UG0753
User Guide
PolarFire FPGA Security

Microsemi
Power Matters.™

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA
Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Fax: +1 (949) 215-4996
Email: sales.support@microsemi.com
www.microsemi.com

About Microsemi

Microsemi Corporation (Nasdaq: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, California, and has approximately 4,800 employees globally. Learn more at www.microsemi.com.

Contents

Figures

Tables

1 Revision History

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the most current publication.

1.1 Revision 2.0

The following is a summary of the changes made in revision 2.0 of this document.

- Updated security overview information. For more information, see PolarFire FPGA Security Overview, page 2.
- Added tamper macro and configurator images. For more information, see Figure 18, page 45, Figure 19, page 45, and Figure 20, page 46.
- Updated Table 30, page 46 by adding tamper detection flags. For more information, see Table 30, page 46.
- Added a note about disabling SerDes I/Os. For more information, see IO Disable, page 48.
- Added voltage detector information. For more information, see Voltage Detectors, page 52.
- Updated tamper and voltage sensor information. See Temperature and Voltage Sensor, page 52.
- Updated data security information. For more information, see PolarFire Data Security, page 67.

1.2 Revision 1.0

Revision 1.0 was the first publication of this document.

2 PolarFire FPGA Security Overview

Applications today are expected to meet very demanding functional requirements, and must do so securely, protecting both, the application design and information. Protecting design and information calls for secure hardware, and design and data security. Microsemi PolarFire FPGAs provide a solid foundation for all application security needs.

Microsemi PolarFire™ FPGAs represent the industry's most advanced programmable security FPGA. Built on the SmartFusion®2 and IGLOO®2 4th-generation flash FPGA security model, the design security, anti-tamper, and data security features have been greatly expanded and enhanced for PolarFire FPGAs.

The key security features in PolarFire devices that provide supply chain assurance and IP protection benefits include:

- AES256 encryption-protected bitstream, and key management protocols using licensed differential power analysis (DPA) countermeasures.
- Built-in tamper countermeasures including voltage monitors, temperature monitors, clock glitch detectors, voltage glitch detectors, protective meshes, and memory scrambling.
- Data integrity verification through built-in cryptographic digest capabilities.
- Built-in zeroization capabilities for all on-chip memories and the FPGA fabric.
- 56 KB of physically unclonable function (PUF) protected secure non-volatile memory (sNVM).
- Integrated SRAM-PUF for key storage.
- Secure re-programmable design security keys using non-volatile memory.
- Secure supply chain management through the use of hardware security modules (HSM) during wafer test and packaging.
- Supply chain assurance via a 1024-byte digitally signed X.509 FPGA certificate embedded in every FPGA.

2.1 PolarFire FPGA Security Features

Microsemi PolarFire FPGAs include features that provide enhanced security during all stages of the device life cycle—from user key injection and bitstream programming, to field updates, and finally to device decommissioning. The following sections of this chapter provide an overview of these security features, which include:

- PolarFire FPGA Security Architecture
- PolarFire FPGA Design Security Features
- PolarFire FPGA Data Security Features
- DPA Patent Portfolio License

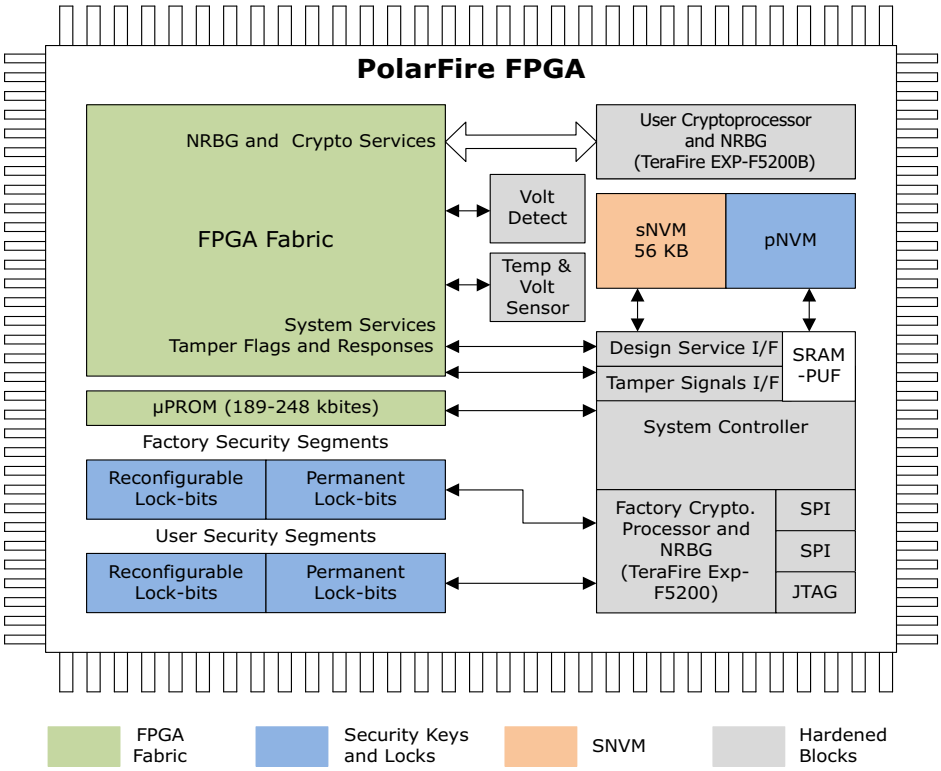
2.1.1 PolarFire FPGA Security Architecture

Figure 1, on page 3 illustrates the PolarFire FPGA device architecture from a security perspective, highlighting the following:

- The non-volatile FPGA fabric is built on a state-of-the-art 28 nm, low power, non-volatile process technology. It includes logic elements, on-chip memories (µSRAM and LSRAM) and math blocks. Flash-based FPGA configuration memory cells located within the FPGA fabric directly control the routing switches and look-up tables (LUT) that are used to implement the user's design.
- The µPROM is constructed with SEU-immune FPGA configuration non-volatile cells that are readable at runtime through the fabric interface. The µPROM is programmed with the FPGA bitstream during fabric programming and cannot be programmed independently. It can be used to store SEU-immune parameters, constants, IDs, and/or parametric or initialization data.
- Below the µPROM (lower left) are the security segments that store some of the factory and user settings (also known as lock bits).
- The integrated Athena TeraFire® EXP-F5200B cryptoprocessor (top right), supports various data security protocols.

- The sNVM allows the user to store data in an encrypted and authenticated form, as authenticated plain text, or as plain text. Individual sNVM pages can be write-protected during bitstream generation, making it easy to protect sensitive data from being overwritten at runtime.
- The pNVM block allows the system controller to privately store various keys and user settings, as well as PUF activation codes and the device's X.509-compliant certificate. The sNVM, and Keys located in the pNVM are accessible only to the system controller, providing user access to the sNVM sector as a system service.
- The system controller, manages all programming, verification, design security, key-management, and related operations. The system controller communicates with the factory's TeraFire® cryptoprocessor, which is used for all device specific cryptographic functions, and the Quiddikey®-Flex SRAM-PUF which is used for secure key storage.
- The system controller provides various system services, ranging from Flash*Freeze™ mode to digital signatures via an FPGA fabric interface. Tamper alarm flags (such as for a mesh break or an under-voltage condition) and built-in response commands (such as zeroization) also use an FPGA fabric interface. Voltages and temperature can be digitally monitored via yet another FPGA fabric interface.

Figure 1 • **PolarFire FPGA Simplified Security Model**



Note: There are three processors in the PolarFire security model. They are:

- The ARM® Cortex®-M3 hardened processor in the system controller is reserved for all programming, FPGA power up, verification, design security key-management, and related operations. It cannot be reconfigured. For more information, see System Controller for Programming and Design Services, page 4.
- The Athena TeraFire® core (F5200ASR), also known as the factory cryptoprocessor, is an additional processor in the system controller used to accelerate cryptographic operations. For more information, see Factory Cryptoprocessor and NRBG Block, page 4.
- The Athena TeraFire® F5200B DPA-resistant cryptoprocessor, also known as the user cryptoprocessor, is capable of implementing all NIST Suite-B plus additional algorithms. It is available in PolarFire FPGAs that have an “S” suffix in the model number. For more information, see User Cryptoprocessor and NRBG Block, page 6.
  You can also use a soft CPU such as an ARM Cortex-M1 processor, or a CPU using RISC-V™ instruction set architecture in the FPGA fabric.

2.1.1.1 System Controller for Programming and Design Services

The system controller in PolarFire devices manages all programming, FPGA power up, verification, design security key-management, and related operations. The system controller includes the industry standard ARM Cortex-M3 hardened processor reserved for these functions, and is not reconfigurable.The FPGA designer cannot write code to be run by the ARM Cortex-M3 hardened processor in the System Controller. Its programming and runtime operations are determined by a dedicated immutable metal-mask ROM. In addition to the ROM, some of the pNVM is reserved by Microsemi for future improvements and bug fixes in new devices.

During the programming process, the system controller authenticates and decrypts incoming bitstreams, erases and writes the target flash memory segments, and responds to other external programming-related protocols, such as key verification. The system controller also provides both, internal and external design services, such as reporting the factory serial number, the JTAG USERCODE value, exporting the device certificate and so on. For more information, see Supply Chain Assurance, page 30.

User security lock bit settings control access to many services. The system controller can optionally be suspended after booting, (See System Controller Suspend Mode, page 66). The system controller has both, a JTAG interface, and an interface to external SPI memory. It also manages anti-tamper features and various chip-level debug features.

2.1.1.2 Factory Cryptoprocessor and NRBG Block

The factory cryptoprocessor is an additional co-processor within the system controller for accelerating cryptographic operations. It allows complex or bulk cryptographic operations to execute asynchronously, and frees up the ARM Cortex-M3 hardened processor in the system controller for other tasks.

All cryptographic algorithms are implemented using patented DPA-resistant techniques to prevent secret key extraction by an adversary using simple or differential power analysis (SPA or DPA), simple or differential electromagnetic analysis (SEMA or DEMA), or timing analysis (TA). This protection extends to the messages digested using any of the secure hash algorithms (SHA), even though they do not directly use a secret key, because they are used in the hash-based message authentication algorithm (HMAC) that does.

The factory cryptoprocessor also incorporates a non-deterministic random bit generator (NRBG), also known as a true random number generator (TRNG). The integrated true random number generator enables modern FPGA cryptographic protocols that provide protection against attacks (such as replay attacks) that are not possible without one. It is also used for high-quality key, nonce, and initialization vector generation.

