04	02	00	Offset(h)	
	.#Egt.#Egt.#é~Ч	8AA7	E97E	0123	6789	2345	8901	4567	0123	00000000	
	è[ üé⊷ÁŠý,†]E.	4501	1590	865D	FDB8	C18A	E9BC	B7FC	E85B	00000010	
	\.'éæì\ÈnØáŸ<;¦₽	A650	3CA1	E19F	6ED8	5CC8	E6EC	27E9	5C01	00000020	
	°œëA.2é‡Ód(Ãc¨	062E	63A8	28C3	D364	E987	0D32	EB41	B09C	00000030	
~											
6407-031522	X2										

Figure 23: Encrypted Key Data Stored in KEY1.KEY Read from the SD Card

- 37. Remove the SD card from the computer and insert the card into the **ZCU102** development board.
- 38. Apply power to the **ZCU102** development board. The menu shown in Figure 20 appears.
- 39. Press **2** to decrypt the data that is stored externally on the SD card.
- 40. Type in the name of the key file and the file extension used in step 29 (Key1.key).
  - a. The key is read from the SD card and placed into OCM for processing.
  - b. The encrypted key blob is displayed.
  - c. The decryption process of the key blob takes place and the decrypted information is displayed showing the IV, key ID, and key.
  - d. The decrypted GCM tag is compared to the GCM tag stored in the encrypted key blob and the software indicates if they match.
  - e. Lastly, the key ID is mapped to and compared to the associated bit stored in the user eFUSEs and the software indicates if the IDs match. In this case, the IDs match. An ID of 0 is mapped to user eFUSE 0 bit 0, an ID of 1 is mapped to user eFUSE 0 bit 1, ..., an ID of 255 is mapped to user eFUSE 7 bit 31.
- 41. Repeat the process and decrypt the second key that was created in step 29.

www.xilinx.com

42. All of the same information from step 40 is displayed and the key is decrypted and passes authentication. However, the software simulates ID 255 being revoked and should not be used. When ID 255 is read from a decrypted key file, the software replaces the actual value read in from User eFUSE 7, 0x0000\_0000, with a simulated value of 0x8000\_0000. Since bit 31 of User eFUSE 7 is now set and appears to be burned, this simulates ID 255 as being revoked. Decrypting the two test keys is shown in the following figure.

#### AMDA XILINX

COM8 - PuTTY × Xilinx Zynq MP First Stage Boot Loader Release 2021.2 Dec 14 2021 - 19:15:26 PMU Firmware 2021.2 Dec 14 2021 19:16:17 PMU\_ROM Version: xpbr-v8.1.0-0 KKDBG: Build Dec 15 2021, 19:51:42 XILINX PUF-BASED USER KEY ENCRYPTION DEMO (1) Encrypt key (2) Decrypt key Enter a file name up to 16 characters : Keyl.key Encrypted key successfully read from the SD card. ENCRYPTED KEY BLOB: 012345678901234567890123E97E8AA7E85BB7FCE9BCC18AFDB8865D159045015C0127E9E6EC5CC86 ED8E19F3CA1A650B09CEB410D32E987D36428C363A8062E Decrypted Key Blob IV: 012345678901234567890123 KEY ID: 42 KEY: 0123456789012345678901234567890123456789012345678901234567890123 GCM-AUTHENTICATION PASSED! user eFUSE 2: 0x00000000 (0x0000000) ID is valid! Key blob is safe for use. Encrypt key (2) Decrypt key Enter a file name up to 16 characters : Key2.key Encrypted key successfully read from the SD card. 18B8E2EB5E30AFB1EBD72040B5DFD464A7B2520924FD3BC Decrypted Key Blob KEY ID: FF GCM-AUTHENTICATION PASSED! user eFUSE 7 simulation: 0x80000000 ID is INVALID! Key blob is NOT safe for use! Encrypt key (2) Decrypt key

X26408-03152

Figure 24: External Key Storage Decryption - Decrypting Two Keys and Simulating a Revocation of Key with ID 255



# Ordering

Because of the additional screening required to ensure entropy, Xilinx offers two versions of the PUF, a 128-bit and a 256-bit. In both cases, the KEK length is 256 bits. These devices require special ordering codes (SCD). The PUF is not supported for the standard ordering codes, except for development and evaluation, as there is no assurance that there is sufficient entropy in the KEK. Entropy is measured as described in *Zynq UltraScale+ MPSoC PUF Characterization Report* (RPT236) [Ref 6] which is a Xilinx proprietary report. Contact your local Xilinx FAE or sales person to obtain a copy of the report. Use of the PUF does not require additional licensing fees.

