XAPP1323 (v1.1) August 30, 2018

Developing Tamper-Resistant Designs with Zynq UltraScale+ Devices

Author: Ed Peterson

Summary

This application note provides anti-tamper (AT) guidance and practical examples to help protect the intellectual property (IP) and sensitive data that might exist within a system enabled by Zynq® UltraScale+™ devices. This protection (in the form of tamper resistance) needs to be effective before, during, and after the Zynq UltraScale+ device has been securely booted with a software image and/or configured with a programmable logic (PL) bitstream. Sensitive data can include the software or configuration data that sets up the functionality of the device logic, critical data and/or parameters that might be included in the boot image (for example, initial memory contents and initial state). It also includes external data that is dynamically brought in and out of the device during post-boot normal operation.

This document summarizes the silicon AT features available within Zynq UltraScale+ devices, explains why these features exist, and provides use cases and implementation details for each feature. This document also provides guidance on various other system-level methods that can be used to provide additional tamper resistance.

By following this application note, you can be assured that you are following the best AT practices available with Zynq UltraScale+ devices and provide a strong hardware root of trust to run your applications. These best practices broadly apply whether the goal is to simply prevent cloning/overbuilding of a commercial design, prevent reverse engineering of a military system's valuable critical technology (CT), or anything in-between.

This application note assumes that you are somewhat knowledgeable and proficient in Zynq UltraScale+ device architecture [Ref 1] and design [Ref 2], [Ref 3], as well as the Vivado® tools flow methodology [Ref 4]. *Solving Today's Design Security Concerns* (WP365) [Ref 5] and *Developing Tamper Resistant Designs with UltraScale and UltraScale+ FPGAs* (XAPP1098) [Ref 6] provide a useful background on the various security threats and solutions for FPGAs and SoCs.

Introduction

Xilinx has been at the forefront of providing FPGA and system-on-a-chip (SoC) AT solutions to its customers for many generations. Zynq UltraScale+ devices continue this trend by including a physical unclonable function (PUF), user-accessible hardened cryptographic blocks, asymmetric authentication, side channel attack protection, and other silicon-based AT features. Additionally, to provide a number of tamper protections post secure boot, Xilinx offers an IP core known as Security Monitor [Ref 7]. Due to certain restrictions, Security Monitor's availability is limited. Contact your local Xilinx representative for details.

Keeping one step ahead of the adversary is a continuous process that involves understanding the existing vulnerabilities and attacks and then developing mitigation techniques (countermeasures) to resist those attacks. Xilinx has a multi-generational commitment and roadmap to secure FPGA and SoC technology in a cost-sensitive manner for the AT-conscious communities which include both commercial and defense markets.

By taking advantage of various Xilinx FPGA/SoC AT features, you can choose how much AT to include with the device design based on program and customer security requirements. AT can be in the form of enabling individual silicon AT features or a combination of these AT features, perhaps tied together by the developer in the PL design and following best practice guidance.

The decision as to how much AT to include primarily depends on the following three factors:

- Value: The perceived value of the IP and the damage it might cause either financially or to national security if it were to become compromised. Certain AT features can be more costly to implement and that cost must be weighed against the value of the technology or data being protected.
- Adversary: Access to the system and the sophistication level and resources available to carry out the attack. For example, can access to the system be prevented by guns, gates, and guards or can it be easily obtained in the open market? Is the adversary a garage-based hacker or a nation-state? The adversary's capabilities could be at these extremes or anywhere in-between.
- Design stage: The point in the system development cycle where the decision is made to enable AT for the Zynq UltraScale+ device design. Xilinx highly recommends that the decision to utilize Zynq UltraScale+ device AT features is made very early on (that is, soon after CT is defined in a system) to help address both schedule and cost concerns. It is always more costly, more time consuming, and often less effective to insert AT features later on in the development process.

Another factor that needs to be considered is how much of the processing system (PS)/ programmable logic (PL) resources are consumed by enabling certain AT features. The overall resource penalty is usually rather small. However, it does depend on how these features are implemented and the relative size of the Zynq UltraScale+ device being used (larger devices experience less of an impact to their PL resources).

Xilinx classifies the silicon AT features as either passive or active. In general, passive security features are those that are either part of the tool flow or built into the device and do not require you to do anything extra in your PS/PL design. Passive security features are also temporal in nature—they come into effect at different times during the normal operating life of the Zynq UltraScale+ device.

- Pre-secure boot (for example, black key storage enabled through the PUF)
- During-secure boot (for example, resistance to side-channel attacks through differential power analysis (DPA))
- Post-secure boot (for example, user data protection through disabling of PL readback)

In contrast, active security features are required to be included in the PS/PL design. These features only come into effect after the Zynq UltraScale+ device has been securely booted and

the design becomes active. Examples are, writing to the configuration security unit (CSU) advanced encryption standard (AES) control register to zeroize the battery-backed AES key or handle an unauthorized JTAG event using the JTAG toggle detect feature.

At a bare minimum, you should always plan on including the appropriate passive security features into your design (for example, image/bitstream encryption and authentication). These features do not affect the function of the design. However, they might create logistical challenges (for example, key management), system challenges (for example, a battery is needed if using battery-backed RAM (BBRAM) for key storage), and increase the secure boot time (for example, using public key authentication might increase secure boot time). Otherwise, these features are freely available in terms of impact to the design and can provide a fair amount of tamper protection. For an already fielded system or a design late in the development stage, these AT features are very appropriate for enabling because they do not affect the software or hardware designs.

Additionally, the AT features and guidance presented in this application note fall into three main AT categories:

- Prevention (for example, JTAG port blocking)
- Detection (for example, voltage and temperature monitoring)
- Response (for example, BBRAM key and PL configuration zeroization penalty)

Table 1 summarizes and classifies the built-in silicon AT features of the Zynq UltraScale+ device.

Table 1: **AT Features Classification and Summary**

Zynq UltraScale+ Device Silicon AT Features	Type	Category	Secure Boot/Config Stage(1)
Image/bitstream confidentiality (symmetric)	Passive	Prevention	Pre and During
Volatile on-chip 256-bit BBRAM AES key storage	Passive	Prevention	Pre
Non-volatile on-chip 256-bit eFUSE AES key storage(3)	Passive	Prevention	Pre
PUF-enabled black key storage (internal eFUSEs or external flash storage)(2)	Passive	Prevention	Pre
Write-only key load with integrity check (BBRAM and eFUSE)	Passive	Prevention	Pre
Image/bitstream authentication (symmetric)	Passive	Prevention	Pre and During
Image/bitstream authentication (asymmetric)(2)	Passive	Prevention	Pre
Non-volatile 384-bit eFUSE public key hash storage to enable RSA authentication(3)	Passive	Prevention	Pre
DPA side-channel attack protections	Passive	Prevention	During
Obfuscation of the user AES key loading and storage	Passive	Prevention	Pre
Hardened readback disabling circuitry	Passive	Prevention	Post
Design for test (DFT) boot mode permanent disable (DFT_DIS eFUSE)(3)	Passive	Prevention	Pre, During, and Post
Uninterruptible internal clock source for CSU	Passive	Prevention	During and Post
JTAG port permanent disable (eFUSE)(3)	Passive or Active	Prevention or Response	Pre, During, and Post

Table 1: **AT Features Classification and Summary *(Cont'd)***

Zynq UltraScale+ Device Silicon AT Features	Type	Category	Secure Boot/Config Stage[1]
JTAG port temporary disable	Passive or Active	Prevention	During and Post
JTAG port monitor	Active	Detection	Post
PL configuration memory integrity checking	Active	Detection	Post
Unique identifiers (device DNA and user eFUSE)	Active	Detection	Post
On-chip temperature and voltage monitors/alarms	Active	Detection and Response	During and Post
PL configuration memory clearing	Active	Response	Post
Uninterruptible internal clock source on PL STARTUP block	Active	Detection	Post
Key agility (BBRAM only)	Active	Prevention and Response	Post
BBRAM key zeroize (erase + verify)	Active	Response	Post
CSU tamper monitor and response	Active	Detection and Response	Post
Public key revocation	Active	Response	Post
Non-volatile (eFUSE) tamper event logging	Active	Response	Post
User accessible crypto blocks[2]	Active	Prevention	Post
ARM® TrustZone	Active	Prevention and Detection	Post
ARM v8 cryptography extensions[2]	Active	Prevention and Detection	Post
Xilinx Memory Protection Unit (XMPU)[2]	Active	Prevention and Detection	Post
Xilinx Peripheral Protection Unit (XPPU)[2]	Active	Prevention and Detection	Post
AXI/APB Isolation Block (AIB)[2]	Active	Prevention and Response	Post
System Memory Management Unit (SMMU)[2]	Active	Prevention	Post
Global 3-state (GTS) enable (PL I/O only)	Active	Response	Post
Global set-reset (GSR) enable (PL I/O only)	Active	Response	Post

Notes:

1. Describes when in the Zynq UltraScale+ device normal operating life this feature is effective (pre-secure boot, during secure boot, or post-secure boot).
2. This feature is new or improved in Zynq UltraScale+ devices.
3. Asserting some of these permanent tamper settings or penalties (eFUSE-based) is irreversible and might affect whether or not the device can be returned to Xilinx.

The following sections explore the features described above in some detail, providing explanations on what they are and why they exist. Instructions on how to properly use them either by themselves, in conjunction with other built-in features and user logic, or both are also provided. Additionally, specific board/system level guidance is given on certain methods and techniques that can be employed to increase the tamper resistance of the Zynq UltraScale+ device design and overall system. Due to the complexity of the Zynq UltraScale+ devices, much

of the low-level details needed to implement many of these security features can be found in other existing Xilinx documents. References to these pertinent documents/sections are made throughout this application note.

Any AT features enabled at the Zynq UltraScale+ device level should always be part of an overall system-level AT solution. The features and techniques outlined in this document provide for a very good AT umbrella for the Zynq UltraScale+ device itself. However, AT is most effective when it is developed with a multi-layer approach taking the entire system into consideration.

Zynq UltraScale+ Device Secure Boot

The starting point for a trusted system is assurance that it is secure from the moment power is applied, up to and including the execution of your application design. For more information, see the Boot and Configuration chapter in the *Zynq UltraScale+ MPSoC Technical Reference Manual* (UG1085) [Ref 2] and System Boot and Configuration chapter in the *Zynq UltraScale+ MPSoC Software Developer Guide* (UG1137) [Ref 3]. The Zynq UltraScale+ devices ensure boot-time security by providing various security features. The enabler for a hardware root-of-trust is providing authentication of the software image and bitstream to prove they have not been modified and that they come from a trusted source (starting with the first stage boot loader (FSBL) which is the first piece of user code loaded).

Figure 1 illustrates the typical steps in a secure boot process that maintain a chain of trust assuming each component is either immutable (for example, BootROM code) or is properly authenticated.

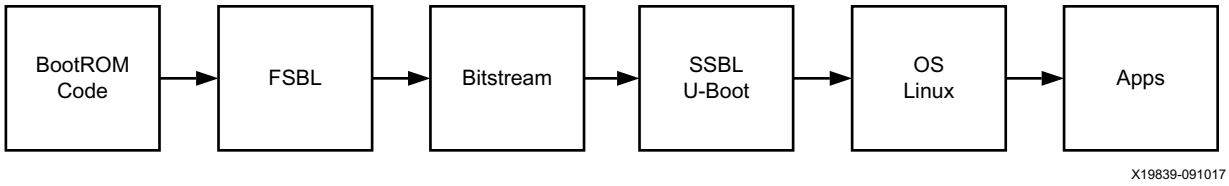

Figure 1: **Secure Boot Chain of Trust**

Along with authentication, the Zynq UltraScale+ device can also provide confidentiality of the image and bitstream to protect against attacks such as cloning, over-building, and reverse engineering. There are a number of other security features such as black key storage and side-channel attack countermeasures that add to the robustness of the Zynq UltraScale+ device secure boot process.

Secure Boot Overview

The Zynq UltraScale+ device provides an independent configuration security unit (CSU) which enables the secure boot process for the Zynq UltraScale+ device. The CSU resides in the low power domain (LPD) and is responsible for security and boot related functions, see *Zynq UltraScale+ MPSoC Technical Reference Manual* (UG1085) [Ref 2]. The CSU is clocked by an internal uninterruptible clock source (SYSOSC).

The CSU is composed of two main blocks as shown in Figure 2. On the left is the secure processor block (SPB) that contains the highly robust triple redundant MicroBlaze™ processor for controlling boot operation. It also contains an associated ROM, a small private RAM, and the necessary control/status registers required to support all secure operations.

The component on the right is the crypto interface block (CIB) and contains the AES-GCM engine, direct memory access (DMA), secure-hash algorithm SHA-3, RSA accelerator, and processor configuration access port (PCAP) interface. After boot, the CSU is used for tamper monitoring and responses, and the crypto blocks (AES-GCM, SHA-3, and RSA) can be accessed and used by an application running in either the PS or PL.