2.1.1.3 SRAM-PUF

The SRAM physically unclonable function (SRAM-PUF) is a novel key storage mechanism called Quiddikey-Flex, licensed from Intrinsic-ID, B.V, having superior security attributes. It combines the passive zeroization feature of volatile memory with tamper-resistant nonvolatile key storage, without requiring batteries. The SRAM-PUF in PolarFire FPGA is an upgraded version of their previous core

used on SmartFusion2 and iGLOO2 FPGAs and incorporates a bus-keeper PUF in addition to the SRAM-PUF. It is effectively two independent PUFs integrated into one. The bus-keepers add additional entropy, making the PUF harder to attack.

Quiddikey uses the random start-up behavior of a 2KB (16384 bits) SRAM block plus 1536 individual bus-keeper cells to determine a static secret unique to each device. This SRAM (including its controller), and the bus keepers have been specially designed with PUF security requirements in mind. In each unique device, the SRAM and bus-keeper power-on independently, down to the single-bit level. In a single device, however, there is sufficient repetition from one power on to the next to reconstruct the same intrinsic secret each time using forward error correction techniques and helper data (effectively parity bits and meta data) called is "activation code". This master PUF secret, derived from the SRAM and bus-keeper static entropy and reconstructed reliably subsequent to the initial enrollment with the aid of the activation code, is used to derive or protect cryptographic keys with a 256-bit security strength. In PolarFire devices, the system controller can reconstruct two master secret keys using two different activation codes stored in the pNVM. One PUF master secret key is used for wrapping design security keys, and the second is used for the sNVM encryption and authentication features.

Intrinsic keys are randomly generated by Quiddikey's internal NRBG, while extrinsic keys are generated elsewhere (for example, by the TeraFire NRBG, or an imported bitstream encryption key) and provided to Quiddikey by the system controller. Both types of keys are in turn protected (wrapped) by the PUF master intrinsic secret. Wrapped keys are called key codes, and contain the AES encrypted key and other metadata such as an initialization vector (IV) for added cryptographic strength and a message authentication tag (MAC tag) that can be used to verify a key has been decrypted correctly—with a properly reconstructed PUF master secret key. Key codes are stored in the private NVM, accessible only by the system controller, thus providing the encrypted key an additional layer of protection.

When power is removed from the SRAM and bus-keepers, the source data for the master secret(s) effectively disappears. There is no known technology to detect the start-up behavior of an SRAM or bus-keeper cell without actually powering it up. The start-up behavior is determined by virtually undetectable atomic-scale manufacturing differences in each transistor, such as the thickness of the gate dielectric, the number of atoms diffused into the channel region, and other random process-related factors. Since each unique device's power-up state is independent and unpredictable, with no two devices ever being the same, the function is deemed physically unclonable, and is analogous in many ways to a biometric identifier such as human fingerprints or iris patterns, which are considered physically unclonable.

The first time the SRAM-PUF is used, a particular intrinsic secret is determined in a process called enrollment. In order to determine the exact same secret on each subsequent power-up cycle in spite of a few percent bit-level power-on to power-on noise, a base activation code (effectively parity data) is stored in a dedicated read- and write-protected region of the pNVM. During subsequent power-on cycles, the Quiddikey logic reads the SRAM and bus-keeper start-up values and applies the base activation code to regenerate the PUF secret.

In this scenario, there is a strong analogy to a human fingerprint. Each time a fingerprint is scanned, the measurement is slightly different due to noise, but still close and unique enough to be able to identify the person. All the SRAM and bus-keeper bits have very high entropy. There is enough entropy in the 16 Kbits of SRAM and 1.5 Kbits of bus-keepers for the PolarFire FPGA to enroll two activation codes to reconstruct two independent, full entropy 256-bit master secret keys—one for each activation code.

The SRAM-PUF is used by the Nonce system service to generate a true random seed from the noisy SRAM and bus-keeper bits, of which there are always a sufficient number after each PUF power cycle to guarantee 256 bits of dynamic entropy. This nonce can be optionally used to provide additional entropy to the NRBG in the user TeraFire cryptoprocessor core. We recommend that the user take advantage of the Nonce service this way to provide additional tamper resistance to the random bits the TeraFire core computes from its internal entropy source and the user-supplied inputs.

All other user PUF functionality is provided via the sNVM encryption/authentication mechanism, which uses the PUF to store the sNVM 256-bit secret key as an encrypted/authenticated key code in the pNVM. The sNVM is intended to provide a highly secure place for the user to store application keys and other sensitive data in authenticated or encrypted/encrypted form.

2.1.1.4 Design Service and Tamper Signal Interfaces

Design system services provide information about the state of the FPGA and allow the user to request the system controller to perform predefined functions. The system controller has a Design Service Interface to allow the user to run various design system services.

The design service interface includes a system service Interface (SSI) and a mailbox interface with FPGA fabric. We recommend using the CoreSysServices_PF directcore soft IP core to run various design system services. The CoreSysServices_PF IP block is connected to the system service and mailbox interfaces on one side, and an AMBA AHB bus interface on the other side. With this IP block, the user can send design service requests and receive the results via the AMBA interface. For more information, see Design System Services, page 53.

The tamper signals interface exposes the built-in tamper detection output flags and tamper response input commands to the FPGA fabric. The tamper detection flags allow the user to monitor tamper event flags and then trigger tamper responses using tamper response command inputs.

2.1.1.5 User Cryptoprocessor and NRBG Block

The user cryptoprocessor (different from the factory cryptoprocessor) is an Athena TeraFire EXP-F5200B DPA-resistant cryptoprocessor, capable of implementing all NIST Suite-B plus additional algorithms. User cryptoprocessors are available in PolarFire FPGAs that have a model number suffixed with an "S".

Many of the commonly used cryptographic operations available are certified by an independent third-party NIST-accredited security laboratory under the NIST cryptographic algorithm validation program (CAVP) scheme. This includes the AES, SHA, ECC, RSA, and DRBG implementations, providing a high level of assurance that they are implemented correctly.

CRI DPA pass-through licensing enables additional DPA-resistant high-speed cryptographic designs in the FPGA fabric, if higher performance (or a different algorithm) is needed. Microsemi prepays the Rambus-CRI royalty on these selected ("S") devices. Since the royalty is already paid, the CRI-patented techniques can be used in any design configuration downloaded to those specific devices, regardless of whether the user or third-party implementation is primarily hardware- (VHDL/Verilog) or software-based (C/assembly).

The following algorithms are supported by the TeraFire EXP-F5200B cryptoprocessor:

- TRNG (a.k.a. NRBG): SP800-90A CTR_DRBG-256, and SP800-90B(draft)
- AES-128/192/256 encryption and decryption (ECB, CBC, CTR, OFB, CFB, GCM, NIST-KeyWrap modes)
- SHA1, SHA2-224, SHA2-256, SHA2-384, and SHA2-512
- HMAC-SHA-1/224/256/384/512; AES-CMAC and AES-GMAC
- Key-Tree
- ECC: NIST P192/224/256/384/521 and Brainpool P256/384/512 curves with: KeyGen, KAS - ECC CDH; ECDSA - SigGen, SigVer, PKG, and PKV
- FFC: 1024/1536/2048/3072/4096-bits with: DSA SigGen and SigVer; and KAS - DH
- IFC: 1024/1536/2048/3072/4096/8192-bits with: RSA encryption and decryption; SSA_PKCS1_V1_5 SigGen and SigVer; and ANSI X9.31 SigGen and SigVer.

PolarFire FPGAs have a second Athena TeraFire core—a model EXP-F5200ASR cryptoprocessor—that is used by the system controller exclusively for design security. Of the NRBG plus algorithms, it implements only those needed for bitstream processing and system services.The two TeraFire cryptoprocessors can perform a large number of possible combinations and permutations of algorithms and parameters (such as key size, encryption mode, etc.).

Microsemi and Athena submitted a representative sample of the most commonly used algorithms implemented by each of the TeraFire cores, with their frequently used modes and parameters, for certification under the NIST cryptographic algorithm validation program (CAVP) scheme. All of the algorithms that were submitted passed the corresponding certifications.
The following are the TeraFire EXP-F5200B NIST CAVP certifications that have been awarded by NIST:

- AES: 3950, 3951
- DSA: 1077
- RSA: 2018
- ECDSA: 867, 868
- SHS: 3258, 3259
- DRBG: 1153, 1154
- HMAC: 2573

See the NIST CAVP website using the certification numbers above for details on which specific algorithms and modes were certified. Where a second certificate number is shown, it is for the TeraFire EXP-F5200ASR used by the system controller for FPGA design security, which is also certified ECC CDH: 790

2.1.1.6 sNVM

Each PolarFire FPGA has 56 KBytes of secure non-volatile memory (sNVM). Individual pages in the sNVM can be designated as write protected when its programming bitstream is generated, to make it easy to control sensitive data and prevent overwriting of those pages at runtime (in essence turning those pages within the sNVM into ROM). The sNVM is only accessible through system service calls. Data written to the sNVM can be protected by the PUF.

For more information about sNVMs, see Secure NVM (sNVM), page 12.

2.1.1.7 Keys, and the Private NVM (pNVM)

The private NVM (pNVM) allows protected non-volatile storage for both, factory and user keys using, PUF encrypted key codes. The keys and passcodes are stored using redundant storage such that an interruption during programming of these elements does not result in the device being left in an unusable state.

The pNVM can only be written to or read by the system controller; the data is protected from the user, thus providing a higher level of reliability and another layer of security against a malicious adversary.

For more information about Keys and pNVM, see Keys and Security pNVM, page 12.

2.1.1.8 Temperature and Voltage Sensor (TVS)

Each PolarFire FPGA device power supply is equipped with a voltage monitor. In addition, each PolarFire FPGA device includes a temperature monitor block that is used to sense the die temperature. The temperature and voltage sensor block gives digital information about the on-chip temperature and the value of the supply voltages available on the chip. The outputs of the temperature sensor and the voltage sensors are translated into standardized temperature and voltage values, and are used by the programming circuits to provide robust programming under all rated environments. The digital output of the temperature sensor is used by hardware logic comparators residing within the system controller logic, which raise an alarm (tamper flag) if the sensed temperature is not between the maximum and minimum threshold levels specified by the customer.

Note: A future version of the user guide will have more info about temperature and voltage sensors.

2.1.2 PolarFire FPGA Design Security Features

Design security assures that the user design programmed onto a device is secure, and operates as intended, for the life of the product. In the context of FPGAs, this implies a secure FPGA fabric, a secure configuration bitstream, secure update scheme for the configuration bitstream, and a secure key storage system. Microsemi PolarFire FPGAs represent the industry's most advanced programmable security FPGA.

Cryptographic design security provides information security of the configuration data. Supply chain assurance provides protection against counterfeiting, and anti-tamper protection provides physical security of the underlying data. The various security mechanisms can be combined and layered to improve and build on overall system-level security. PolarFire FPGA device security begins with silicon design and manufacturing, and continues through system deployment and operations, and finally to secure decommissioning, if required.