# Conclusion

This application note guides a user on how to use the PUF's device-unique encryption key in conjunction with the AES-GCM hardware in order to encrypt user generated data and store the encrypted data externally. The encrypted data can then be read from external storage and decrypted using the AES-GCM hardware in conjunction with the PUF's device-unique key. In addition, this application note shows how to perform data validation of decrypted data packets by utilizing values stored in the user programmable section of eFUSEs.

## **Reference Design**

Download the reference design files for this application note from the Xilinx website. The table below displays the reference design matrix.

Table 1: Reference Design Checklist

Parameter	Description				
General					
Developer Name(s)	Jim Wesselkamper, Nathan Menhorn, Krzysztof Kepa				
Target Devices	Zynq UltraScale+ devices				
Source code provided?	Yes				
Source code format (if provided)	С				
Design uses code or IP from existing reference design, application note, 3rd party or Vivado software? If yes, list.					
Simulation					
Functional simulation performed	No				
Timing simulation performed?	No				



Parameter	Description					
Testbench provided for functional and timing simulation?	No					
Testbench format	N/A					
Simulator software and version	N/A					
SPICE/IBIS simulations	N/A					
Implementation software tool(s) and version	Vitis 2021.2					
Static timing analysis performed?	No					
Hardware Verification						
Hardware verified?	Yes					
Platform used for verification	ZCU102 evaluation board					

#### **Document Navigator and Design Hubs**

Xilinx<sup>®</sup> Documentation Navigator provides access to Xilinx documents, videos, and support resources, which you can filter and search to find information. To open the Xilinx Documentation Navigator (DocNav):

- From the Vivado® IDE, select Help > Documentation and Tutorials.
- On Windows, select Start > All Programs > Xilinx Design Tools > DocNav.
- At the Linux command prompt, enter docnav.

Xilinx Design Hubs provide links to documentation organized by design tasks and other topics, which you can use to learn key concepts and address frequently asked questions. To access the Design Hubs:

- In the Xilinx Documentation Navigator, click the **Design Hubs View** tab.
- On the Xilinx website, see the Design Hubs page.

*Note:* For more information on Documentation Navigator, see the Documentation Navigator page o the Xilinx website.

## **Appendix A**

# Creating the Zynq UltraScale+ ZCU102 Evaluation Board Hardware Design

- 1. Open Vivado Design Suite.
- 2. In the **Quick Start** tab click **Create Project**.
- 3. Click Next in the Create a New Vivado Project page.



- 4. Enter **ZCU102** in the Project name.
- 5. Enter or select an appropriate working directory in the Project location.
- 6. Click Next on the Project Name page.
- 7. In **Project Type**, select **RTL Project** and uncheck two boxes for **Do not specify sources at this time** and **Project is an extensible Vitis platform**.
- 8. Click Next on the Project Type page.
- 9. Click Next on the Add Sources page.
- 10. Click Next on the Add Constraints (optional) page.
- 11. On the Default Part page, click the **Boards** tab.
- 12. Type in **ZCU102** in the Search.
- 13. Click the Zynq UltraScale+ ZCU102 Evaluation Board.
- 14. Click Next on the Default Part page.
- 15. Click **Finish** on the **New Project Summary** Page and wait while the project is being created.
- 16. In the Project Manager tab located on the left of the **Vivado** workspace, click **IP INTEGRATOR > Create Block Design**.
- 17. When the **Create Block Design** window appears, type in **ZCU102** in **Design name**. Leave everything else set to default.
- 18. Click **OK** and wait while the design is created.
- 19. In the **Diagram** section of the workspace, located on the top right, click the + button to add IP.
- 20. When the Search box appears, type in **ZYNQ**.
- 21. Double-click **Zynq UltraScale+ MPSoC** and wait while the part is added to the design.
- 22. Click Run Block Automation at the top of the Diagram window.
- 23. After the **Run Block Automation** window appears, select **All Automation** and **Apply Board Preset**, click **OK** and wait while the automation takes place.
- 24. Double-click the Zynq UltraScale+ part in the Diagram window.
- 25. Click **Page Navigator > PS-PL Configuration** located on the left of the Zynq UltraScale+ (3.3) window.
- 26. Click **PS-PL Interfaces** located in the PS-PL Configuration window.
- 27. Click Master Interface and uncheck the AXI HPM0 FPD and AXI HPM1 FPD parameters.
- 28. Click **OK** to close the window.
- 29. Pres F6 to validate the design.
- 30. Click **OK** when the **Validate Design** window opens indicating the validation was successful.
- 31. In the **BLOCK DESIGN** window, click the **Sources** tab in the upper left-hand corner.