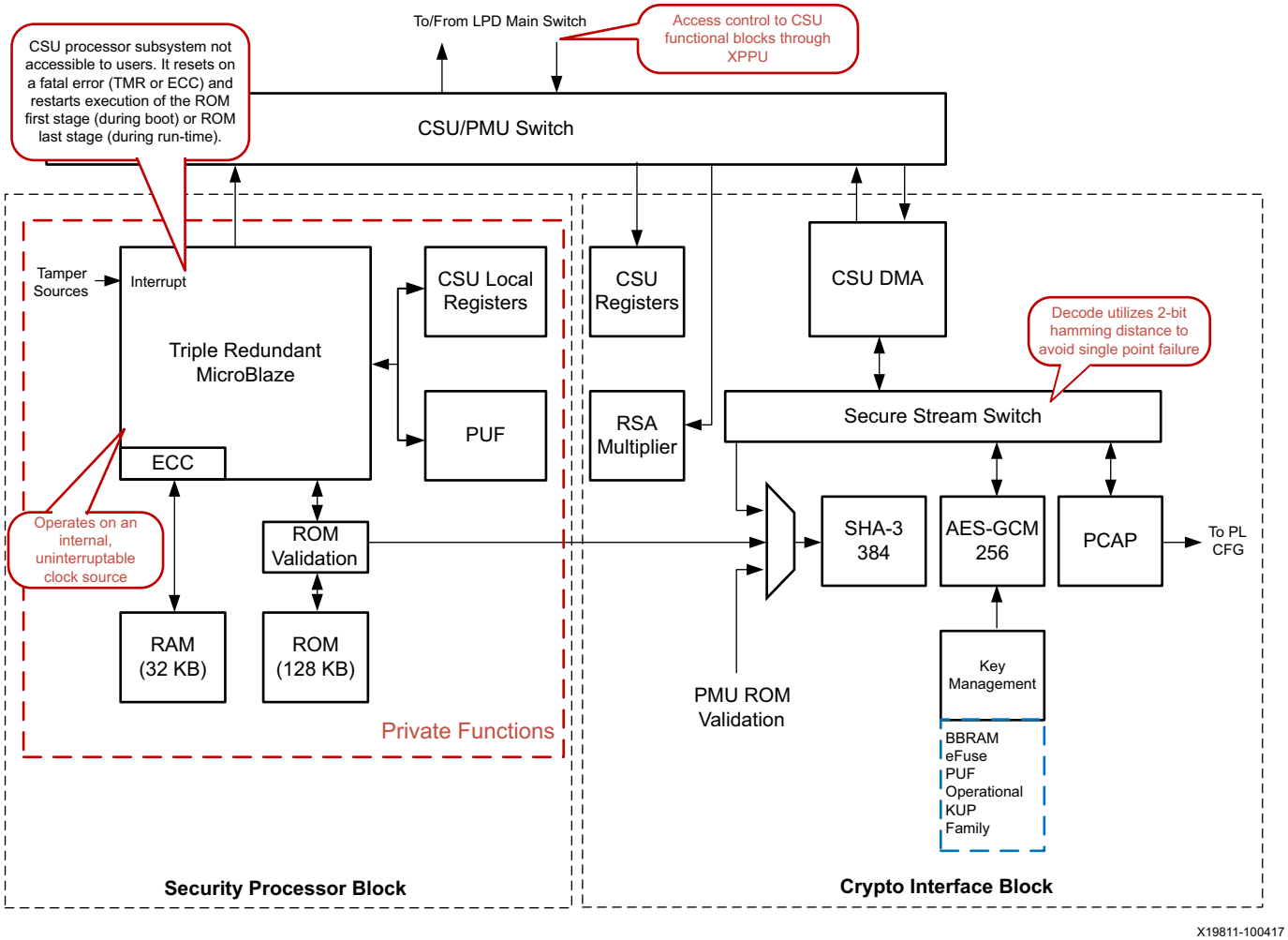

Figure 2: **CSU Block Diagram**

The Zynq UltraScale+ device secure boot process begins with the execution of a hardware state machine. After the hardware state machine completes its checks and functions, it releases the reset to the platform management unit (PMU) so it can start executing its metal-masked boot ROM code (see the Platform Management Unit chapter in the *Zynq UltraScale+ MPSoC Technical Reference Manual* (UG1085) [Ref 2]). Like the CSU, the PMU is a highly robust triple redundant MicroBlaze processor. After the PMU has completed, it performs an integrity check of the CSU ROM and then releases the reset to the CSU. If the integrity check passes, the CSU can start executing its own metal-masked boot ROM code.

The main functions performed by each functional unit are as follows (* signifies optional functions).

1. Hardware State Machine
    a. Test interface lockdown
    b. Zeroize PMU registers
    c. Run logic built-in self-test (LBIST)*
    d. SHA-3/384 integrity check of PMU ROM
    e. Release reset to PMU
2. PMU
    a. Zeroize registers LPD/ full-power domain (FPD)*
    b. Zeroize PMU RAM
    c. Voltage checks (LPD, AUX, I/O)
    d. Zeroize memories on CSU, LPD, and FPD
    e. SHA-3/384 integrity check of CSU ROM
    f. Release reset to CSU
3. CSU
    a. Enforces hardware (HW) root-of-trust when RSA authentication is enabled
    b. Enforces security state (secure boot mode)
    c. Validate integrity of user public key
    d. Public key revocation
    e. Load first stage boot loader (FSBL) and PMU user firmware (FW)
    f. Authentication* and/or decryption*
    g. Zeroize storage elements after processing (including fallback)
    h. Start execution of FSBL

Figure 3 illustrates the Zynq UltraScale+ device secure boot timeline.

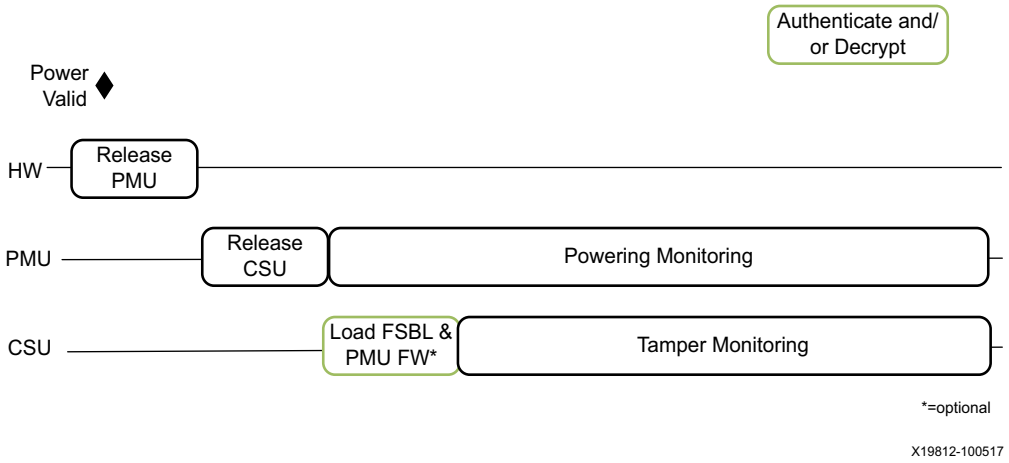

Figure 3: **Secure Boot Timeline**

Post secure boot, you can run your own firmware on the PMU to customize its operation. However, no user code can be run on the CSU. Only the immutable masked ROM code can be executed on the CSU. Post secure boot, the CSU can perform tamper monitoring and response functions which can be enabled through a register interface. For more information, see the Security chapter in the *Zynq UltraScale+ MPSoC Technical Reference Manual* (UG1085) [Ref 2] and *Zynq UltraScale+ MPSoC Register Reference* (UG1087) [Ref 8]. The CSU tamper monitor and response register bits are read/write, set-only (RWSO)—after the bit is set it can only be cleared by a power-on-reset (POR) or soft reset (SRST). Incorrectly written or rogue software cannot reduce a tamper response penalty—the penalty can only be increased.

During the secure boot process there are a number of attacks an adversary can attempt. For Zynq UltraScale+ devices there are many countermeasures that come into play during secure boot to provide a robust hardware root-of-trust. Table 2 provides a summary of these attacks and countermeasures.

Table 2: **Secure Boot Protections Summary**

Attack	Device Countermeasure
Side-channel	Built-in differential power analysis (DPA) countermeasures
Glitching	CSU/PMU triple redundant processors
	ECC on CSU/PMU memories
	Temporal and physical redundancy in HW and ROM code
	SHA integrity checks on immutable ROM code
Focused Ion Beam (FIB) Probing	CSU/PMU triple redundant processors
	ECC on CSU/PMU memories
	Temporal and physical redundancy in HW and ROM code
	SHA integrity checks on immutable ROM code

Table 2: **Secure Boot Protections Summary *(Cont'd)***

Attack	Device Countermeasure
Environmental	Internal run-time voltage check (LPD, AUX, I/Os)
	ECC on CSU/PMU memories
	SHA integrity checks on immutable ROM code
Test Interface Disable	By design (disabled upon power up and fault tolerant)
	JTAG monitoring
	Permanent disable capability
General	Immutable ROM code
	PMU/CSU clocked by internal, uninterruptible clock source
	Pre-boot: the only sensitive information is the device key—protected through PUF

Passive AT Silicon Features

Image/Bitstream Confidentiality and Authentication (Symmetric)

Storing an encrypted image/bitstream in external flash (or other means) and then decrypting it during the secure boot of the Zynq UltraScale+ device (through the device's internal decryption engine) provides for a very high level of confidentiality. This ensures that information contained in the image/bitstream is only accessible to those who share the same (symmetric) secret key. Image/bitstream encryption and decryption provides confidentiality while the system is at rest and during secure boot. It protects the Zynq UltraScale+ device PS and PL design contents, including PL block RAM and flip-flop initialization data. Xilinx highly recommends that the externally stored image/bitstream always be in an encrypted form.

Note: Zynq UltraScale+ devices use the National Institute of Standards and Technology (NIST)-approved AES in Galois/Counter Mode (GCM) with a 256-bit key. The AES NIST CAVP validation posting for the Xilinx® Zynq® UltraScale+ device is available here:
http://csrc.nist.gov/groups/STM/cavp/documents/aes/aesval.html#4438

To take advantage of this security feature, the image/bitstream must first be encrypted using the BootGen software. BootGen is integrated into the Xilinx Software Development Kit (SDK) but is also available as a standalone tool. For more information see the Bootgen Image Creation chapter in the *Zynq UltraScale+ MPSoC Software Developer Guide* (UG1137) [Ref 3]. The BootGen software uses a key supplied by you to perform the encryption. If an AES key is not supplied, the BootGen software also offers the option to generate one automatically. However, keys generated by the BootGen software are pseudorandom. Truly random keys are the most secure and are highly recommended. The key is then loaded into the BBRAM or eFUSEs on the Zynq UltraScale+ device through an application running on the PS. See *Programming BBRAM and eFUSEs* (XAPP1319) [Ref 10].

When image/bitstream decryption is enabled, symmetric authentication is automatically enabled because the chosen method of encryption, AES-GCM, is an authenticated encryption and decryption algorithm. AES-GCM combines the counter mode for confidentiality with an authentication mechanism that is based on a universal hash (authentication tag) function. Therefore, AES-GCM provides not only confidentiality but integrity and authentication at the same time. This cryptographically strong authentication scheme ensures that any attempt at modifying the image/bitstream (even just a single bit) prevents the device from starting up.

If the authentication check passes, the device then begins normal operation and the startup commands take place. Evidence of the authentication step passing can be seen in the aes_status register with the GCM_TAG_PASS bit set to a 1. For more information, see the Security chapter in the *Zynq UltraScale+ MPSoC Technical Reference Manual* (UG1085) [Ref 2]. An authentication failure could indicate the tampering of the image/bitstream. It could also indicate that the channel used for image/bitstream loading is noisy and bit corruption(s) are taking place during the configuration process.

Volatile and Non-Volatile Key Storage

The 256-bit symmetric AES-GCM key can be loaded into either volatile BBRAM or non-volatile eFUSE one-time programmable (OTP) storage locations within the Zynq UltraScale+ device. To decide which storage location to use for the key, an understanding of the advantages and disadvantages of BBRAM (Table 3) and eFUSE (Table 4) storage is necessary.

Table 3: **BBRAM Storage Location: Advantages and Disadvantages**

Advantages	Disadvantages
• Volatile and reprogrammable. • Passive and active key clearing (that is, the evidence can be removed). • Tamper resistant, no physical readback path.	• Requires an external battery. • Many battery vendors do not specify operation at high temperature and/or long lifetimes (although some vendors are now starting to offer betavoltaic type batteries to help address these issues). • Can only store red (plaintext) key in BBRAM.

Table 4: **eFUSE Storage Location: Advantages and Disadvantages**

Advantages	Disadvantages
• No external battery required. • Makes spoofing difficult (would require the device on the board to be replaced). • Can store the key in red (plaintext) or black (encrypted) form (see PUF-enabled Black Key Storage). • Tamper resistant, no physical readback path.	• Key cannot be updated. • Key cannot be cleared.

PUF-enabled Black Key Storage

In Zynq UltraScale+ devices there is a hardened physical (sometimes also called physically) unclonable function (PUF) (see the Security chapter in the *Zynq UltraScale+ MPSoC Technical Reference Manual* (UG1085) [Ref 2]) which is a function that produces a unique signature or fingerprint for a device. The PUF output is different from device to device, even those devices

from the same silicon wafer. In some PUF implementations, as is the case in Zynq UltraScale+ devices, the output (or response) is only known to the device itself and no one else (including the manufacturer).

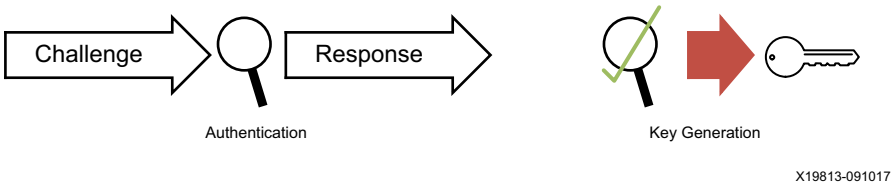

Figure 4: **PUF**

A PUF leverages minute CMOS manufacturing process variations that produce uniqueness between devices, such as threshold voltage, oxide thickness, metal shape, resistances, and capacitances. Xilinx devices are designed to operate within these normal process variations without affecting functionality. For a PUF to be useful and effective it must be able to reliably recreate this unique signature or fingerprint without being impacted by environmental conditions or aging effects.