The following is a partial list of the design security features available in PolarFire devices:

- Cryptographic design security
  - Bitstream protection and key management
  - Integrated SRAM-PUF for ultimate key storage
  - FPGA hardware access control
  - Version control
- Supply chain assurance
  - Device certificates
  - Back-tracking prevention
  - Certificate of conformance (C-of-C)
  - Key confirmation verification protocols
- Device-level anti-tamper features
  - Resistance to differential/side-channel analysis
  - Built-in tamper detectors including voltage monitors, a temperature monitor, clock glitch detectors, voltage glitch detectors, and a protective mesh
  - Memory scrambling
  - Operational non-volatile memory integrity verification

PolarFire FPGAs fit into a larger security ecosystem that includes security-related third-party and Microsemi IP, design preparation tools in the Libero SoC PolarFire FPGA software, and the exclusive Microsemi Secure Production Programming Solution (SPPS) that provides FPGA security unrivaled in the reconfigurable logic industry.

Important attributes of the PolarFire FPGA design security features are described in the following sub-sections.

2.1.2.1 Cryptographic Design Security

Important attributes for a strong cryptographic FPGA design security solution are described in the following sub-sections.

2.1.2.1.1 Bitstream Protection and Key Management

Bitstream Protection: All PolarFire FPGA configuration bitstreams are protected with AES 256-bit encryption as specified in the NIST federal information processing standards publication FIPS-197. The bitstreams authentication is provided by a licensed protocol from CRI. This protocol uses SHA-256 and proprietary algorithms for checking authenticity in a way that resists differential power analysis (DPA) and other side-channel attacks. Under this protocol, data is never decrypted unless it has first been authenticated. Additionally, decryption keys are hashed frequently to improve DPA resistance. The Libero SoC software tool flow does not generate plain text bitstreams. A default bitstream key is used if no user keys are specified.

Key Management: Key management is often the critical link in a secure system. Multiple user-selectable keys are available and a secure mechanism is provided to update encryption keys and pass-codes. The FPGA and configuration bitstreams are protected by a cryptographic key. Key management includes securely generating, distributing, and storing keys. PolarFire devices contain factory provisioned key material that can be used to authenticate a device and provide a starting point for enrolling private user keys. Secret device keys (both factory provided and user enrolled) are stored in encrypted form in all

members of the device family. For more information about various key management features, see Cryptographic Security Features, page 11.

2.1.2.1.2 Integrated PUF for Secure Key Storage

PolarFire devices are required to store passcodes and encryption keys. To improve the security of the non-volatile storage used, all passcodes are hashed and all keys are encrypted by the SRAM PUF before storage. Decryption keys are then reconstructed by the PUF before being used. Thus, an attacker who manages to somehow measure the states of the non-volatile cells or monitor a data bus to or from the non-volatile storage cannot directly learn the actual pass-code or encryption key.

2.1.2.1.3 FPGA Hardware Access Control

PolarFire devices incorporate a number of configurable access control policies to prevent over-writing any element of a design, similar to the previous generation of Microsemi Flash device. This includes the FlashLock® passcode—also known as user passcode key 1 (UPK1)—and various FPGA lock bits that enforce more granular protection for both, general purpose memory and security segments. For more information about FlashLock passcodes and lock bits, see FPGA Hardware Access Controls, page 20.

2.1.2.2 Supply Chain Assurance

Supply chain assurance implies that the device purchased is a genuine Microsemi FPGA component that has not been tampered with during its design, manufacture, distribution, or programming. This includes assurances that the devices have not been counterfeited or fraudulently marked with characteristics other than what the device contains (that is, speed grade, revision number, etc.).

All PolarFire devices include a device certificate. The device certificate is a factory-signed X.509-compliant certificate programmed during manufacturing. When the certificate is requested, 1024 bytes are exported. As the certificate may be less than 1024 bytes, the extra bytes must be discarded by the requester. The certificate is used to guarantee the authenticity of a device and its characteristics. The certificate indicates whether the user TeraFire cryptoprocessor is available to the user (for example, if it is an "S" device). For more information about the PolarFire FPGA X.509-compliant device certificate and its uses, see Device Certificate System Service, page 30.

In addition, unique device keys enable precise controls over the customer manufacturing process to prevent over-building of customer systems at third-party manufacturing facilities, and to detect if anything other than the correct bitstream has been used to configure a device. For more information about manufacturing process controls, see Supply Chain Assurance, page 30.

2.1.2.3 Device-Level Anti-Tamper Features

PolarFire devices incorporate differential power analysis (DPA) countermeasures to protect bitstream keys from discovery, using side-channel analysis. The system controller in PolarFire FPGAs can detect a number of conditions that may indicate attempted tampering with the device. In addition, the anti-tamper system contains various voltage, frequency, and other sensors. When a tamper condition is detected, a notification event is sent to the fabric via one of many dedicated control lines. The fabric, in response to the detected tamper event, can request that the controller disables all IO pins, locks-down, resets, or zeroizes the device via the tamper response interface. Ignoring the event does not impact user design operation.

2.1.3 PolarFire FPGA Data Security Features

PolarFire devices include an Athena TeraFire EXP-F5200B DPA-resistant cryptographic processor capable of implementing all NIST Suite-B algorithms with approved key sizes, modes of operation, and other options. It can also implement many other commonly used algorithms (such as RSA), key sizes, and parameters not included in Suite-B. The user may instantiate a general purpose soft processor in the FPGA fabric for command and control, requesting the hardened Athena core to perform the desired function. For more information about data security, see PolarFire Data Security, page 67.

2.1.4 CRI DPA Patent Portfolio License

PolarFire devices with an "S" suffix in the model number, ship with a Rambus Cryptography Research Incorporated (CRI) DPA pass-through license. Users are not required to negotiate a separate license or royalty with Rambus to implement DPA-resistant designs in PolarFire FPGAs using current or future CRI DPA patents, no matter what type of implementation (hardware or software) is used. The implementation

may either be designed by the PolarFire FPGA user, or a licensed IP obtained from a CRI-authorized third-party provider. Microsemi has already paid the CRI DPA patent license royalty covering end-user applications, which is built into the price of all PolarFire FPGA "S" devices. If you have a separate DPA patent license with CRI with royalties covering systems with PolarFire FPGA "S" FPGAs in them, you should contact CRI to see how you can avoid paying twice.

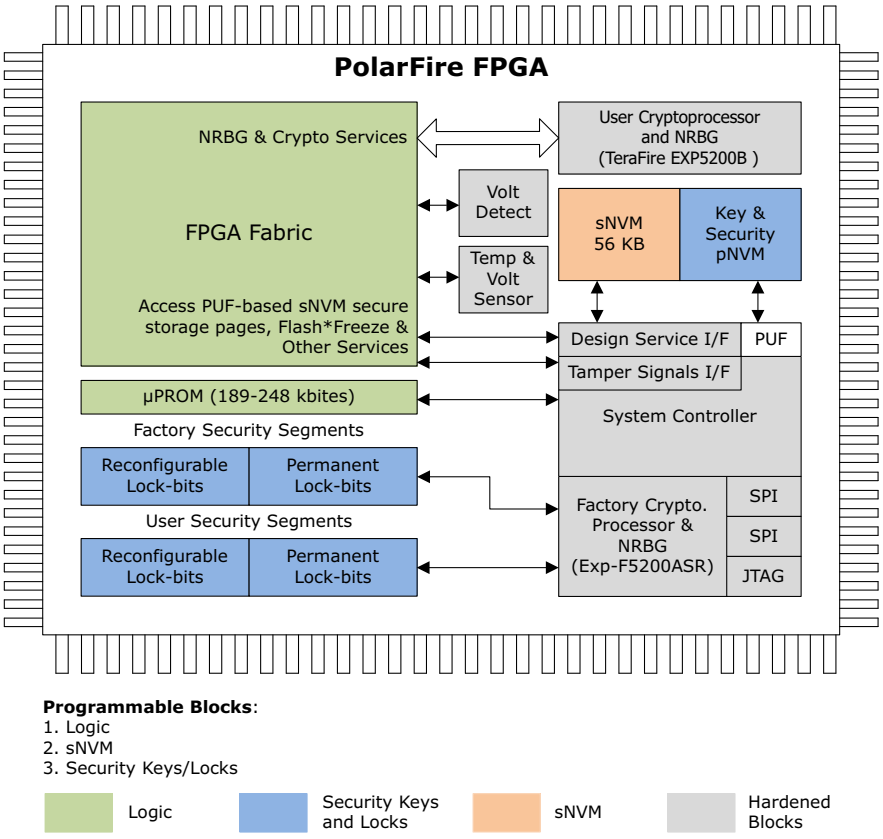
3 Cryptographic Security Features

Cryptographic design security provides the assurance that the user design on a device is secure and operates as intended, for the life of the product. PolarFire FPGAs include features that provide enhanced security during all stages of the device life cycle—from wafer probe and initial Microsemi provisioning of factory keys and certificates, through assurance that the supply chain has delivered genuine devices to the user, to user key injection and bitstream programming, to field updates, and finally to device decommissioning. This chapter describes the cryptographic security feature of PolarFire FPGAs.

3.1 PolarFire FPGA Programming Model

There are three main user programmable sub-blocks within PolarFire devices—the FPGA fabric, the sNVM and the custom security block. The following figure depicts a simplified programming model for PolarFire FPGAs.

Figure 2 • **PolarFire FPGA Simplified Programming Model**



The PolarFire FPGA fabric contains configurable user logic and Micro-PROM (µPROM). It is a large, single atomic block that can be programmed independently of the sNVM or custom security blocks, but always in its entirety, or not at all. See FPGA Fabric, page 12 for more information.

The sNVM block is a user non-volatile flash memory that can be programmed independently. The atomic unit for erasing or programming the sNVM is a 2048-bit page (including some hidden meta-data bits). See Secure NVM (sNVM), page 12 for more information.

The custom security block has several sub-blocks. It includes keys and security pNVM segments that are tamper protected and used to safely store factory and user security keys and settings. The custom security block also includes lock bits. Some of the lock bits are stored in security segments that are

programmable independently of other segments. See Custom Security Block, page 12 for more information.

All programming operations including erase, program, verify, and security key management are managed by the system controller. In programming mode, the system controller authenticates and decrypts the incoming bitstream, erases and writes the target programmable sub-blocks (memory segments and pages), and responds to other external programming-related protocols, such as key verification. The system controller and associated security hardware includes hardware-based security countermeasures to protect the device against a broad range of threats, and manages hardware-based security countermeasures throughout the rest of the device.

Programming commands and bitstreams are sent into the device either via the JTAG (slave) port, or a dedicated SPI slave port. The SPI master interface can be used to attach an external SPI flash device that holds an update and/or a golden (trusted error-recovery) configuration bitstream.This can be used either for automatic initial user programming of the PolarFire FPGA, or for (in-the-field) bitstream updates.