- 32. Right-click ZCU102 under Design Sources and select Create HDL Wrapper.
- 33. When the **Create HDL Wrapper** window opens, click **Let Vivado manage wrapper and auto-update**, then click **OK** and wait while the sources are created.
- 34. In the **BLOCK DESIGN** window in the upper left corner on the **Sources** tab, expand the ZCU102\_wrapper.
- 35. Right-click **ZCU102\_i: ZCU102** and select **Generate Output Products**.
- 36. Leave the default settings in the **Generate Output Products** window. Click **Generate** and wait while the IP is being generated.
- 37. Click **OK** when the **Generate Output Products** window displays **Out-of-context module run was launched for generating output products**.

#### Exporting the ZCU102 Hardware and Launching Xilinx Vitis IDE

- 1. In the main **Vivado Design Suite** toolbar select **File > Export > Export Hardware**.
- 2. Click Next in the Export Hardware Platform window.
- 3. Select **Pre-synthesis output** and then click **Next** in **Output** window.
- 4. Leave the XSA file name and location default values and click **Next** in **Files** window.
- 5. Click Finish in Exporting Hardware Platform window.
- 6. In the main Vivado Design Suite toolbar select Tools -> Launch Vitis IDE.
- 7. If **Vitis IDE Launcher** window opens then in the Workspace provide appropriate working directory to create **Vitis workspace** and click **Launch**.
- 8. In the Welcome tab click **Create Platform Project** under Project column. If no Welcome tab is present then in the main toolbar click **File -> New -> Platform Project**.
- 9. Type in **ZCU102 XAPP1333** as Platform project name in **Create new platform** window and click **Next**.
- 10. In **Hardware Specification** section of **Platform window** click **Browse** to search the XSA file location.
- 11. In **Create Platform from XSA** window navigate to location that was provided in step 3, select the previously experted XSA file and click **Open**. Back in **Hardware Specification** section of **Platform window** the file should appear in the XSA File selection.
- 12. Leave other settings with default values, and then click **Finish**. After importing the hardware, you should see a project named ZCU102\_XAPP1333 that was automatically created based upon the **ZCU102** evaluation board. The platform is marked *out-of-date* because the software components are not yet built by Vitis.

www.xilinx.com

13. To build the platform in **Vitis** Select the **ZCU102 XAPP1333** platform in the explorer view on the upper, left side and in the main toolbar click **Project -> Build Project**.