The PUF in Zynq UltraScale+ devices has been robustly characterized to show high level of entropy from device-to-device (that is, inter-device randomness), stability (that is, intra-device repeatability) and lifetime reliability by performing accelerated aging over environmental extremes (temperature and voltage). See the *Zynq UltraScale+ MPSoC PUF Characterization Report* (RPT236) [Ref 9].

In Xilinx devices prior to the Zynq UltraScale+ device, the symmetric AES-256 key is stored on the device in either the BBRAM or in non-volatile eFUSEs. This key is used in conjunction with the on-chip AES crypto engine to decrypt the encrypted FPGA bitstream or SoC software images. This on-chip storage, while secure, is still susceptible to physical attacks by an adversary because the key itself is stored unencrypted (or red).

The primary use case for the Zynq UltraScale+ device on-chip PUF is to create a key encryption key (KEK). The KEK is used to encrypt the red key prior to storing it on the device in the eFUSEs so that it is stored in a very secure encrypted (or black) form. If an attacker is able to get to the on-chip storage contents, it will be a wasted effort because the key is encrypted and contains no useful information. Also, because the key is encrypted it can optionally be stored in external flash. See *Programming BBRAM and eFUSEs* (XAPP1319) [Ref 10] and *Zynq UltraScale+ MPSoC: Embedded Design Tutorial (*UG1209) [Ref 11].

Figure 5 illustrates the PUF use case for a Zynq UltraScale+ device as a KEK for the user red key.

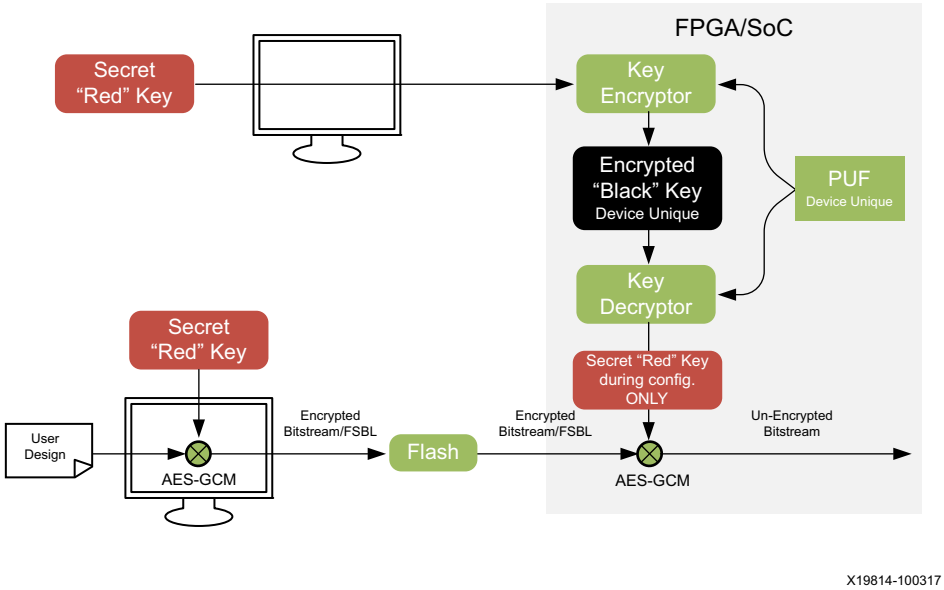

Figure 5: **Zynq UltraScale+ Device PUF Use Case**

The process of encrypting the user's red key and storing it on the device (either in the eFUSEs or externally in the flash memory) is called provisioning or registration. See the appendix for XilSKey library (xilskey_puf_registration) in the *Zynq UltraScale+ MPSoC Software Developer Guide* (UG1137) [Ref 3]. Prior to field deployment, the registration process is undertaken in which the PUF KEK is used to wrap the user red key. The registration process also creates a 4 Kb set of helper data. The helper data is necessary to be able to recreate the PUF's KEK bit-exact each time (even in the presence of noise caused by voltage, temperature, and aging effects) because there is some random noise associated with the PUF regeneration process. The wrapped (or black) key and helper data can then be stored in the eFUSEs or external non-volatile memory (NVM) such as flash memory.

Note: A PUF-encrypted key cannot be stored in the BBRAM. It must be stored in eFUSE or external flash.

After a device has been registered and fielded, the PUF is then used in a process called regeneration. Basically, as part of the secure boot process, the PUF circuitry regenerates the KEK bit-exact so that it can decrypt the black key to retrieve the user's original red key. After decryption, the red key is then used to decrypt the firmware/bitstream and then is zeroized immediately after use. With this scheme, the secret red key can be common across platforms but is stored as a unique black value per device because each PUF KEK is unique.

If the black key and helper data are stored off-chip they are always authenticated (using RSA-4096) by the device prior to use to prevent attacks such as differential power analysis (DPA) and fault injection. In the case of Zynq UltraScale+ devices, the PUF is used primarily for the secure boot process. It can also be used for encrypting small amounts of user application data (for example, user keys). For more information, see *External Secure Storage Using the PUF* (XAPP1333) [Ref 12].

There are competing technologies on how to create a PUF in a silicon device (for example, arbiter, ring oscillator (RO), and SRAM memory). However, it's important to consider the PUF

application when selecting the PUF technology. An important consideration that needs to be made is the type of technology being protected. For example, an SRAM-stored key is best served by protecting it with an RO-based PUF (that is, an asymmetric technology). This is because an SRAM-based PUF is subject to the same attacks as the SRAM storage. The Zynq UltraScale+ device PUF is based upon an RO design.

Figure 6 illustrates this technology ceiling concept for an SRAM-based PUF and a RO-based PUF protecting both an SRAM and eFUSE keys. From Figure 6 it is apparent that using an asymmetric technology for the PUF provides the highest level of relative security because it requires an adversary to master skills required to compromise multiple technologies.

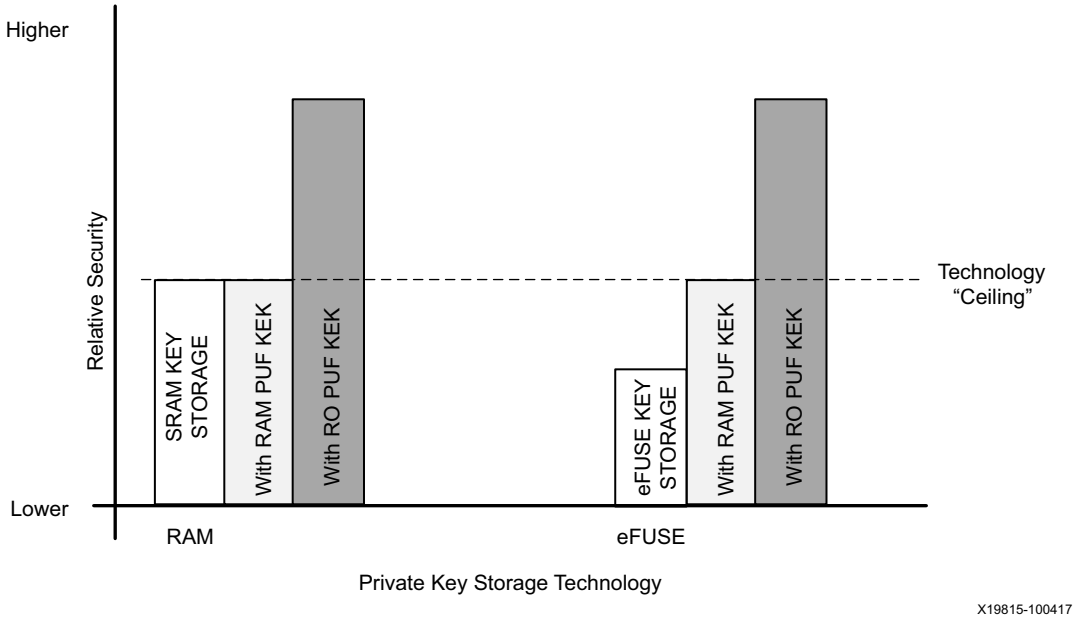

Figure 6: **RAM vs. RO-based PUF**

To ensure sufficient entropy, additional screening is performed by Xilinx. Because of the additional screening required to ensure entropy, Xilinx offers two versions of the PUF: 128-bit and 256-bit. In both cases, the actual 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. There is no assurance that there is sufficient entropy in the KEK for standard ordering codes. Entropy is measured as described in *Zynq UltraScale+ MPSoC PUF Characterization Report* (RPT236) [Ref 9]. Use of the PUF does not require additional licensing fees.

Write-only Key Load with Integrity Check

Both the BBRAM and eFUSE 256-bit symmetric keys are loaded into the BBRAM or eFUSEs on the Zynq UltraScale+ devices through an application running on the PS. See *Programming BBRAM and eFUSEs* (XAPP1319) [Ref 10]. For Zynq UltraScale+ devices, this key loading path is write-only to the device. There is no physical datapath to read back either key. When a key is written to the device through JTAG or internally, a key integrity check is started by writing the expected CRC32 value through JTAG to the device. An actual CRC32 integrity check is calculated

on the stored key by the device (internally) and compared to the expected CRC32 that was just received resulting in a pass/fail response.

Note: For BBRAM-based keys, prior to writing the key, the existing key in BBRAM is zeroized (erased and verified).

Zynq UltraScale+ Device Key Management

Key management in the Zynq UltraScale+ device requires more attention than in previous FPGA/SoC families due to the availability of multiple key sources as well as optional black key storage (Figure 7). To use the AES-GCM engine, one of the key sources must first be loaded into the engine. For more information, see the Security chapter in the *Zynq UltraScale+ MPSoC Technical Reference Manual* (UG1085) [Ref 2] and Programming View of Zynq UltraScale+ MPSoC Devices chapter in the *Zynq UltraScale+ MPSoC Software Developer Guide* (UG1137) [Ref 3]. During boot, the CSU boot ROM selects the source of the keys. After secure boot, software running on the PS can select between the device key and the key update register (KUP).

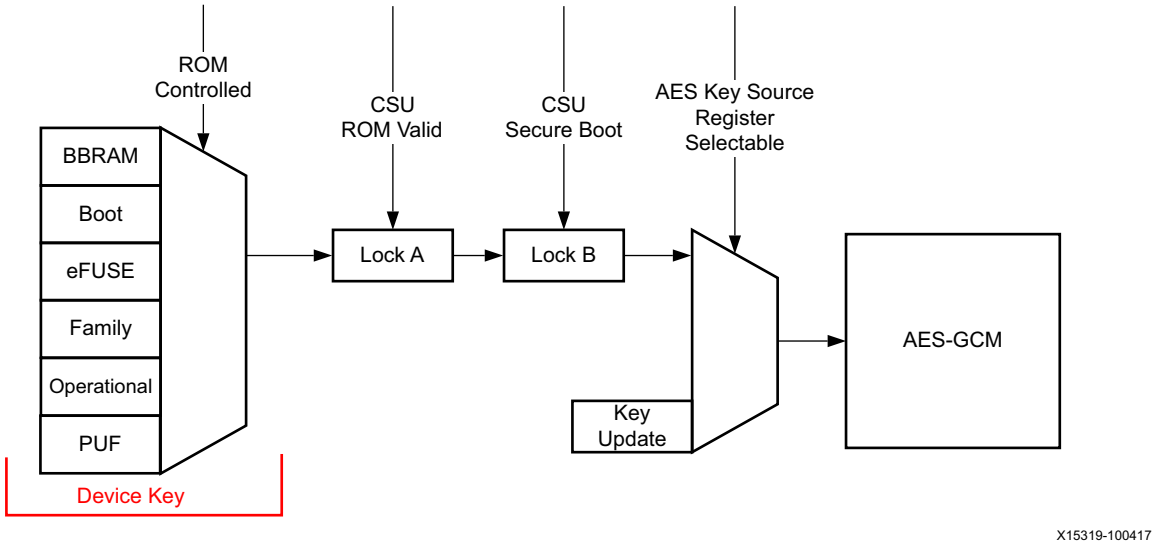

Figure 7: **Key Management**

Device Key

A device key is selected by the CSU ROM at the boot time based on the boot header. Post CSU boot, the device key source cannot be changed until the next POR. A device key is only available when an AES-GCM engine is used in secure boot or with authentication enabled. The authentication-only bit is set in the boot header image attributes and instructs the CSU boot ROM to unlock the device key for later use. No device key is available in non-secure boot (that is, when neither the authentication nor encryption is enabled). The device key location is determined by the encryption status field of the boot header and is selected from one of the sources listed below.

- The BBRAM which holds a 256-bit plain text (red) key.

- The boot key register which holds the decrypted obfuscated key while the key is in use.
- The eFUSE which holds a 256-bit plain text (red) key, an obfuscated key, or a PUF-encrypted key.
- The family key is a constant AES key value hard-coded into the devices. This key is only used while the CSU ROM is in execution to decrypt the obfuscated key. The decrypted obfuscated key is a plain text (red) key used for decrypting the boot images. The obfuscated key can be stored in either the eFUSEs or boot header.
- The operational key is obtained by decrypting the secure header (block 0) using a plain text key obtained from the other device key sources. For a secure boot, this key is optional and not mandatory. The operational key is specified in the boot header and minimizes the use of the device key limiting its exposure.
- The PUF key is a key encryption key (KEK) that can be used to wrap either the internal eFUSE key or the external key in flash.

Key Update Register

The key update register (KUP) is a write-only register. This register is used during boot for the key rolling feature where a different AES key must be loaded multiple times. After boot, any key can be loaded into this register through the advanced peripheral bus (APB) by software running on the PS. A 256-bit KUP key is stored in the eight AES key update registers.