3.1.1 FPGA Fabric

The FPGA fabric contains configurable user logic and can be programmed independently. Being a single atomic unit (including the PROM), it is always written or erased in its entirety. If for any reason the FPGA fabric is only partially programmed, the FPGA refuses to boot up when power is applied. The device may attempt to take corrective action, such as retrying, or loading a golden image from the external SPI memory, depending on user configuration settings.

3.1.1.1 FPGA Fabric Security

The FPGA fabric interfaces with various hardened blocks. These IP blocks within the FPGA fabric provide their own security mechanisms that cannot be overridden by passcodes.

3.1.1.2 Fabric Memory Security

Fabric memories (LSRAM, µRAM, µPROM) may be accessed by the fabric or SmartDebug. The SmartDebug software uses the fabric configuration bus (FCB) to access the memory. Each fabric memory provides a security bit that prevents FCB access, thus blocking system initialization and user debug scenarios. The memory configurator allows the user to set this lock bit. A bypass mechanism is, however, provided to support zeroization. For more information, see “Embedded Memory Blocks” in the UG0680 User Guide”.

3.1.2 Secure NVM (sNVM)

The sNVM is 56 KB of user non-volatile flash memory organized into 221 pages of 236 or 252 user-accessible bytes, depending on whether the data is stored as plain text, or encrypted or authenticated data. Each sNVM page can be used as ROM, containing fixed read-only user-defined data, or as NVM that can be both—read and written to—by the user design via system services. Pages marked as ROM can only be modified by another bitstream. Data from sNVM pages stored in plain text can be used to initialize LSRAM and µSRAM blocks in the FPGA fabric during boot-up. The key used for the optional authentication or authentication and encryption of user-selected sNVM data pages is protected by the PUF.

3.1.3 Custom Security Block

The custom security block comprises:

- Keys and security pNVM segments
- Factory and user lock segments (UFS)

3.1.3.1 Keys and Security pNVM

The system controller private NVM (pNVM) allows protected non-volatile storage for both, user (indirectly) and factory usage. The following sections describe the main groups of security segments inside the keys and security pNVM.

3.1.3.2 Factory Code and Parameters Segment

This segment contains the factory code and device parameters including the factory passcode (FPK) and the device serial number (DSN).

3.1.3.2.1 Factory Parameters and Device Parameters

The factory parameters and device parameters sub-section contains calibration data used for programming and other device operations.

3.1.3.2.2 Factory Passcodes (FPK)

The factory code and parameter segment holds the unique factory passcode (FPK) in hashed form for each device. The factory passcode is used to put the device in factory test mode. The 256-bit factory passcode passkey, unique to every device, must be matched. If correct, the factory test mode is entered. This allows in-depth testing of the device by Microsemi for failure analysis. When a passcode is entered for attempted validation, the entered passcode is also hashed, and the hashed versions are compared in the hardware. By default, the factory passcode is disabled when the user keys and passcodes are programmed.This must be re-enabled manually by the user with their passcode if they wish Microsemi to perform failure analysis on a given device. The factory passcode can also be permanently disabled by the user. When that is done, no future failure analysis is possible.

3.1.3.2.3 Device Serial Number Part 1: Factory Serial Number

The device serial number (DSN) is a 128-bit unique device ID. It comprises two parts—the factory serial number (FSN) and the serial number modifier (SNM). The first part of the device serial number is the 64-bit FSN that uniquely identifies a device and is located in the factory code and parameter segment. The factory serial number is zeroized if "unrecoverable" zeroization mode is selected. For more information about zeroization, see Zeroization, page 49.

3.1.3.3 Key Segment

The factory key segment is used for storing factory key and certificate data. It is programmed during the manufacturing test and is not intended to be updated in the field. This segment may be subject to zeroization, but is recoverable. When it is zeroized, user device access is restricted until recovery procedures are executed.

3.1.3.3.1 Device Serial Number Part 2: Serial Number Modifier

The SNM is the second part of the DSN and is stored in the factory keys segment. The 64-bit SNM is zeroized during Recoverable or Unrecoverable zeroization mode action (along with all the contents of the factory keys segment). The “new” SNM is used after the Recoverable zeroization mode action, to tell if a device has been zeroized, and a new set of factory keys are installed. For more information about zeroization mode, see Zeroization, page 49.

3.1.3.3.2 Microsemi Certificate Public Key (CPK)

The Microsemi certificate public key (CPK) is a trusted immutable NIST P-384 768-bit elliptic curve public key used for checking the Microsemi signature of the device's X.509-compliant certificate—also known as the supply chain assurance certificate, SCAC. CPK is stored in the pNVM of each device that has a certificate signed by the associated Microsemi private signing key The system controller uses this to verify the signature on its own certificate every time it is requested to export it. Although CPK is a public key used in multiple devices of the same type and vintage, it is encoded by PUF to provide confidentiality and to check for authenticity, in order to prevent tampering.

3.1.3.3.3 Key Loading Key (KLK)

The default key-loading key (KLK) is the default 256-bit symmetric key. It is used to load user keys and security settings in situations where high levels of security are not required. The default key is a pre-placed factory key used to encrypt any of the flash configuration segments present in a bitstream.

This key should be used only where high levels of bitstream security are not required. One such situation could be where programming is done in a completely trusted secure facility with cleared personnel and stringent data handling and protection processes in place. Another is where the design IP is not very valuable and security is not a primary concern of the user. In this case, KLK can be selected as the root key for encryption and authentication of the bitstream component used to load the user's keys.

The 256 bit default Key-Loading Key is common to a relatively large number of devices of the same type and version, and is frequented within the programming tool software. This makes it the easiest key to use, but is not as secure as the other options, having a "software" rather than a "hardware" level of protection. After the user's security settings are loaded, the KLK is automatically disabled by a user lock bit reserved for this purpose, without any action required by the user. After this point, any programming update requires using the user keys. Typically, the user security settings (user lock security segment) and the first user key and the FlashLock passcode (in the pNVM) are loaded from the same bitstream file.

KLK is the key used to provision user keys when the optional secure production programming solution from Microsemi is not used.

3.1.3.3.4 Factory ECC Key

Factory ECC Key (FEK) is the device's 384-bit private NIST P-384 elliptic curve key loaded by Microsemi. The key is randomly generated by a Microsemi factory hardware security model (HSM) during key provisioning and is thus unique to each device. The corresponding public key—certificate public key (CPK)—is certified in the device's X.509 supply chain assurance certificate (SCAC), which is also injected at the factory.

The primary use model is to support initial loading of user keys, wherein an ECDH operation is executed to derive a shared secret key with which to encrypt a bitstream containing the user keys. Since the public key is certified by Microsemi in the SCAC, the user can be assured that the communication transpires with an authentic device and not a clone or man-in-the-middle. FEK may also be used as a signing key for device-generated certificates via a system service meant for this purpose. The authenticity of any such certificate can thus be checked using the public key from the SCAC, providing a strong cryptographic chain to Microsemi and the device PUF.

To utilize FEK and the associated public-key method to provision user keys into a PolarFire FPGA requires use of the optional Secure Production Programming Solution (SPPS) available from Microsemi.

3.1.3.3.5 Factory Key

The factory key (FK) is a 256-bit symmetric AES key unique to each device. It is a secure, quantum-safe alternative to FEK (see above) that can be used to load the user's keys, if it is selected as the root key for encryption and authentication of the bitstream component containing them. More accurately, it is used as the encryption key for an authorization code bitstream component that permits loading of the remainder of the authenticated/encrypted user bitstream (containing the user's keys and security settings) into the device. After the user's security settings are loaded, the factory key is automatically disabled for encryption purposes by a user lock bit without any action required by the user.

Since the factory key is a symmetric key, the programmer must know a derived/related key (for every device) in order to prepare bitstreams that can be decrypted by the devices, or to verify that the device is familiar with the factory key. The process is performed by installing an encrypted key database–distributed by Microsemi–into the Microsemi-supplied Secure Production Programming Solution's hardware security module (HSM) and its server. Each key database distributed by Microsemi is encrypted by a key unique to the targeted HSM, and the contents are also unique. This prevents anyone else with a key database and HSM from decrypting another user's bitstream files. The security imperative is that no copies are made of a key database (and the associated database encryption key that unlocks it) which could find a way into an adversary's hands. FK is destroyed by the *recoverable* and *unrecoverable* zeroization mode actions.

3.1.3.3.6 PUF Emulation Key

The 256-bit factory-defined PUF emulation key (PEK) is randomly generated by the device during initialization of the Factory Key segment. It is unique to each device. This key is used only in the PUF emulation protocol.

Note: The PEK is not generated by the PUF circuit every time the user recalls it, but is instead a static programmed value that is read from Flash bits.

3.1.3.3.7 PUF Personalization String

PUF personalization string (PPS) is a device-generated 288-bit key for personalizing the PUF fingerprint. The key is generated randomly by the device during initialization of the Factory Key segment, and is thus unique per device. When PPS is modified, all previously generated PUF Activation Codes, and thus any associated key codes, are invalidated. It provides an added layer of security to the PUF secure key storage mechanism since the PUF won't work without it.

3.1.3.3.8 Factory (Zeroization) Recovery Key (FRK)

The factory (zeroization) recovery key (FRK) is a 256-bit zeroization recovery key. It is a temporary key created during the zeroization recovery procedure and is destroyed when the recovery procedure is complete. It is used to ensure that a zeroization recovery key file bitstream only loads into the intended device–the zeroized device–that was verified to have sent the key-recovery request.