# Validate the Hardware and Software with the Hello World Application

- 1. In the main Xilinx Vitis toolbar, select File > New > Application Project. If the Create a New Application Project window appears, click Next..
- 2. Select ZCU102 XAPP1333 platform in the Platform window and then click Next.
- 3. In the **Application Project Details** type in **HelloWorld** in the Application project name. Leave remaining parameters at their default value and click **Next**.
- 4. Leave the domain as standalone on psu\_cortexa53\_0 and click Next.
- 5. On the **Template** page of the **New Project** window, select **Hello World**.
- 6. Click Finish.
- 7. Right-click on the **platform.spr** that is located under ZCU102\_XAPP1333 platform in the **Explorer** view and click **Open**.
- 8. Select Board Support Package Settings under standalone on psu\_cortexa53\_0 in the ZCU102 XAPP1333 platform view and click Modify BSP settings.
- 9. In the **Board Support Package Settings** window expand the **Overview** tree and then click **standalone**.
- 10. Make sure the stdin and stdout functions are mapped to psu\_uart\_0 and click OK.
- 11. Right-click the HelloWorld project and select **Build Project**.
- 12. Connect either the USB JTAG connector J2 to the ZCU102 development board and then to a computer or connect the Platform JTAG to the ZCU102 via J8 and the associated hardware to a computer.
- Connect a USB cable from the USB Serial port connector J83 on the ZCU102 board to a computer and make note of which COM port was enumerated with the Silicon Labs Quad CP2108 USB to UART Bridge: Interface 0.
- 14. Open a terminal program such as PuTTY or Tera Term and connect to the COM port listed above at 115,200 baud.
- 15. On the **ZCU102** development board set the dip switch to configure the board for JTAG boot mode as shown in Figure 7.
- 16. Right-click the HelloWorld project and select Run As > Launch on Hardware (Single).
- 17. Verify that "Hello World" is output on the terminal screen. The hardware and software is properly configured and is now ready for use.

#### AMDA XILINX

## References

- 1. Design Security Lounge
- 2. Zynq UltraScale+ MPSoC Device: Technical Reference Manual (UG1085)
- 3. Programming BBRAM and eFUSEs (XAPP1319)
- 4. Developing Tamper-Resistant Designs with Zynq UltraScale+ Devices (XAPP1323)
- 5. Zynq UltraScale+ MPSoC: Embedded Design Tutorial (UG1209)
- 6. *Zynq UltraScale+ MPSoC PUF Characterization Report* (RPT236). Available in Design Security Lounge.
- 7. NIST Special Publication (SP) 800-38D Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM)

# **Revision History**

The following table shows the revision history for this document.

Section	Revision Summary					
04/12/2022 Version 1.2						
Throughout document	Updated SDK information to Vitis across all content					
Introduction Updated 2 block diagrams						
PUF eFUSE Configuration	Added 1 new figure, replaced 2 and removed 1					
PUF Encryption and Decryption	Added 2 new figures, replaced 6 and removed 2					
Ordering	Updated SDK steps to Vitis steps and removed sub-section Creating the First Stage Boot Loader (FSBL) and Board Support Package (BSP)					
0	05/28/2021 Version 1.1					
Introduction	<ul> <li>Added a note for further clarity about boot header permissibility</li> <li>Added a note about the PUF Key</li> </ul>					
0	06/26/2018 Version 1.0					
Initial Xilinx release.	N/A					

# **Please Read: Important Legal Notices**

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL





WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of Xilinx's limited warranty, please refer to Xilinx's Terms of Sale which can be viewed at https://www.xilinx.com/legal.htm#tos; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx's Terms of Sale which can be viewed at https://www.xilinx.com/legal.htm#tos.

#### AUTOMOTIVE APPLICATIONS DISCLAIMER

AUTOMOTIVE PRODUCTS (IDENTIFIED AS "XA" IN THE PART NUMBER) ARE NOT WARRANTED FOR USE IN THE DEPLOYMENT OF AIRBAGS OR FOR USE IN APPLICATIONS THAT AFFECT CONTROL OF A VEHICLE ("SAFETY APPLICATION") UNLESS THERE IS A SAFETY CONCEPT OR REDUNDANCY FEATURE CONSISTENT WITH THE ISO 26262 AUTOMOTIVE SAFETY STANDARD ("SAFETY DESIGN"). CUSTOMER SHALL, PRIOR TO USING OR DISTRIBUTING ANY SYSTEMS THAT INCORPORATE PRODUCTS, THOROUGHLY TEST SUCH SYSTEMS FOR SAFETY PURPOSES. USE OF PRODUCTS IN A SAFETY APPLICATION WITHOUT A SAFETY DESIGN IS FULLY AT THE RISK OF CUSTOMER, SUBJECT ONLY TO APPLICABLE LAWS AND REGULATIONS GOVERNING LIMITATIONS ON PRODUCT LIABILITY.

© Copyright 2018-2022 Xilinx, Inc. Xilinx, the Xilinx logo, Alveo, Artix, ISE, Kintex, Kria, Spartan, Versal, Virtex, Vitis, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.