Image/Bitstream Authentication (Asymmetric)

Zynq UltraScale+ devices have the capability to authenticate the entire encrypted or unencrypted image/bitstream before sending it to the on-chip decryption engine (that is, authenticate-then-decrypt). See the Security chapter in the *Zynq UltraScale+ MPSoC Technical Reference Manual* (UG1085) [Ref 2] and the appendix for the XilRSA library in the *Zynq UltraScale+ MPSoC Software Developer Guide* (UG1137) [Ref 3]). If the image/bitstream has been modified in any way (including just a single bit change), the device's asymmetric authentication function detects these change(s) and not only disables the decryption engine but also prevents the startup of the device by entering a secure lockdown mode when the SEC_LK eFUSE is programmed (see the Security chapter in the *Zynq UltraScale+ MPSoC Technical Reference Manual* (UG1085) [Ref 2]). In short, if this feature is enabled (through the RSA_EN eFUSEs), only an authorized image/bitstream can configure the Zynq UltraScale+ device.

This method uses the RSA-4096 asymmetric digital signature (authentication) algorithm and, therefore, it does not require the device to contain a secret to accomplish this authentication task. Instead, the asymmetric authentication function uses user-defined public key information. Due to limited space, the feature stores a 384-bit SHA-3 hash of the 4096-bit public key in the eFUSE bits of the Zynq UltraScale+ device. It is up to you to define the private and public key pairs for this operation. There are a number of open source and commercial products that can be used to create these key pairs (such as OpenSSL and SafeNet). This authentication scheme does not require a secret in the device to operate. Hence, attacks such as side channel analysis do not reveal any information that is useful to an attacker.

The reasons to use the RSA asymmetric authentication are listed below.

- Authenticate the entire image/bitstream before decrypting it. This method is part of a DPA attack countermeasure described in DPA Protections.
- Prevent unauthorized users from ever running their own (potentially malicious) designs on the Zynq UltraScale+ device. After an authorized user programs the public key hash into the eFUSE bits and any of the fifteen RSA-enabled eFUSEs have been programmed (forcing RSA authentication), only authorized images/bitstreams can be loaded.
- Authentication of unencrypted images/bitstreams. A Zynq UltraScale+ device design might not contain sensitive or confidential information but still have requirements that it be authentic. Some example use cases are:
  - The design contains a publicly known function (for example, the AES encryption algorithm). The design does not need to be confidential but you need to ensure that it has not been modified to output red keys or data on external pins, for example.
  - The Zynq UltraScale+ device design has different levels of functionality (for example, basic to advanced features). For instance, only premium-paying end customers have all the features—all others can only access the basic features. The unencrypted bitstream cannot be modified by an adversary to try and turn on any of the advanced features without being detected.

Zynq UltraScale+ devices have the ability to store the 384-bit hashes of two primary public keys (PPKs) in on-chip eFUSEs (the full PPKs are located in the boot image). To limit the use of the primary private/public keys, the PPKs should only be used to authenticate secondary public keys (SPKs) which are located in the boot image itself and then used to authenticate the rest of the boot image/bitstream. Zynq UltraScale+ devices can have up to 32 SPKs. For more information on how a secure boot image is constructed, refer to the Boot and Configuration chapter and Security chapter in the *Zynq UltraScale+ MPSoC Technical Reference Manual* (UG1085) [Ref 2], and the Bootgen Image Creation chapter in the *Zynq UltraScale+ MPSoC Software Developer Guide* (UG1137) [Ref 3].

Starting with Zynq UltraScale+ devices, the public keys associated with the secure boot process have the capability to be revoked. Both PPKs can be revoked (sec_ctrl eFUSE register) and the 32 SPKs can be revoked (spk_id eFUSE register). For more information, see the Security chapter in the *Zynq UltraScale+ MPSoC Technical Reference Manual* (UG1085) [Ref 2] and *Zynq UltraScale+ MPSoC Register Reference* (UG1087) [Ref 8]. If both PPKs are revoked (and any of the RSA_EN eFUSEs are programmed), the Zynq UltraScale+ device becomes permanently inoperable (that is, a brick), which could be used as a severe tamper penalty. Revoking the SPK in a fielded system provides a way to create a temporary tamper penalty for the device until a new image/bitstream is signed with a different valid secondary private key, and the new corresponding SPK ID is programmed into the device.

The entire image/bitstream is authenticated prior to its use. There is a secure boot/configuration time penalty when using the RSA asymmetric authentication scheme (with or without an encrypted image/bitstream). Xilinx provides a boot time estimator spreadsheet to calculate the impact on boot time. See the Zynq UltraScale+ MPSoC - Boot Times Estimation (Xilinx Answer Record 67475) [Ref 13].

DPA Protections

Instead of trying to directly attack a security function (for example, breaking Zynq UltraScale+ device image/bitstream AES-256 decryption using a non-feasible brute-force key attack), an attacker will often look for an easier solution such as side-channel analysis. The side-channel is an unintentional information leakage path that might exist in an electronic device. If observed for a long enough period, it might be possible to extract secret information from the side-channel (for example, a cryptographic function's key data).

Differential power analysis (DPA) is a side-channel technique that observes and records samples of the power supply fluctuations of a functioning electronic device. Signal processing and statistical methods are then applied to the recorded data to extract the AES-GCM key. The number of data samples required has been consistently reduced over time as the capabilities of attackers have improved.

Xilinx provides DPA resistance by limiting the amount of side-channel data that an adversary can collect on any one key. This protocol-based data limiting technique is used on Zynq UltraScale+ devices to mitigate against DPA attacks of the on-chip bitstream decryptor. This technique offers the most long-term flexibility because the level of protection is programmable and can be increased as the capabilities of the attacker improve.

There are two types of data that must be limited in this regard, invalid/random bitstream data and valid bitstream data. To be effective, there must be countermeasures for both invalid/random and valid bitstream data attacks.

Invalid or Random Image/Bitstream Data

Traditionally, a DPA attacker would simply feed an unlimited amount of random data into the decryptor's ciphertext input port and then collect the side-channel information for analysis. Zynq UltraScale+ devices can detect this random (or invalid) image/bitstream by authenticating the encrypted image/bitstream using asymmetric authentication (RSA-4096) before being fed to the decryptor. Side-channel attacks on the signature verification process are useless because there are no secrets to discover (the public key and public key hash can be known by anyone). This method can be used for eFUSE or BBRAM-based AES-GCM keys and it does not automatically incur a tamper penalty (for example, key zeroization).

Valid Image/Bitstream Data

Bitstream lengths on modern FPGAs and SoCs are now large enough for a DPA attack to be attempted on a single valid image/bitstream load (the valid image/bitstream data appears random enough with many input changes). To protect against such an attack, images/bitstreams in Zynq UltraScale+ devices can be broken up into multiple smaller blocks. Each block is encrypted using its own unique user-defined key. The size of each block is programmable and depends on the security requirements. Smaller blocks are more secure because side-channel data available for any one key is less. However, if the blocks become too small, the boot time could increase in some instances.

To avoid having to store all the decryption keys in on-chip memory (BBRAM or eFUSE), Zynq UltraScale+ devices use a key rolling technique where only the initial key (key 0) is stored on-chip. Keys for each successive block are encrypted (wrapped) in the previous block. Figure 8 illustrates this concept.

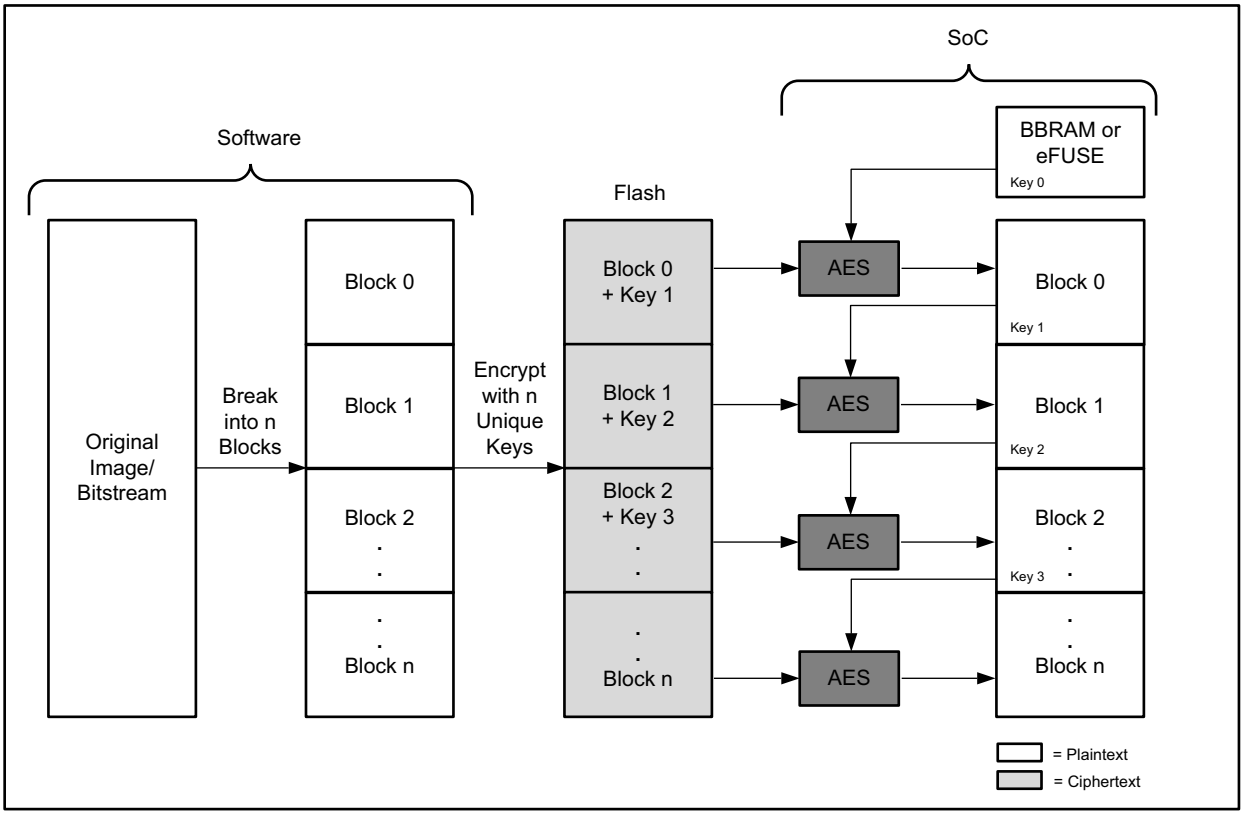

Figure 8: **Key Rolling**

Although it is possible to load a valid image/bitstream an unlimited number of times, additional configurations do not reveal any new side-channel information to the adversary. They can only reduce the signal-to-noise ratio of the side-channel data contained in the first configuration. The key rolling method (along with authenticate-then-decrypt) prevents the adversary from having enough changing values applied into the ciphertext port.

IMPORTANT: *You must use key rolling (valid data attack countermeasure) with the random data attack countermeasure (authenticate-then-decrypt).*

Obfuscated Key Loading and Storage

Optionally, key data written and stored into a Zynq UltraScale+ device eFUSE array can be obfuscated. The key data is encrypted using a fixed family key that is identical for all Zynq UltraScale+ devices, and is known only to Xilinx. (The Zynq UltraScale+ device family key is different from the UltraScale+ FPGA family key). This provides for an increased level of security in commercial production situations (for example, secret red key protection at a contract manufacturer). The internally stored obfuscated key is decrypted at the beginning of an encrypted image/bitstream load and then used to decrypt the image/bitstream that follows it. This feature is enabled in the boot header. Figure 9 provides a high-level summary of this operation. For more information, see the Boot and Configuration chapter and Security chapter in the *Zynq UltraScale+ MPSoC Technical Reference Manual* (UG1085) [Ref 2].

Note: Xilinx does not provide the family key as part of the Vivado or SDK tool sets. To request a family key, send a request to secure-solutions@xilinx.com. Qualified customers will receive the family key through Xilinx Product Licensing.

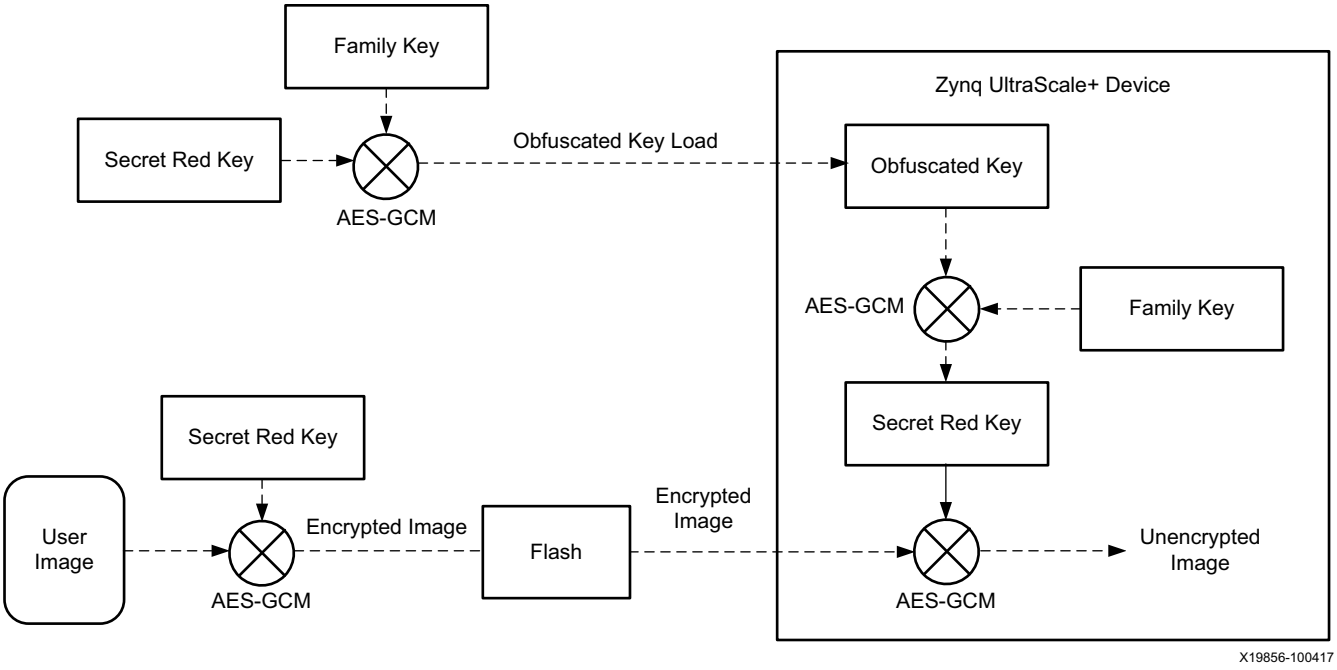

Figure 9: **Summary of Obfuscated Key Loading and Storage**

Hardened Readback Disabling Circuitry

Whenever an encrypted or authenticated bitstream has been loaded into the Zynq UltraScale+ device, readback of the internal PL configuration memory cannot be performed by any external interface (including JTAG). All external readback is automatically blocked (disabled) by a multi-tiered scheme. The only readback access to the configuration memory after an encrypted bitstream load is through the PS processor configuration access port (PCAP) or the PL internal configuration access port (ICAP). Access to the PCAP is controlled by the authenticated software running on the PS, therefore, it is considered a trusted channel. The ICAP is also considered a trusted channel because the PL bitstream is authenticated during the loading process, and it can only be used through a direct connection to the design within the PL. If the PL design does not instantiate the ICAP, it cannot be used at all. A direct connection between the ICAP and user I/O pins through PL logic creates a channel that cannot be trusted and is hence, not recommended.