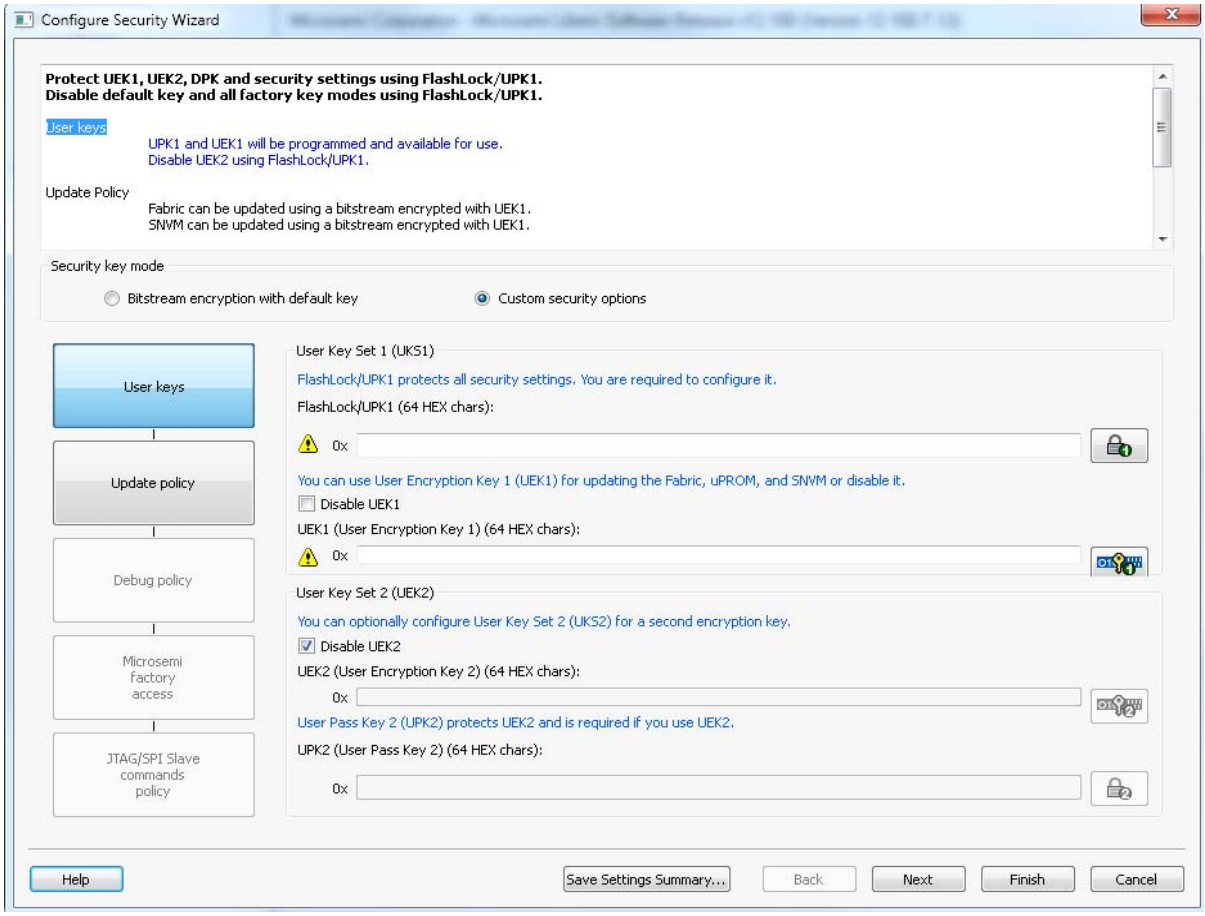
3.1.3.4 User Key Segment

The user key segment contains user keys and passcodes. In general, critical user keys and passcodes are stored using redundant storage such that an interruption during programming of these elements does not result in the device being left in an unusable state.

3.1.3.4.1 User Encryption Keys 1 and 2

User encryption key1 (UEK1) and user encryption key1 2 (UEK2) are user-defined AES-256 symmetric keys. These keys are wrapped by the PUF as key codes and then stored in the user key security segment in the pNVM. Either of these keys can be used as the root key for encrypting and decrypting bitstreams, and to authenticate them. The security policy manager in the Libero SoC software allows the user to set UEK1 and UEK2, as shown in the following figure.

Figure 3 • **Security Policy Manager in Libero SoC Software**



UEK2 can be used interchangeably with UEK1. Use of UEK2 is strictly optional. Having a second user key can facilitate use models that would be difficult to implement with just one user key. It allows you to update one key at a time, reducing the chance of corrupting both keys and making the device unusable. The secondary key can be used to program or update a subset or class of products in the field. For example, UEK1 could be common across all the devices in a project, and UEK2 could be unique in every device. Thus, when preparing an update that is only intended to go into one device, the UEK2 from that single device could be used as the root key, thus preventing the bitstream from being copied and loaded into other devices in the project accidentally or maliciously.

3.1.3.4.2 User Passcode Keys 1 and 2

User passcode key 1 (UPK1), also known as the FlashLock passcode, is the primary user passcode that overrides (unlocks) almost all non-permanent user-defined locks when matched by the user. (A few locks require clearing and reset before they are disabled.) UPK1 may be matched using either the plain text, or the one-time passcode protocol. Passcodes are never used for encryption. They are used only for escalating privileges during the session when the passcode is matched successfully. This passcode is loaded (provisioned) along with the other user keys and passcodes using an encrypted bitstream, and is stored (after being hashed) in the user key security segment. When the FlashLock passcode is subsequently successfully matched using either the plain text or one-time-use encrypted passcode protocols, it unlocks many of the security lock bits set by the user. The privilege escalation provided by UPK1 or UPK2 stays in effect only until the device is reset or power cycled.

User passcode key 2 (UPK2) is the secondary user passcode protecting the secondary user key segment UKEY2, which contains UEK2. Like all passcodes, UPK2 may be matched using the plain text or the one-time passcode protocol. When matched, it allows itself or UEK2 to be overwritten.

The Security Policy Manager in the Libero SoC software allows the user to set UPK1 and UPK2, as shown in Figure 3, page 16.

3.1.3.4.3 Debug Pass Key

Debug pass key (DPK) is a debug passcode that overrides all debug-related locks. DPK may be unlocked using the plain text or the one-time passcode protocol. It unlocks just certain lock bits related to FPGA debugging, but does not unlock as many lock bits as the FlashLock passcode does. For example, it does not allow the user to overwrite any keys, passcodes, or security settings. It stays in effect only until the device is reset or power cycled.

3.1.3.4.4 User ECC Key

User ECC key (UEK) is the 384-bit user private P-384 elliptic curve (EC) key. The key is randomly generated by the device during initial user device configuration, and is protected as a PUF encrypted/authenticated key code stored in the private-NVM. It is restricted to the legal range for P-384 private keys. PolarFire devices support JTAG and SPI instruction that can create and/or retrieve the user ECC public key (UEPK) associated with the user EC private key, UEK. When the public key is exported, it is signed by the private half of the device's factory-certified ECC key pair (by FEK). This provides a verifiable method of creating a NIST ECC P-384 key pair and securely exporting the public key in a way that can avoid man-in-the-middle attacks on it. Thus, the device can be enrolled in a user public key infrastructure (PKI) by having a certificate authority (CA) sign (certify) the exported public key. After it has been verified, the device provides proof of possession (PoP) of the private half of the key pair, and any other steps deemed necessary by the CA or local registration authority (LRA). UEK can be repeatedly regenerated until locked by the `UL_UEK_PROTECT` lock.

A future version of this user guide will have more information about this topic.

3.1.3.4.5 sNVM Master Key

The sNVM master key (SMK) is a 512-bit symmetric key for securing the content of the sNVM. It is the concatenation of a 256-bit key used for authentication and another 256-bit key used for encryption. It is randomly self-generated on each PolarFire FPGA, so is unique per device, and is stored as an encrypted/authenticated key code in the private-NVM using the PUF key-wrapping mechanism. It is the primary key used for the optional user-specified authentication or authenticated/encryption of sNVM pages using AES and the synthetic initialization vector (SIV) cipher mode. SIV mode can effectively incorporate a "tweak" that allows each page to be encrypted and authenticated with different resulting cipher text and authentication tags, even if the plain text contents are the same. For more information about details, see the sNVM section.

A future version of this user guide will have more information about this topic.

3.1.3.5 Factory and User Security Segments (UFS)

The factory and user segments are embedded within the device fabric. This segment allows controlling of the locks and hardware configuration.

3.1.3.5.1 Factory Security Segment

The factory security segment contains factory locks and factory data. The segment is programmed during manufacturing test and is not intended to be updated in the field. Access to this segment is protected by the factory passcode, FPK.

3.1.3.5.2 User Security Segment

The user security segment contains user locks. These locks are protected by the user passcode, UPK1. Some of these locks can be unlocked using debug passcode (DPK). This segment also contains the user permanent lock segment which:

- may be programmed only once.
- cannot be unlocked by a passcode.
- is immune to zeroization.

For more information about use models for the user lock bits, see FPGA Security Locks, page 21.

3.2 Bitstream Security

PolarFire devices have layered protection to ensure that the user's intent is met. These protection layers include the use of encryption to protect the confidentiality of the design IP and prevent reverse engineering, and authentication to ensure that only legitimate bitstream files are loaded by devices. All bitstreams are encrypted and authenticated to be DPA-resistant . A default encryption key is used if the user does not specify any encryption key, thus disallowing the use of a plain text bitstream.

Versioning is used to ensure that previously valid bitstreams are not reloaded in a replay attack against fielded devices. Since every device is shipped from Microsemi is already securely provisioned with a number of secret keys and a public certificate, it is possible to be assured that the devices are genuine Microsemi devices of the correct type, and to strictly control the number of devices configured with a given bitstream, thus preventing overbuilding, with no requirement for expensive trusted facilities and cleared personnel, or the physical transfer of devices between such a trusted facility and a more normal less-trusted manufacturing facility. Microsemi has implemented cryptographic protocols in these devices and associated software programming tools to ensure that all of the device programming—including initial user key injection—can be securely performed in an ordinary manufacturing environment without undue fear of interference from rogue insiders.

Note: Most of the explanation below is just for informational purposes, with enough background for expert users with special use model scenarios to be able to make informed decisions where alternatives are offered.

3.2.1 Bitstream Encryption Overview

PolarFire devices are configured using a Microsemi proprietary and confidential format bitstream file. All bitstreams are encrypted with the Advanced Encryption Standard (AES) block cipher using secret keys, and then an authentication tag is added using a type of symmetric message authentication code (MAC) based on the Secure Hash SHA-256.

3.2.2 Bitstream Content

PolarFire FPGAs are flash-based devices configured using a Microsemi proprietary and confidential format bitstream file. The PolarFire FPGA device bitstreams are divided into three main components that can be targeted during the configuration process—FPGA fabric, sNVM, and custom security (KEYS) segments.

- **FPGA fabric—**The FPGA fabric configuration component (FPGA) holds the configuration bits that configure the routing switches and look-up tables of the logic elements that define the user's design, as well as the I/O cells, embedded memories, math blocks, transceiver blocks, uPROM, etc.
- **sNVM—**The sNVM configuration component (SNVM) contains programming data for one or more sNVM pages.
- **Custom security segment component (KEYS)**—Security segments hold cryptographic keys and passcodes required for design security, and as unique device identifiers. All of the keys are stored in encrypted form and all of the passcodes are stored only after cryptographically hashing them.

Special mandatory header and footer components (BITS and EOB), program the FPGA configuration security segment. A special component called an authorization code (AUTH) is used to transport an encrypted ephemeral key into the device for use in decrypting one or more of the downstream components. A special factory keys component (FKEYS) is used for zeroization recovery after the factory keys have been erased.

3.2.3 Bitstream Encryption and Authentication

PolarFire FPGAs are configured using a Microsemi proprietary and confidential format bitstream file. Encryption of configuration data uses the AES-256 algorithm as specified in the NIST publication FIPS-197.

Authentication is provided by a licensed CRI protocol. This protocol uses SHA-256 and proprietary algorithms to check for authenticity in a manner resistant to differential power analysis (DPA) attacks. Under this protocol, data is never decrypted unless it has first been authenticated. Additionally, decryption keys are hashed frequently to provide DPA resistance at the protocol level. DPA resistance is

also included in the AES and SHA algorithms as implemented in the system controller's TeraFire cryptographic processor.

3.2.4 Key Storage

To improve the security of the non-volatile storage used, all passcodes are hashed and all keys are enciphered as key codes by the SRAM PUF before being stored. Decryption keys are then reconstructed by the PUF from the key codes before being used. Thus, an attacker who manages to somehow measure the states of the non-volatile cells or monitor a data bus to/from the non-volatile storage cannot directly learn the actual pass-code or encryption key.

3.2.5 Initial Key Loading

When a device is blank, there are no user secrets on the device that can be used to encrypt the bitstream to load user keys. On SmartFusion2 and IGLOO2 devices there were two approaches for handling this:

- A factory-defined default key (KLK) was programmed onto all devices of a similar type and version. Alternatively, a key, DFK, derived from a device-unique factory key FK, could be used. This allows a bitstream to be encrypted for a specific device but creates a logistical challenge for distributing DFK keys to users. DFK mode is supported in all SmartFusion2 and IGLOO2 FPGAs. SPPS is required to use the DFK key.
- In addition, -060, -090 and -150 capacity SmartFusion2 and IGLOO2 FPGAs can use an ECDH-based scheme whereby the device and the programmer dynamically derive an ephemeral shared secret key unique to each FPGA and each session. This allows for truly secure initial user key loading since the shared key is never exposed, it is ephemeral and only exists for a few moments during initial user key provisioning, only the Microsemi certified target device is capable of decrypting the bitstream, and it avoids the logistical challenges of distributing shared keys.Thus, this is the preferred method for initial user key loading in the devices that support it. SPPS is required.

Like the SmartFusion2 and IGLOO2 devices, PolarFire FPGA also supports the DFK and KLK key modes. The KLK key mode provides customers not concerned with bitstream security a simple method for programming devices that doesn't require the secure production programming solution (SPPS). Since all PolarFire FPGAs support the preferred ECDH key mode, DFK mode is mainly reserved as an alternative quantum-safe mode in case quantum computing makes elliptic curve cryptography obsolete. Until then, in most cases users equipped with the SPPS may want to use the ECDH-based key mode.

3.3 Key Management

All PolarFire devices are shipped provisioned with a set of unique factory keys, referred to as the factory key segment. The unique secret keys along with the device's X.509-compliant certificate enable a strong key management scheme and provides assurance that devices are genuine Microsemi devices of the correct type. The Factory unique keys are necessary since they enable a manufacturing process to authenticate a part and enable process controls over it, as necessary (for example, tying it back to a signed Microsemi Certificate, and allowing for overbuilding controls). In addition, the end users can also enroll their own security keys, thus providing complete independence from using Microsemi provided keys. These factory and user keys allow various key modes.

A future version of this user guide will have more detailed information on alternative key modes supported by PolarFire FPGAs and their programming ecosystem.

3.4 PolarFire FPGA Design Security Service Features

Design security provides the assurance that the user design programmed into a device is secure and operates as intended for the life of the product. To allow customers to build secure systems, a number of internal runtime design system services are required. For more details, see Design System Services, page 53.

4 FPGA Hardware Access Controls

This chapter describes the access control policies that can be configured to prevent overwriting any element of a design. PolarFire devices implement the following configurable access control policies to prevent overwriting any element of a design.

- **FlashLock (UPK1) passcode**—factory passcode and debug passcode security protects against unauthorized changes to security policies.
- **FPGA lock bits**—configurable hardware flags enforce more granular write-protections when compared to the passcode security, for fabric, SNVM and the security segments.

4.1 PolarFire FPGA Passcodes

PolarFire devices include the following passcodes:

- FlashLock™ user passcode 1(UPK1)
- user passcode 2 (UPK2)
- factory passcode (FPK)
- debug passcode (DPK)

Every PolarFire FPGA device is required to persistently store symmetric encryption/decryption keys (UEK1 and UEK2), passcodes, and public key pairs (FEK and its corresponding public key, CPK). To improve security of the non-volatile storage used, all passcodes are hashed and all secret keys are encrypted by the SRAM PUF as key codes before being stored. Some non-secret information also uses the PUF key code format just to take advantage of the authentication feature. The keys are then authenticated and unwrapped by the PUF regenerated intrinsic master secret before being used. Thus, an attacker who manages to somehow measure the states of the non-volatile cells or monitor a data bus to/from the non-volatile storage cannot directly learn the actual pre-image of the hashed pass-code or the plain text version of a secret key.

Note: These algorithms make use of additional device-unique secrets beside those provided by the PUF SRAM and bus-keeper elements themselves, and the key codes are stored in private memory that is unreadable by the user, providing additional layers of security for the secret keys and passcodes.

4.2 FlashLock Passcodes

FlashLock refers to a specific 256-bit passcode, also known as the user passcode1 (UPK1), which overrides a majority of the non-permanent user-defined locks. The FlashLock passcode is stored in the device in hashed format. When the passcode is re-entered and applied to the device, it is hashed and compared with the stored hashed passcode. If the hashed value of the passcode is successfully matched, it temporarily allows changes to the data items normally protected by the affected user-defined locks. The device returns to normal locked state (as defined by the non-volatile lock bit settings) on the next device or JTAG reset, or power cycle. The FlashLock passcode is the basic level of security. The higher levels of security may be obtained by using the permanent lock features of the devices or by setting additional individual lock bits.

Note: The user must be careful when a FlashLock passcode is matched and before the device is reset or power cycled. During this time, not only is a new authenticated bitstream loaded, but the device can also be erased, and other lock bits are temporarily unlocked.

The use of passcodes to unlock a device is not recommended in untrusted locations. In a use model, where field updates to security segments are required in untrusted locations, we recommend that in order to prevent overwriting, only bitstream authentication features be used. In this scenario, the lock bit that prevents overwriting is not used, therefore matching the FlashLock passcode is not required in the field.

The Security Policy Manager in the Libero SoC software is used to apply the FlashLock passcode for the user design. We recommend, however, to allow the secure production programming solution (SPPS) with a hardware security module (HSM), when available, to select the FlashLock passcode (and all other user keys and passcodes) rather than importing it manually or generating it on a less-secure general-

purpose workstation. The HSM generates a high-quality 256-bit true random bit string, and stores it securely within its certified hardware security boundary. The random bit string, if exported for storage on the host workstation, injection into the device (as a bitstream), or for unlocking the device (using the one-time passcode protocol), is strongly encrypted. The HSM does not export secret keys or passcodes in plain text form, even to the host workstation.

The FlashLock passcode may be matched using either the plain text or the one-time passcode protocol. With previous families (Smartfusion2 and IGLOO2), matching a passcode always unlocked all locks in its associated class. For PolarFire devices, the one-time passcode protocol is enhanced to permit selective unlocking of individual locks. The plain text user passcodes are supported to maintain legacy functionality.

A plain text passcode does not provide selective unlocking. Furthermore, a plain text passcode is subject to possible monitoring attacks. Once known, it can be used to escalate privileges associated with that passcode forever (unless plain text passcodes are prohibited by the device's security settings). The use of plain text passcodes is therefore discouraged.

When a one-time passcode is used, a subset of locks to override may be specified in the one-time passcode protocol. During debugging, it is recommended to use the one-time-use encrypted passcode protocol only (versus the deprecated plain text passcode protocol).

The HSM used in SPPS flow supports the one-time-use passcode generation. The one-time-use passcode protocol can be used to match the FlashLock passcode (without revealing its value outside the SPPS HSM). One possible use of the FlashLock passcode is to have it allow bitstream updates again, even if overwriting the FPGA fabric or sNVM was disabled by locks in the security policy stored on the device.

4.3 FPGA Security Locks

The factory and user lock segment (UFS) in PolarFire devices includes factory security segment and user security segments that holds various lock bits. Some of the lock bits act as access control (read or write) bits to the security segment to which they are applied. The factory lock bits are set and locked in the factory security segments before shipping the parts. The user lock bits are set and locked in the user security segments. Some factory lock bits prohibit the same function as a user lock bit. In this case, if either one is set, the function is disabled. The user lock bits can be temporarily unlocked using the appropriate passcode assigned to that bit. Some bits can only be modified by erasing or overwriting the security segment to which they belong using an encrypted/authenticated bitstream. If lock bits are unlocked using a passcode, it is just temporary, until the next device reset, JTAG reset, or power-down. Any permanent change to the user and factory security segment segments must come from a bitstream and take place after the reset, or at the next power-up cycle.

Although a lock bit may be referred to in the singular in this document, that is just a reference to its logical existence. All lock bits are actually stored with physical redundancy. One form of this redundancy is the extensive use of cryptographic digests. The most important lock bits, from an anti-tamper perspective, also use parity bits to detect any loss of integrity. These bits are monitored continuously during run-time, and generate a tamper detection flag immediately if a tamper event is detected. This process is independent of whether there is any programming or security-related operation going on in the FPGA device. These lock bits, and also less important bits, are monitored at the time they are consumed, by recomputing and comparing a digest value before using the stored data.

The Security Policy Manager (SPM) in the Libero SoC software is used to apply these lock bits.

4.3.1 Factory Security Locks

This group of lock bits prevents modification of the factory security segment and is set by Microsemi. The factory passcode (FPK) is required to change these lock bits. The matching of the factory passcode is, however, always locked by a user lock bit that is set when the user programs the user keys, passcodes, and security settings. In other words, when the user programs a device, the factory passcode is automatically disabled.