Design for Test (DFT) Boot Mode Disable (Passive)

DFT boot mode (used only by Xilinx) is always disabled if secure boot (authentication and/or confidentiality) is enabled. Additionally, the DFT_DIS eFUSE can be programmed to permanently disable internal DFT capability.

JTAG Port Disable (Passive)

The JTAG controller can be permanently disabled by programming the JTAG_DIS eFUSE bit in the SEC_CTRL eFUSE register. See the System Test and Debug chapter in the *Zynq UltraScale+ MPSoC Technical Reference Manual* (UG1085) [Ref 2] and *Zynq UltraScale+ MPSoC Register Reference* (UG1087) [Ref 8]. When this eFUSE is blown, the JTAG is restricted to two commands, IDCODE and BYPASS. IDCODE is only available by resetting the JTAG controller (going to the test-logic-reset state). All commands shifted into the instruction register (IR) are converted to BYPASS. Also, when the JTAG disable eFUSE is blown, all security gates are permanently enabled, making it impossible to reach the Zynq UltraScale+ device test access port (TAP) or the ARM core debug access port (DAP).

- IDCODE is available by shifting JTAG to test-logic-reset state.
- BYPASS is available by shifting in any other instruction to the IR.
- All security gates are permanently enabled.

Security Related eFUSEs

There are a large number of security-related eFUSEs on the Zynq UltraScale+ device for tamper logging, storing a black AES key and helper data, red AES key, primary public key hashes, and secondary public key ID. For details refer to the Tamper and Monitor Registers table in the Security chapter in the *Zynq UltraScale+ MPSoC Technical Reference Manual* (UG1085) [Ref 2] and the *Zynq UltraScale+ MPSoC Register Reference* (UG1087) [Ref 8].

Active AT Silicon Features

As mentioned in the Introduction, the active AT features require something in the PS code or PL logic design to take advantage of a particular feature. For example, a logic 1 can be written through an application running in one of the PS processors to the aes_key_clear bit in the CSU AES control register in response to some tamper event (refer to the Security chapter in the *Zynq UltraScale+ MPSoC Technical Reference Manual* (UG1085) [Ref 2]). Table 5 summarizes each of these active features, their use cases, and how the feature can be implemented.

Table 5: **Active Security Features Use Cases Summary**

Feature	Use Case	User How-to
JTAG port permanent disable (eFUSE)	Permanently prevent unauthorized JTAG access in response to a tamper event.	Dynamically program the jtag_dis eFUSE bit in the sec_ctrl eFUSE register.
JTAG port temporary disable	Prevent an unauthorized JTAG access.	Boot device securely (JTAG is disabled by default).
JTAG port monitor	Detect unauthorized JTAG access.	Enable the JTAG toggle detect in the corresponding csu_tamper register.
Configuration memory integrity checking	In-the-background check of configuration memory integrity (non-interfering run-time check).	Instantiate the soft error mitigation (SEM) IP core [Ref 14].
Unique identifiers (device DNA and user eFUSE)	Prevent the design from operating (or operate in a limited manner) if unique identifier is not recognized.	Develop PS code or PL logic to be able to read and process the unique identifier(s) and determine if they are valid.
On-chip temperature and voltage monitor/alarms	Ensure device is operating within normal environmental limits.	Instantiate the PS and PL system monitor (SYSMON) primitives and develop PS code and PL logic to check and respond to environment status (see the *UltraScale Architecture System Monitor User Guide* (UG580) [Ref 16]).
Uninterruptible internal clock source	Ensure active AT functions cannot be disabled by simply removing an external clock source.	Instantiate PL STARTUP primitive, connect to the CFGMCLK output, and use that as the clock source for user-defined AT functions (see the *UltraScale Architecture Libraries Guide* (UG974) [Ref 15]).
PL Configuration memory clearing	Erase the configuration memory in response to a tamper event.	Develop PS code to determine the proper conditions for sending an IPROG command through the PCAP.
Key agility (BBRAM only)	Update the BBRAM key securely in the field without having to return the board or module to a secure facility.	Develop PS code or PL logic that can perform a secure key exchange in logic in response to a key management event and load the new BBRAM key through the PS.
BBRAM key zeroize (erase + verify)	Zeroize the battery-backed key in response to a tamper event.	Develop PS code to determine the proper conditions for clearing the AES key through the aes_key_clear register and for reading the verification bit in the aes_status register.

Table 5: **Active Security Features Use Cases Summary *(Cont'd)***

Feature	Use Case	User How-to
CSU tamper monitor and responses[1]	Monitor various security-critical device parameters (voltages, temperature) and assert appropriate tamper penalties.	Develop PS code to set up the CSU tamper and response registers as needed by the system.
Public key revocation[1]	Revoke the primary public key (PPK) or secondary public key (SPK) in response to a key management event which could be due to a breach in security or a normal crypto period key update.	Develop PS code to invalidate an expired PPK or SPKs.
Non-volatile (eFUSE) tamper event logging[1]	Securely log a tamper event in non-volatile memory (eFUSE) for later forensic analysis.	Develop PS code for logging tamper events in the USER_0 through USER_7 eFUSE registers.
User accessible crypto blocks[1]	Access and use the AES-GCM, RSA and/or SHA hardened crypto accelerators post secure boot.	Develop PS code or PL logic to interface with the desired crypto accelerator for use by the application.
ARM TrustZone	Allows and maintains isolation between secure and non-secure processes within the same system.	Develop PS code or use an operating system that takes advantage of ARM TrustZone features.
ARM v8 Cryptography Extensions[1]	Allows the ARM PS application to call crypto accelerator instructions for AES, SHA, and finite-field arithmetic.	Develop PS code or use an operating system that takes advantage of the ARM v8 cryptography extensions.
Memory protection unit (XMPU)[1]	Provides memory partitioning protection for memory and full-power domain (FPD) slaves.	Use the Vivado Design Suite processing system configuration wizard (PCW) to set up the XMPU blocks and/or through code running on the PS.
Peripheral protection unit (XPPU)[1]	Provides low-power domain (LPD) peripheral isolation and IP integrator (IPI) protection.	Use the Vivado Design Suite PCW to set up the XPPU blocks and/or through code running on the PS.
AXI/APB isolation block (AIB)[1]	Functionally isolates the AXI/APB masters from the slaves.	Use the Vivado Design Suite PCW to set up the AIB blocks and/or through code running on the PS.
System memory management unit (SMMU)[1]	Provides address translation and isolation to any DMA-capable agent other than the CPU.	Use the Vivado Design Suite PCW to set up the SMMU block and/or through code running on the PS.
GTS	Shut off PL outputs in response to a tamper event to prevent any information leakage out of the device.	Develop PS code to set up the CSU tamper responses for a secure lockdown and all GPIO to be tri-stated and/or instantiate PL STARTUP primitive and develop PL logic to determine the proper conditions for GTS assertion.
GSR	Restore PL flip-flop states to initial conditions in response to a tamper event, effectively clearing possible CT from within the device.	Instantiate PL STARTUP primitive and develop PL logic to determine the proper conditions for GSR assertion.

Notes:

1. This feature is new or improved in the Zynq UltraScale+ devices.

JTAG Port Disable (Active)

An attacker often starts at the external JTAG port when trying to break into a system. With Zynq UltraScale+ devices, there are a number of active methods to block the JTAG port that can be done both temporarily (through register setting) and permanently (through eFUSE).

By default, the JTAG port initially starts out disabled and is only enabled if a non-secure boot mode is detected or post secure boot by authenticated software. The bits in the jtag_sec register are used to enable/disable the security gates to control the JTAG paths (the bits are enabled by default).

If there are any JTAG-based debugging tools in your Zynq UltraScale+ device logic design (that are connected to the dedicated external JTAG port), breaking the JTAG chain does not allow them to function. During the Zynq UltraScale+ device debug phase, the JTAG chain can be left intact and then broken later on in the development cycle when the JTAG-based debugger tool is no longer required.

JTAG Monitoring (Detection)

The JTAG toggle detect is a security feature used to trigger a tamper response in the CSU when the JTAG signals are toggled. The toggle detect sends an alert to the CSU if the test data in (TDI) or test mode select (TMS) is asserted and the test clock (TCK) is running. The alert is sticky and remains asserted until a POR is received. The alert to the CSU requires three cycles of TCK to generate. This should prevent false detects from board power-up or other circumstances.

The tamper response is only serviced by the CSU ROM when the tamper response register is set in the CSU. The JTAG toggle detect is disabled in the CSU if any of the JTAG security gates are disabled. This allows secure software to have a built-in debug mode (assuming system IRQ is the highest level response programmed into the tamper response register).

PL Configuration Memory Integrity Checking (Detection)

Corruption of any of the PL internal configuration memory cells (that are configured by the decrypted bitstream) could cause the Zynq UltraScale+ device to operate in an unknown or undesired manner. The corruption could occur by an intentional post-configuration tamper attack or by an unintentional event such as a single-event upset (SEU). By using the SEM IP core [Ref 14], continuous readback of configuration data in the background of a design is performed to detect any bit flips. The SEM IP core can also perform SEU corrections.

Unique Identifiers (Detection)

Two types of unique identifiers (UIs) are available for use, device DNA and user eFUSE. These UIs can be used as anti-cloning security measures (that is, someone steals your image/bitstream and uses it to program their own devices) or for enabling or disabling certain features (upgrade or downgrade) depending on the value of the UI.

Device DNA consists of a 96-bit device-specific serial number and is set by Xilinx in one-time programmable (OTP) eFUSE bits on the Zynq UltraScale+ device during the manufacturing flow. User eFUSE provides 256 bits of user read/write OTP area and can be set by you internally by

code running on the PS. See *Programming BBRAM and eFUSEs (*XAPP1319) [Ref 10]. Both of these UIs can be used separately or in conjunction for security purposes.

Note: Use of device DNA or user eFUSE provides for unique IDs, but does not provide cryptographically strong confidentiality or authentication (such as AES-GCM). AES-GCM encryption is the preferred method of providing anti-cloning protection. However, by taking advantage of these UIs, you can add another layer to the overall AT scheme.

These UIs can be used to link the image/bitstream to one particular device (in the case of device DNA, or multiple devices in the case of user eFUSE). The UI comparison is put into the Zynq UltraScale+ device PS or PL design by you and the results of this comparison can be used to gate Zynq UltraScale+ device activity. For example, if the UI comparison fails, the design can refuse to function or function with limited capability. An example use case of the UIs is as described below.

1. Setup: Read the UI value(s) from the PS/PL design, generate a hash from the UI value(s), and store in a flash device accessible to the device using a robust one-way function (a keyed function such as HMAC or CMAC is the most secure) as shown in Figure 10.

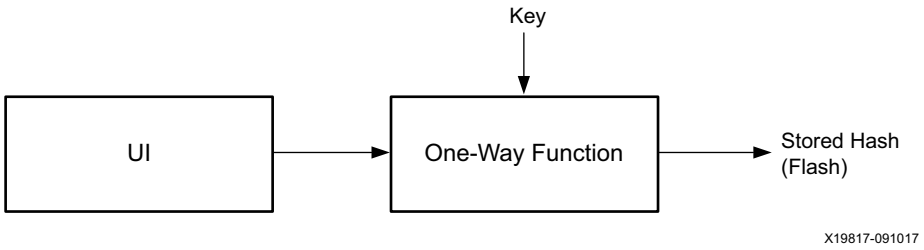

Figure 10: **Encrypt the DNA Value with a Confidential Key**

   The key for the one-way function shown in Figure 10 could be stored within the encrypted image/bitstream. If image/bitstream encryption is not used, this approach relies on the complexity of the image/bitstream to keep the hash key confidential.

2. Securely boot the Zynq UltraScale+ device.
3. Compare: The Zynq UltraScale+ device PS code reads the UI value from the DNA_0 through DNA_2 registers (see the *Zynq UltraScale+ MPSoC Register Reference* (UG1087) [Ref 8]), and then calculates the hash using the same algorithm (see Figure 11). The design then compares the calculated hash with the hash read from flash. If the hash passes, the design is allowed to become active.