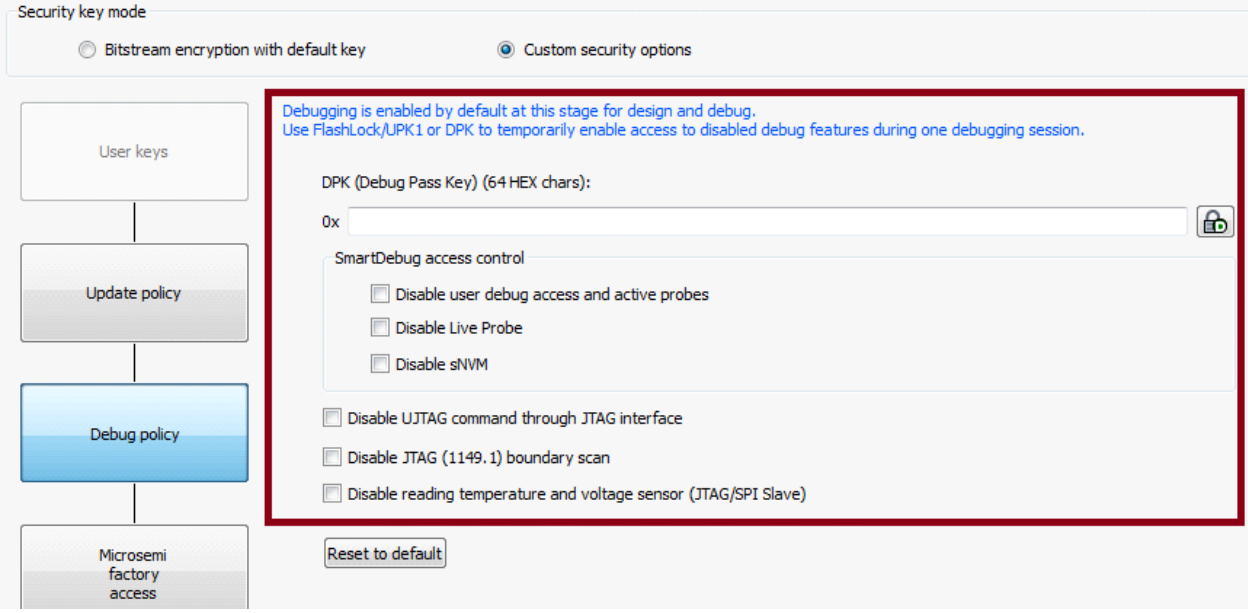
4.3.2 User Debug Security Locks

The debugging features are deactivated using these user debug lock bits. These lock bits enable the following:

- Disable the user debug access, except live probes (independently protected).
- Disable access to live probes.
- Disable sNVM
- Disable the User JTAG interface to the FPGA.
- Prevent the JTAG Boundary Scan Register (BSR) from observing and controlling the device IO pins and fabric IO control signals. JTAG instructions which select the BSR still has access to the shift register, but:
  - the Capture-DR state captures zeroes.
  - the boundary scan mode signals are not asserted
- Disable external access to the temperature and voltage monitor registers via the JTAG and SPI slave interfaces.
- Enable the user JTAG security monitor.

The debug lock bits can be temporarily overridden by matching the FlashLock Passcode (UPK1) or the Debug Passcode (DPK). The **Configure Security Policy Manager>Debug Security Policy** settings in the Libero SoC software allows the user to lock the debugging features, as shown in the following figure.

Figure 4 • **Security Policy Manager in Libero SoC Software**



4.4 User Lock Erase/Write Disable Lock Bits

This group of locks prevents erasing and writing of the user security segments that include the user encryption key 1 (UEK1), user encryption key 2 (UEK2), and the user lock security segments. These locks are automatically set when UEK1 and UEK2 are set by user. The lock for UEK1 and user lock security segments can be temporarily overridden in the event of a FlashLock (UPK1) passcode match. The lock for UEK2 can be temporarily overridden by UPK2. When user security segments locks are set, SPM locks the user security segment against erasing, overwriting, and verifying after they are first programmed, without any action on the part of the user.

4.4.1 Fabric and sNVM Update (Programming/Erase) Lock Bits

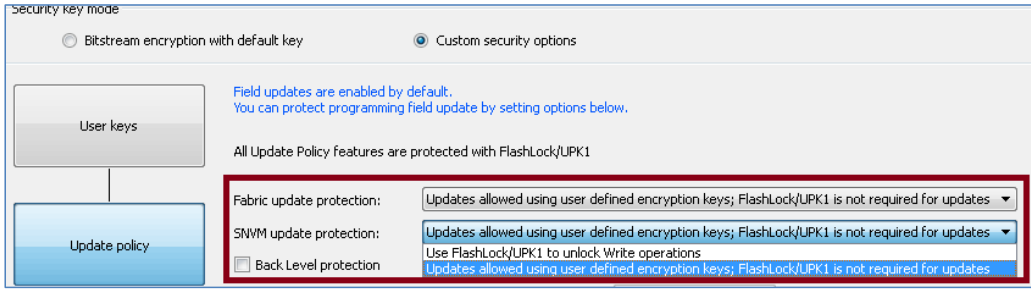
These lock bits prevent FPGA and sNVM from being erased and written with a new encrypted or authenticated bitstream. These locks can be temporarily unlocked by matching the FlashLock (UPK1) passcode.

In the Security Policy Manager in Libero SoC software, the Update Policy is used to apply the FPGA fabric and sNVM update protection lock bits. When the lock is set, you must use the FlashLock passcode to unlock the update operation. When the lock is cleared (either because it was never set, or was temporarily unlocked by a passcode match), and you have provisioned a user encryption key (UEK1 or UEK2), you need to use one of the user encryption keys to update either the fabric or the sNVM.

To set the FPGA fabric update protection lock bit, select **Use FlashLock/UPK1 to unlock Write operations** under **Fabric Update Protection**.

To set the sNVM update protection lock bit, select **Use FlashLock/UPK1 to unlock Write operations** under **SNVM update protection**.

Figure 5 • **Update Policy Page in Libero SoC software**



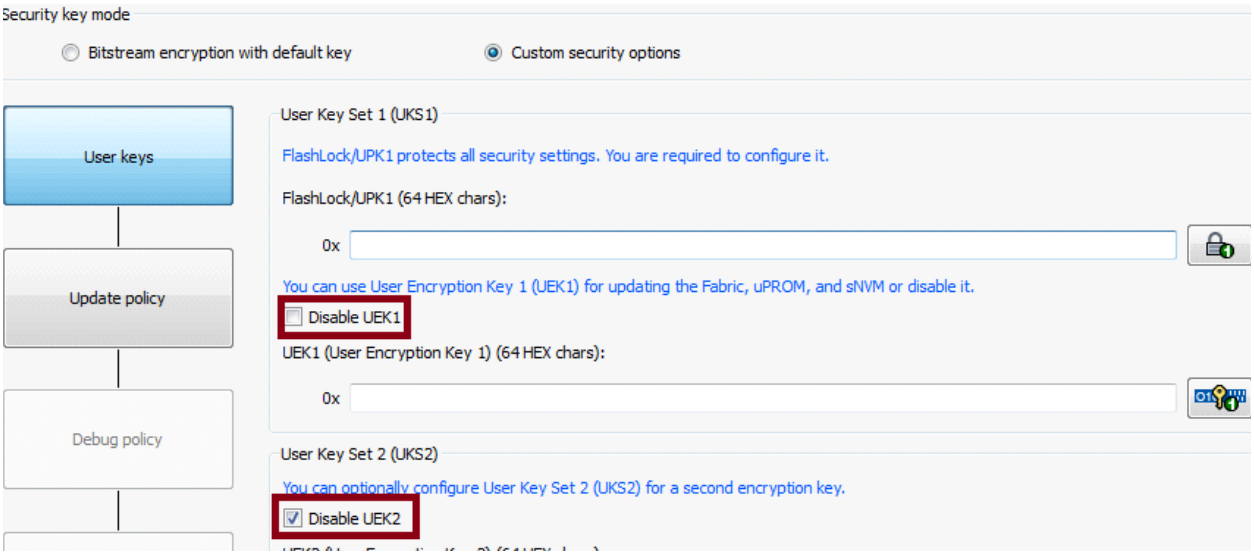
4.4.2 Key Mode Lock Bits

Key modes are used to select the root key and algorithm used to encrypt and/or authenticate data in a PolarFire FPGA protocol, for example, the bitstream loading protocol. Not all key modes are applicable to every protocol, and some combinations are only supported if the optional secure production programming solution (SPPS) is used. Any key mode can be disabled using user (or factory) lock bits.

In a new device, any one of the supported factory key modes may be used to load the initial user keys in encrypted form. After the user keys are loaded, all the factory key modes are automatically disabled, leaving only the user key modes in operation. Thus, any subsequent bitstream update must be done using the user keys. Key modes associated with keys that are not loaded are also automatically disabled. If all key modes are disabled, this is effectively another layer of security to prevent bitstream updates. The key mode lock bits are not temporarily unlocked by the FlashLock passcode as with most other lock bits. It is required to match the FlashLock passcode to allow the key mode lock bits to be erased, after which they can be reprogrammed by a new bitstream.

Use the **Security Policy Manager** in the Libero SoC software and select **User keys: page to disable the UEK1 and UEK2**, as shown in the following figure.

Figure 6 • **Disable Key Mode in Libero SoC Software**



4.4.3 Plain Text Passcode Disable Lock Bit

This lock bit disables all plain text passcode matching and forces the user to use the single-use encrypted passcode protocol. It affects all the passcodes in the device:

- Factory Passcode (FPK)
- FlashLock Passcode (UPK1)
- User Passcode 2 (UPK2)
- Debug Passcode (DPK)

Transmitting any crypto variable (CV) in plain text is considered unsafe. We highly recommend using this lock bit. The advantage of the single-use encrypted passcodes is that the communication with the device when the passcodes are being transmitted into it may be monitored by an adversary (for example, rogue insider) without exposing the passcode value. Also, additional keying material must be known by the challenged entity to ensure it knows the passcode, increasing the security for a successful match.

The secure production programming solution (SPPS) is required to use the one-time passcode protocol. It computes the correct encrypted one-time response using the required secret passcode and key and transmits it to the FPGA. The plain text passcode and key are never exposed outside the hardware security boundary of the SPPS HSM.