   Note: The PS DNA value is not protected against changes by intentional writes (if RSA authentication is used, then only authorized software can access the PS DNA values). Use the PL DNA_PORTE2 primitive for applications requiring an unchangeable and unique device identifier. The PS DNA value is different than the PL DNA value. For more information, see Zynq UltraScale+ Device – PS DNA is not write protected and is a different value than the PL DNA (Xilinx Answer 71342) [Ref 17].

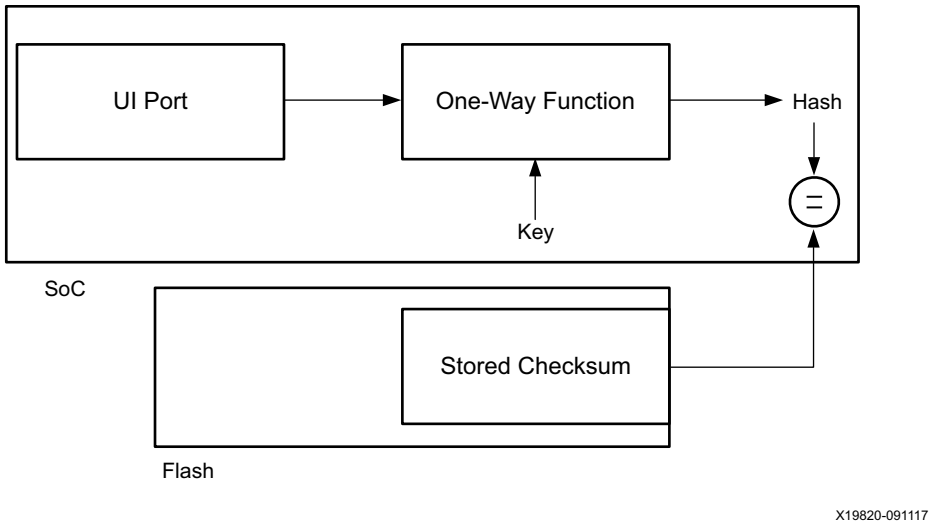

Figure 11: **Hash Comparison**

For additional information on these UIs, refer to the *Zynq UltraScale+ MPSoC Register Reference* (UG1087) [Ref 8] and *Security Solutions Using Spartan-3 Generation FPGAs* (WP266) [Ref 18] which also talks about device DNA operation for the Spartan®-3 generation FPGAs.

On-Chip Temperature and Voltage Monitors/Alarms (Detection/Response)

By modifying the normal operating voltages and/or temperature of an FPGA or SoC, an attacker might attempt to cause the device to behave in an unintended way, such as to extract data from it or cause it to bypass certain security features. For example, the Federal Information Processing Standards Publication (FIPS) 140-2 *Security Requirements for Cryptographic Modules* [Ref 19] states: "In particular, the cryptographic module shall monitor and correctly respond to fluctuations in the operating temperature and voltage outside of the specified normal operating ranges."

To help meet this type of requirement, Zynq UltraScale+ devices have one system monitoring (SYSMON) block each in the PS and the PL. For more information, see the PS System Monitor chapter in the *Zynq UltraScale+ MPSoC Technical Reference Manual* (UG1085) [Ref 2] and *UltraScale Architecture System Monitor User Guide* (UG580) [Ref 16]. The SYSMON block has a register interface that can be used to configure the block and provide the capability to monitor on-chip voltages as well as junction temperature. The PL SYSMON block is capable of monitoring off-chip voltages. The SYSMON block also has built-in alarm generation logic that is used to interrupt the processor based on certain alarm conditions. For example, it can shut down the system based on an over-temperature (OT) alarm generated from the SYSMON block or invoke a tamper penalty if one of the internal voltage rails goes too high or too low.

The option of using an external or internal voltage reference (VREF) for SYSMON is available as well. Although the external reference is more accurate, the internal reference is more secure because it is much more difficult for an adversary to tamper with. The selection between the internal and external reference is controlled by an external pin. The appropriate SYSMON status

register should be read to confirm that the internal VREF is being used. For more information, see the *UltraScale Architecture System Monitor User Guide* (UG580) [Ref 16].

Upper and lower alarm limits can be programmed directly into SYSMON for the on-chip parameters. Additional PL logic can be used to create alarm limits for external voltage inputs (for example, an external analog voltage tamper loop or the output of a pressure sensor). The status of the alarm signals can be used by your design or system to determine the appropriate course of action in case they become active (that is, determine the appropriate tamper penalty). The analog inputs are bandwidth limited. See the *UltraScale Architecture System Monitor User Guide* (UG580) [Ref 16] for the maximum input frequencies.

If detection of very fast changes in temperature or voltages is required, an off-chip solution might be required. It is up to you to define the required detection bandwidth. *The Sorcerer's Apprentice Guide to Fault Attacks* [Ref 20] describes a number of methods to mount an attack on a chip, one of them being variations in the power supply voltage.

Uninterruptible Internal Clock Source (Detection)

When using bitstream encryption, you can take advantage of an uninterruptible clock source named CFGMCLK (configuration internal oscillator clock output) located as an output on the PL STARTUP block primitive. This clock is always active and can be used as the basis for a user clock (or other critical user signal) monitoring function. Even though CFGMCLK can vary from its nominal value of 50 MHz ± 15% (see the *Zynq UltraScale+ MPSoC Data Sheet: DC and AC Switching Characteristics* (DS925) [Ref 21]), it can still be quite useful in a monitoring function to make sure a critical user clock or signal is still alive and toggling between a lower and upper frequency range (which takes into account the CFGMCLK variation). If the critical user clock or signal falls out of this range, it can indicate either that the design has malfunctioned or is being tampered with, and the appropriate penalty could be asserted.

CFGMCLK can also be used as the clock source for any other user-defined AT functions in the PL. It is important that AT functions cannot be halted by simply removing an external clock source.

PL Configuration Memory Clearing (Response/Penalty)

IPROG is an internal command sent by an application running on one of the PS processors through the PCAP interface that clears the PL configuration memory, all flip-flop contents, and key expansion memory, but does not clear the BBRAM AES key itself. IPROG is equivalent to the assertion of the external PROGRAM_B pin in FPGA families. This command effectively clears PL configuration memory (configuration data, block RAMs, UltraRAMs, and flip-flop states).

If using red key storage in the BBRAM and both of the AES key clear and IPROG penalties are invoked, the Zynq UltraScale+ device becomes inoperable because the existing image/bitstream can no longer be decrypted. The fact that device secure boot is no longer possible with the encrypted image/bitstream is an indication that a tamper event has occurred. At this point, your design might choose to load in an unencrypted image/bitstream so that there is some basic functionality without exposing any of the CT. Of course, an unencrypted image/bitstream cannot be loaded if using an eFUSE-based key and the enc_only eFUSE is also programmed.

Key Agility (Response)

Key agility refers to the ability to update or change the AES decryption key in the BBRAM through an application running on one of the PS processors (see *Programming BBRAM and eFUSEs* (XAPP1319) [Ref 10]). This does not apply to eFUSE-based keys because they are one-time programmable (OTP).

In Figure 12, a Zynq UltraScale+ device is shown with its initial key already loaded into the BBRAM. An external encrypted image/bitstream can then be loaded, decrypted, and have the initial interconnect (PL) logic design up and running.

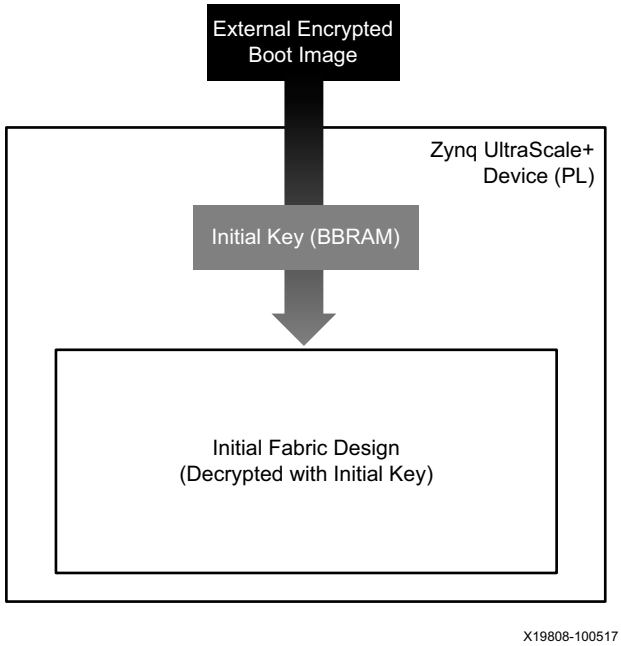

Figure 12: **Zynq UltraScale+ device with Key Loaded into BBRAM**

At some time in the future, a key management event occurs, which means that the key is required to be changed or updated due to a security breach, good cryptographic practice, or new design. By using some sort of secure key exchange algorithm (which would be an IP core running in PL logic or code running in the PS, such as the Diffie-Hellman protocol) an external encrypted key can be brought in, perhaps even through the Internet. This key can then be decrypted using logic in the PL or code in the PS, loaded into the BBRAM, and the CRC integrity check can be performed internally (see Figure 13).

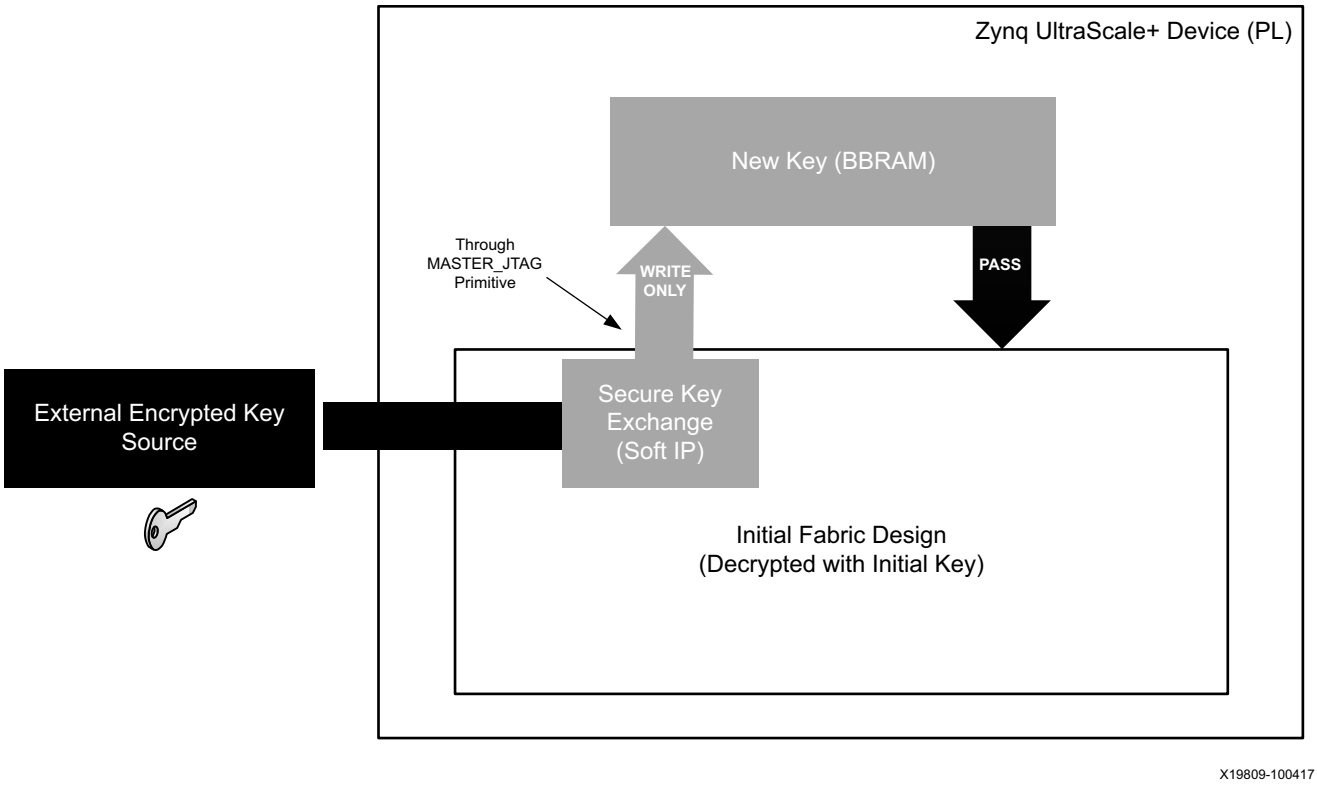

Figure 13: **Key Management Event**

Then perform a POR, load in an external encrypted image/bitstream (encrypted on the new key), decrypt it, and have new PS and PL designs that are decrypted with this new key as shown in Figure 14.

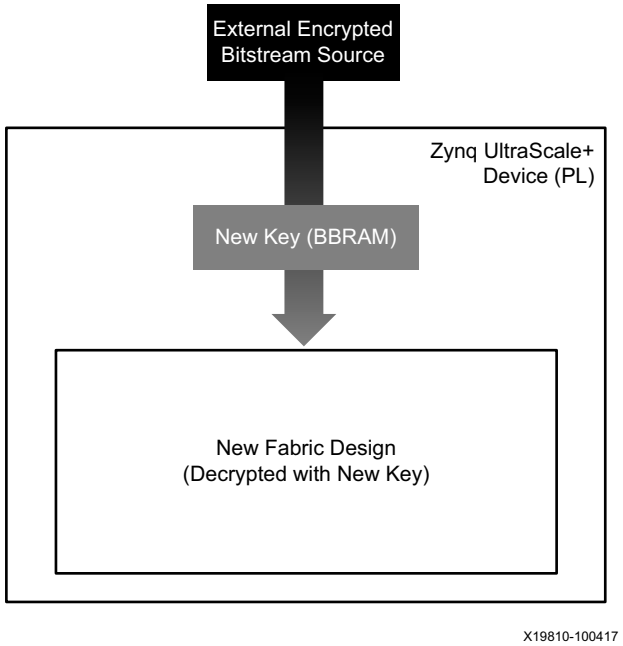

Figure 14: **New PL Design**

The above scheme can be used to update the key in the field without having to bring the board or the system all the way back to a secure facility.

BBRAM Key Zeroize (Response/Penalty)

If the BBRAM red key storage is being used, PS code can be developed that would determine the proper conditions for clearing the AES key through the aes_key_clear register and for reading the verification bit in the aes_status register to prove zeroization. After the key is zeroized, the Zynq UltraScale+ device is useless until reprogrammed with the same key or rebooted with an unencrypted image/bitstream (perhaps with reduced functionality). Ensure that the aes_key_clear is only asserted under the proper conditions. In many cases, equipment must be taken out of the field and sent back to a central depot or manufacturing facility to be re-enabled with a key load operation.

The AES key clearing can also be combined with the IPROG command (see PL Configuration Memory Clearing (Response/Penalty)) in response to tampering to also erase the PL configuration memory.

CSU Tamper Monitor and Response (Detection/Response)

Table 6 and Table 7 summarize the monitoring and optional tamper response capabilities of the CSU after the device has securely booted. For more information, see the Security chapter in the *Zynq UltraScale+ MPSoC Technical Reference Manual* (UG1085) [Ref 2], Programming View of Zynq UltraScale+ Devices chapter in the *Zynq UltraScale+ MPSoC Software Developer Guide* (UG1137) [Ref 3], and the *Zynq UltraScale+ MPSoC Register Reference* (UG1087) [Ref 8].

Table 6: **CSU Tamper and Monitor Registers**

Register	Monitor Description
csu_tamper_12	SYSMON voltage alarm for transceiver
csu_tamper_11	SYSMON voltage alarm for PSIO bank 3
csu_tamper_10	SYSMON voltage alarm for PSIO bank 0/1/2
csu_tamper_9	SYSMON voltage alarm for DDRPHY
csu_tamper_8	SYSMON voltage alarm for VCCPAUX
csu_tamper_7	SYSMON voltage alarm for VCCPINT_FPD
csu_tamper_6	SYSMON voltage alarm for VCCPINT_LPD
csu_tamper_5	SYSMON over temperature alarm for APU
csu_tamper_4	SYSMON over temperature alarm for LPD
csu_tamper_3	PL SEU error
csu_tamper_2	JTAG toggle detect
csu_tamper_1	External MIO
csu_tamper_0	CSU register

After a tamper event occurs, how the CSU responds is user configurable. Table 7 indicates which bit in the tamper response registers to set to obtain a specific tamper response for each tamper event. Multiple tamper response bits can be set for each tamper event.

Table 7: **Tamper Monitor and Response Bits**

Bit	Response/Penalty Description
4	Erase the BBRAM key (in addition to any of the options below)
3	Secure lockdown and 3-state all I/O
2	Secure lockdown
1	System reset
0	System interrupt

The registers are readable but can only be set on write accesses. Specifically, after a specific tamper response is selected for a given tamper event, the bit selecting that response cannot be cleared except by a POR. This prevents incorrect or rogue software from accidentally decreasing the tamper response penalty. Tamper responses can only be added.

Public Key Revocation (Response)

There are two public key types used in the Zynq UltraScale+ devices: the primary public key (PPK) and the secondary public key (SPK). Table 8 lists the characteristics of each public key type. For more information, see the Security chapter in the *Zynq UltraScale+ MPSoC Technical Reference Manual* (UG1085) [Ref 2] and the *Zynq UltraScale+ MPSoC Register Reference* (UG1087) [Ref 8].

Table 8: **Public Key Characteristics**

Public Key	Number	Location	Revocation	Notes
Primary (PPK)	2	External memory and hash in eFUSEs	Can be revoked	Only used to authenticate SPK and authentication header.
Secondary (SPK)	Up to 32	Boot image	Can be revoked	Signed by PPK. Used to authenticate everything else.

As a tamper penalty, if the current PPK and/or SPK is revoked then the device becomes inoperable (unable to securely boot) until it is securely updated in the field (or sent back to a secure depot) with a new signed image/bitstream and a new PPK hash and/or SPK ID programmed into the eFUSEs. If both PPKs are revoked, the device essentially becomes a brick and cannot be used again because it will never boot.

Non-Volatile (eFUSE) Tamper Event Logging (Response)

The Zynq UltraScale+ devices have a 256-bit USER eFUSE register. This register can be used to flexibly meet the needs of a non-volatile area such as tamper, maintenance logging, or both. The eFUSE register bits are programmed through the PS into USER_0 through USER_7 32-bit eFUSE registers.

IMPORTANT: *After these eFUSE bits are programmed, it is possible to program other eFUSEs (a write disable) that prevents any more programming of the eFUSE register bits (that is, to lock the door). This is done by programming the corresponding lock bits in the misc_user_ctlr eFUSE register. Thus, an adversary cannot attempt to overwrite the tamper log information.*

User Accessible Crypto Blocks (Prevention)

Post secure boot, you can take advantage of the hardened CSU crypto accelerators and use them for the PS or PL applications. The available crypto functions are listed below.

- Confidentiality: AES-GCM 256-bit device key or user key (key update register - KUP). The AES-GCM core has a 32-bit word-based data interface for symmetric key-based encryption and decryption.
- Authenticity: Montgomery multiplier for RSA-4096 calculations.
- Integrity: SHA-3/384 engine for 384-bit hash calculation.

For details on using these crypto accelerators in a design, refer to the XilSecure Library Reference appendix in the *Zynq UltraScale+ MPSoC Software Developer Guide* (UG1137) [Ref 3] and the Security chapter in the *Zynq UltraScale+ MPSoC Technical Reference Manual* (UG1085) [Ref 2].

ARM TrustZone (Prevention)

The ARM TrustZone technology support allows and maintains isolation between the secure and non-secure processes within the same system (see the *Zynq UltraScale+ MPSoC Technical Reference Manual* (UG1085) [Ref 2] and *Zynq UltraScale+ MPSoC Software Developer Guide* (UG1137) [Ref 3]). The basic principle behind TrustZone technology is the isolation of all software and hardware states and resources into two worlds, trusted and not trusted.

A non-secure virtual processor can only access non-secure system resources, whereas a secure virtual processor can see all the resources. Resource access is extended to bus accesses using the NS flag which is mapped to the AxPROT[1] on the AXI bus. Any part of the system can be designed to be part of the secure world including debug, peripherals, interrupts, and memory. By creating a security subsystem, assets can be protected from software attacks and common hardware attacks.

Typical example use cases of TrustZone technology include firmware protection, security management, and peripheral/IO protection.

ARM v8 Cryptography Extensions (Prevention)

The cryptography extension supports the ARM v8 cryptography extensions. See the Application Processing Unit chapter in the *Zynq UltraScale+ MPSoC Technical Reference Manual* (UG1085) [Ref 2]. The cryptography extension adds new A64, A32, and T32 instructions to advanced single-instruction multiple-data (SIMD) that accelerate the following:

- Advanced encryption standard (AES) encryption and decryption.
- Secure-hash algorithm (SHA) functions SHA-1, SHA-224, and SHA-256.
- Finite-field arithmetic used in algorithms such as Galois/counter mode and elliptic curve cryptography.

Note: Cryptography extensions can be permanently disabled through eFUSEs.

Xilinx Memory Protection Unit (Prevention)

The Xilinx memory protection unit (XMPU) provides memory partitioning and TrustZone protection for memory and FPD slaves. For more information, see the System Protection Unit chapter in the *Zynq UltraScale+ MPSoC Technical Reference Manual* (UG1085) [Ref 2], Security Features chapter in the *Zynq UltraScale+ MPSoC Software Developer Guide* (UG1137) [Ref 3], and *Isolation Methods in Zynq UltraScale+ MPSoCs* (XAPP1320) [Ref 22]. The XMPU can be configured to isolate a master or a given set of masters to a programmable set of address ranges.

The XMPU consists of the following features.

- Slave AXI port.
- Master AXI port with poison output.
- APB slave for programming the XMPU.
- Level-sensitive, asynchronous interrupt output.
- AXI clock (same for master and slave ports) and APB clock.
- XMPU has a lock register which can be set to prevent any further programming of the XMPU until a POR.

Xilinx Peripheral Protection Unit (Prevention)

The Xilinx Peripheral Protection Unit (XPPU) is a look-up based peripheral protection unit. For more information, see the System Protection Unit chapter in the *Zynq UltraScale+ MPSoC Technical Reference Manual* (UG1085) [Ref 2], Security Features chapter in the *Zynq UltraScale+ MPSoC Software Developer Guide* (UG1137) [Ref 3], and *Isolation Methods in Zynq UltraScale+ MPSoCs* (XAPP1320) [Ref 22]. The XPPU is for protecting peripherals, message buffers (for inter-processor interrupts and communications), and Quad-SPI flash memory. The XPPU also has a mechanism to protect itself.

In comparison with the XMPU, the XPPU uses finer grained address matching, provides many more address apertures, and suits the different needs of peripherals, IPI, and Quad-SPI flash memory.

The XPPU consists of the following features.

- Slave AXI port where master ID is carried on lower bits of AxUSER.
- Master AXI port where master ID is carried on lower bits of AxUSER.
- APB slave for programming the XPPU.
- Level-sensitive, asynchronous interrupt output.
- AXI clock (same for master and slave ports) and APB clock.

AXI and APB Isolation Block (Prevention/Response)

The Xilinx AXI/IPB Isolation Block (AIB) is part of the interconnect that is responsible for functionally isolating the AXI/APB master from the slave in preparation for an AXI/APB master or slave to be powered down. For more information, see the Interconnect chapter in the *Zynq UltraScale+ MPSoC Technical Reference Manual* (UG1085) [Ref 2] and *Isolation Methods in Zynq UltraScale+ MPSoCs* (XAPP1320) [Ref 22]. The AIB manages AXI and APB interfaces during the isolation process resulting in a graceful transition to a power-down state. The AIB can also be used to force isolation in a design and can be enabled in response to a tamper event.

System Memory Management Unit (Prevention)

The system memory management unit (SMMU) offers isolation services. For more information, see the System Protection Unit chapter in the *Zynq UltraScale+ MPSoC Technical Reference Manual* (UG1085) [Ref 2], Security Features chapter in the *Zynq UltraScale+ MPSoC Software Developer Guide* (UG1137) [Ref 3], and *Isolation Methods in Zynq UltraScale+ MPSoCs* (XAPP1320) [Ref 22]. The SMMU provides address translation for an I/O device to identify more than its actual addressing capability. In absence of memory isolation, I/O devices can corrupt system memory. The SMMU provides device isolation to prevent DMA attacks. To offer isolation and memory protection, it restricts device access for DMA-capable I/O to a pre-assigned physical address space.

Global 3-State (Response/Penalty)

Depending on the system design, critical (red) information might flow out of the external Zynq UltraScale+ device pins (for example, a cryptographic module). Asserting the GTS input on the STARTUP block in response to a tamper event causes all PL outputs to immediately enter a high-Z state and prevent any more data from flowing outside the device. This could be an immediate step to take just prior to an IPROG or key clear to ensure that red data flow is halted as soon as possible. The CSU tamper registers can also be used to cause a secure lockdown and put all the GPIO in a high-Z state in response to a particular tamper event.

Global Reset (Response/Penalty)

Critical data or sensitive parameters can be stored in PL logic registers, for example, a user key (not the AES BBRAM bitstream decryption key). Asserting the GSR input on the STARTUP block in response to a tamper event causes all PL registers (that is, flip-flops) to be restored to their default state. This could be an immediate step to take along with key clear to make sure all sensitive data in the Zynq UltraScale+ device is erased as soon as possible. The GSR does not impact the shift register look-up table (SRL), block RAM (BRAM) or UltraRAM (URAM) contents. These must be cleared by the design or by an IPROG command.

Tamper Resistance Guidance

This section provides guidance and technical tips that can be used in conjunction with the previously addressed built-in silicon AT features to create tamper-resistant designs using Zynq UltraScale+ devices.

Limiting Device Key Usage (Prevention)

Good security practice is to limit the use of keys. The goal is to minimize or eliminate the use of keys stored on the Zynq UltraScale+ device, which also reduces the risk of having to change them more often. To minimize the symmetric key (AES) usage, specify the use of the operational (OP) key in the boot header such that the AES device key is only used to decrypt the secure header in the boot image, which consists of the OP key and initialization vector (IV) for the first block of the boot image. Therefore, the first FBSL block is decrypted with the OP key.

The use of the PPK asymmetric key (RSA) should also be minimized such that it is only used to authenticate the SPKs. Limiting the use of the PPK key limits the exposure of the primary private key which must be highly protected by the customer in their secure facility.

Load CT Only When Needed (Prevention)

If the design can be partitioned into sections that contain non-critical and critical technology blocks, it might be possible to only have the non-CT portion of the design resident at all times and use partial reconfiguration (PR) features of the Zynq UltraScale+ device to allow the CT to be loaded only when needed. The CT can then be erased by loading in a black box version of the PR region when it has completed its tasks. The partial bitstream for the CT can be decrypted in the device's PL by your algorithm of choice. In response to a tamper event, both the PR region and the key for the CT (perhaps stored in the block RAM or PL) could be erased.

For example, Figure 15 and Figure 16 illustrate a general system with a PL, CPU, and external memory devices (for PL configuration, PR, CPU code, and CPU boot code).

In Figure 15, the PR region (named user CT logic region) is empty.

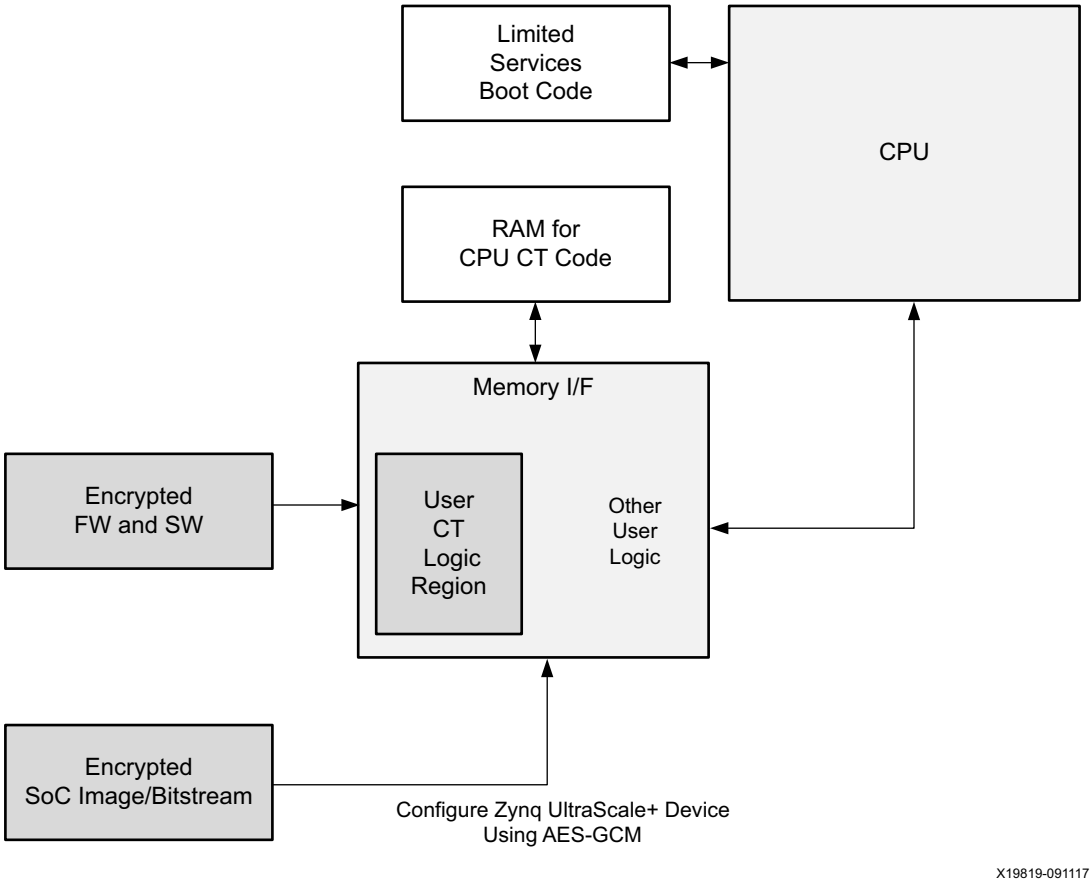

Figure 15: **Use Model: Protecting System CT**

In Figure 16, the PR region on the device has been dynamically loaded with CT logic through the PR (named user CT logic). When the CT function is completed or a tamper event occurs, the PR region can be returned to the state shown in Figure 15 by loading in an empty version of the PR module.

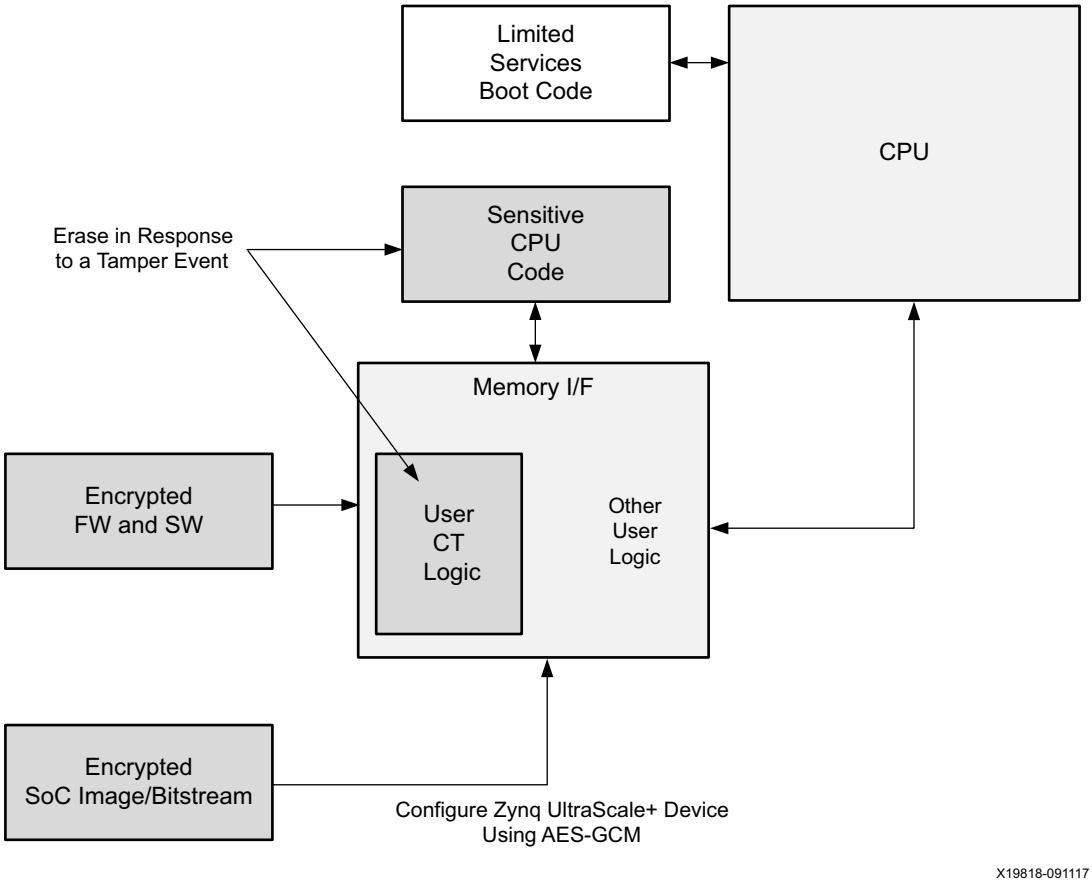

Figure 16: **Use Model: Protecting System CT-Tamper Response**

Note: If a tamper event occurs, the external sensitive CPU code memory could be erased so that only encrypted, cleared, or non-sensitive external memory contents remain.

In these examples, the PCAP is used to perform the PR. The PCAP is a trusted channel and therefore, allows encrypted or unencrypted PR bitstreams (even if an eFUSE key is being used with the cfg_aes_only eFUSE bit blown). An encrypted and authenticated PR bitstream is always recommended where the decryption and authentication take place with your decryptor authentication engine of choice in the PS or PL.

Key Erase via External Shunt

Another method to erase the BBRAM key is through an external shunt to ground on the VCC_PSBATT line. This method can be used to erase the key when main power (VCCINT and VCCAUX) to the Zynq UltraScale+ device is not applied because active features, such as key clear, can be used only after the device is powered up and configured. For instance, if a system-level tamper event is detected prior to the device having its main power applied (perhaps a tamper switch becomes activated), the external battery power line to the Zynq UltraScale+ device VCC_PSBATT pin can be opened and the VCC_PSBATT pin driven to ground with some sort of transistor shunt. Care must be taken that the circuit is designed to open the battery connection before shunting the VCC_PSBATT pin to ground. Another option is to connect the battery to the VCC_PSBATT pin through a resistor. (The VCC_PSBATT pin maximum input current ICC_PSBATT is between 3650 and 150 nA depending on the battery voltage and whether the real-time clock (RTC) is enabled or disabled.) See the *Zynq UltraScale+ MPSoC Data Sheet: DC and AC Switching Characteristics* (DS925) [Ref 21]. By choosing the appropriate resistance value, the VCC_PSBATT pin can be shunted to ground directly without causing excessive current flow out of the battery.

If the Zynq UltraScale+ device is not powered up (no VCCINT, VCCAUX, or other voltages except for VCC_PSBATT), it takes a worst-case maximum time of 200 ms for the AES key stored in the BBRAM to become erased if the VCC_PSBATT pin is properly shunted to ground.

IMPORTANT: *For improved security and to keep leakage as small as possible, use a battery whose voltage is as low as possible. For more information on battery voltage levels, refer to the Zynq UltraScale+ MPSoC Data Sheet: DC and AC Switching Characteristics (DS925) [Ref 21].*

Preemptive BBRAM Key Zeroize

Another use case for key zeroize could be a preemptive measure. After loading and decrypting the image/bitstream, the BBRAM key can be purposely zeroized before sending the system out into the field. Of course, this would only work for a system that is not intended to be power cycled after it becomes deployed (for example, a missile after being launched). This could also be used for an eFUSE key by writing over it with all ones by the PS before deployment as long as eFUSE key reading and writing has not yet been disabled.

Avoiding Weak or Duplicate Keys

All zeros, all ones, or repetitive patterns should never be used in user keys. Keys should not be reused if at all possible. Personnel access to the key values should be tightly controlled (that is, only those with a need to know should have access to key data). Ideally, a random source should be used to create the keys. Avoid using weak keys. For example, an all zeros random key is theoretically possible but should never be used. The Vivado Design Suite can automatically generate the AES-GCM keys. However, it uses a pseudorandom process seeded with the current date and time. The most secure keying material comes from a truly random process.

Key management is a very important element (and probably the most complex) in any cryptographic system. For additional help on this subject, the NIST Key Management Guideline [Ref 23] is a useful reference.

Sending Tamper Status Outputs to System

Upon a tamper event (in addition to asserting a penalty) your design could send tamper status information back to the system (instead of or, in addition to logging it locally in user eFUSE space). The system could then store this information away for future auditing purposes. It would have to be designed to transmit the data before an IPROG command (tamper penalty) is given.

Restricting Access to Zynq UltraScale+ Device Probe Points

Making it difficult for an attacker to get near the Zynq UltraScale+ device is a good example of a layered approach. A robust tamper boundary (perhaps with tamper detect switches) can be used around any device(s) that might contain CT. For example, an activated tamper switch could cause a shunt to go active on the Zynq UltraScale+ device VCC_PSBATT line. Buried vias and routing can be used on the printed circuit board for Zynq UltraScale+ device signals, power supply routing can be kept within buried layers (and difficult to get to), and adequate decoupling can be used (buried capacitance technology should be used, if possible). JTAG boundary-scan techniques can be relied on for board-level factory testing, and test points should be removed from production boards.

Conclusion

This application note summarizes the AT features currently available in Zynq UltraScale+ devices and gives practical examples of how to use them effectively. By taking advantage of these features and following the AT guidance early on in the design cycle, a tamper-resistant Zynq UltraScale+ device enabled system design can be realized.

No single AT feature or technique is 100% effective all of the time or can meet all the AT needs for the entire system. However, making the adversary's job as difficult and expensive as possible and following a multi-layered approach almost always yields very good results.

The tools and technologies for the development and testing of new integrated circuits including FPGAs and SoCs are always evolving and improving. In parallel, the tools used by adversaries also evolve and improve, so it is important to be aware of the available AT features and techniques. Additionally, Xilinx is committed to staying abreast of these developments to enhance and develop new features to protect customer IP now and into the future.

Documentation Navigator and Design Hubs

Xilinx® 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 on the Xilinx website.

References

1. Xilinx Zynq UltraScale+ MPSoC
2. *Zynq UltraScale+ MPSoC Technical Reference Manual* (UG1085)
3. *Zynq UltraScale+ MPSoC Software Developer Guide* (UG1137)
4. Xilinx Vivado Design Suite
5. *Solving Today's Design Security Concerns* (WP365)
6. *Developing Tamper-Resistant Designs with UltraScale and UltraScale+ FPGAs* (XAPP1098)
7. Security Monitor Product Brief
8. *Zynq UltraScale+ MPSoC Register Reference* (UG1087)
9. *Zynq UltraScale+ MPSoC PUF Characterization Report* (RPT236)
10. *Programming BBRAM and eFUSEs* (XAPP1319)
11. *Zynq UltraScale+ MPSoC: Embedded Design Tutorial* (UG1209)
12. *External Secure Storage Using the PUF* (XAPP1333)
13. Zynq UltraScale+ MPSoC - Boot Times Estimation (Xilinx Answer 67475)
14. Soft Error Mitigation (SEM) Core
15. *UltraScale Architecture Libraries Guide* (UG974)
16. *UltraScale Architecture System Monitor User Guide* (UG580)

17. Zynq UltraScale+ Device – PS DNA is not write protected and is a different value than the PL DNA (Xilinx Answer 71342)

18. *Security Solutions Using Spartan-3 Generation FPGAs* (WP266)

19. *Security Requirements for Cryptographic Modules*, FIPS PUB 140-2 www.nist.gov/itl/upload/fips1402.pdf

20. Hagai Bar-El, Hamid Choukri, David Naccache, Michael Tunstall, and Claire Whelan. *The Sorcerer's Apprentice Guide to Fault Attacks*. http://eprint.iacr.org/2004/100.pdf

21. *Zynq UltraScale+ MPSoC Data Sheet: DC and AC Switching Characteristics* (DS925)

22. *Isolation Methods in Zynq UltraScale+ MPSoCs* (XAPP1320)

23. NIST Key Management Guideline csrc.nist.gov/groups/ST/toolkit/key_management.html

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
10/13/2017	1.0	Initial Xilinx release.
08/30/2018	1.1	Updated Table 1. Added a note to Obfuscated Key Loading and Storage. Added Design for Test (DFT) Boot Mode Disable (Passive). Added a note about the PS DNA value on page 24 in Unique Identifiers (Detection).

Please Read: Important Legal Notices