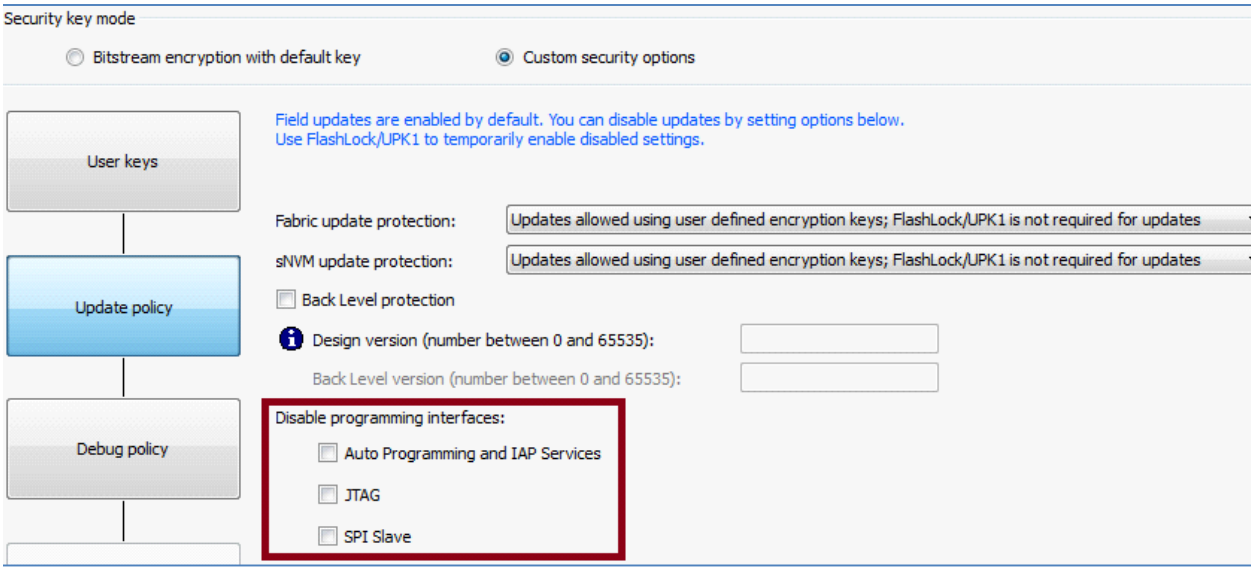
4.4.4 Programming Port Lock Bits

Several user lock bits are available to block access to programming through specific programming ports. These lock bits include:

- **Autoprogramming and IAP disable lock bit**—This lock disables IAP bitstream functionality, including recovery and auto-programming. SPI initialization functionality is not affected.
- **JTAG TAP disable lock bit**—This lock completely disables the JTAG interface, rendering non-compliant with IEEE 1149.1. If the device is part of a serial JTAG chain, the chain is broken. The System Controller TAP controller is held in the Test-Logic-Reset state. The JTAG pins themselves remain active but the TAP controller does not respond to activity on these pins.
- **SPI Slave disable lock bit**—This lock completely disables the SPI slave interface. Any activity on the SPI pins is ignored.

Use the Security Policy Manager in the Libero SoC software and select **Update Policy** page to apply the programming port lock bits, as shown in Figure 7, page 25.

Figure 7 • **Protect Programming Interfaces in Libero SoC Software**



4.4.5 Programming Action Protection Lock Bits

There are several user lock bits to block the various programming actions on the bitstream.

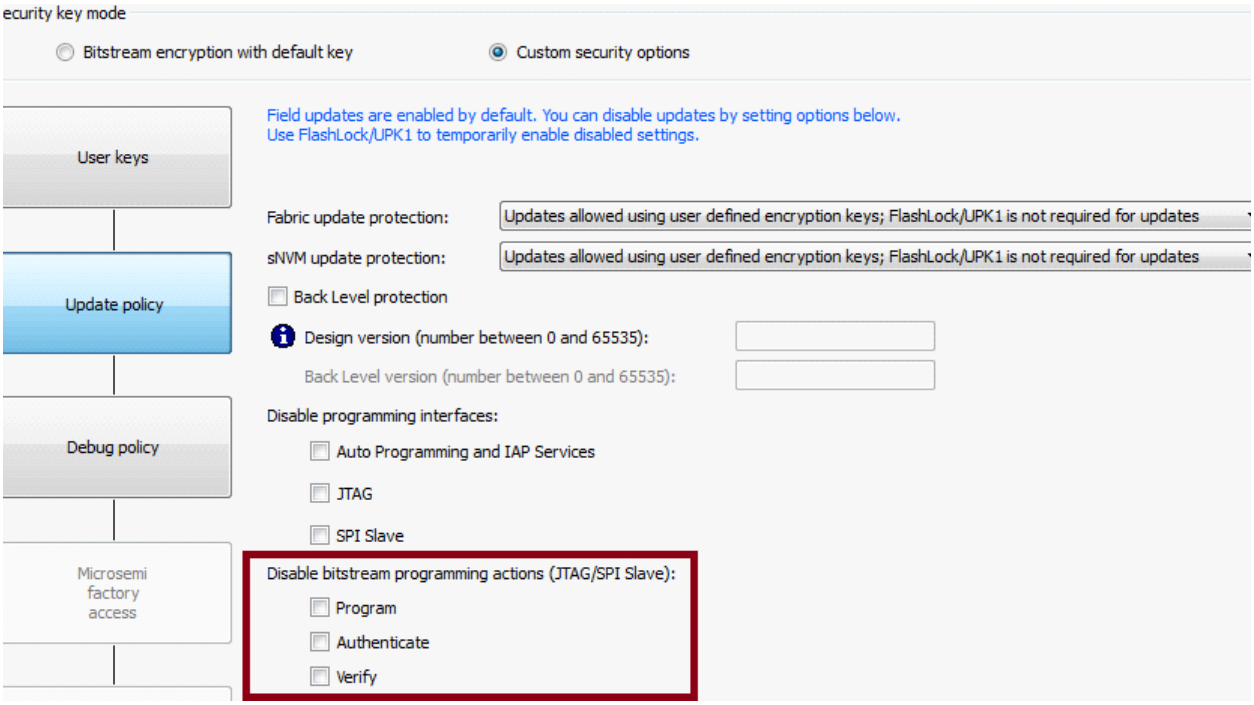
Note: Bitstream can be used in any of three modes—authenticate, program, and verify. Each mode is protected by one of the following dedicated lock bits:

- **Bitstream Program Disable**—This lock disables all bitstream program operations. If User Passcode Disable is also set, then the device content cannot be modified by the system controller.
- **Bitstream Verify Disable**—This lock disables all bitstream standalone verify operations. Verification is executed only during programming operations. Digest checks are not affected. There is an overall lock bit that disables digest checks initiated through the external JTAG and SPI slave ports. Therefore, if this lock bit is set, no verification by any method can be performed and any attempt to do so fails unless the FlashLock passcode is used first to unlock it.
- **Bitstream Authentication Disable**—This lock disables bitstream authentication mode.

Note: The Bitstream Authentication Disable lock bit does not affect the required authentication checks used in the program and verify modes.

Use the Security Policy Manager in the Libero SoC software and select the **Update Policy** page to apply the programming port lock bits, as shown in Figure 8, page 26. These locks only apply when bitstreams are loaded via the JTAG or SPI slave interfaces, and they do not apply to IAP functions.

Figure 8 • **Protect Bitstream Programming Action in Libero SoC Software**



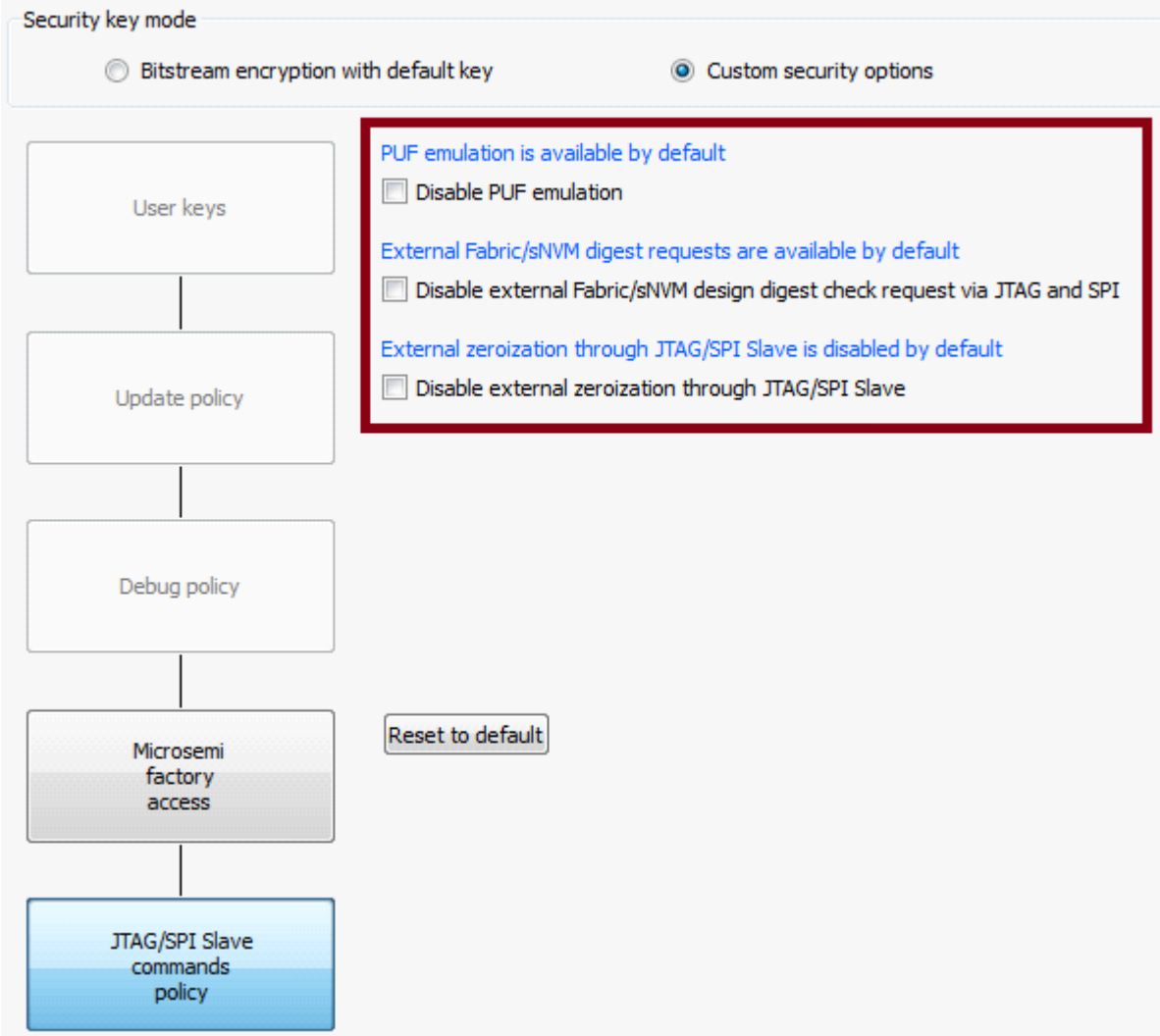
4.4.6 JTAG/SPI Command Policy Lock

The JTAG/SPI Command Policy lock bits enable the following:

- **User PUF emulation disable lock bit**—This lock disables all external access to the PUF emulation protocol.
- **External Digest request disable lock bit**—This lock disables digest checks initiated through the external JTAG and SPI slave ports. Export of stored digests is not affected.
- **External Zeroization disable lock bit**—Disable zeroization request from JTAG/SPI slave.

These locks can be temporarily overridden in the event of a FlashLock (UPK) passcode match. Use the Security Policy Manager in the Libero SoC software and select the **JTAG/SPI Slave command policy** page to apply the JTAG/SPI command policy lock bits, as shown in Figure 9, page 27. These locks only apply when bitstreams are being loaded via the JTAG or SPI slave interfaces, and they do not apply to IAP functions.

Figure 9 • **User Command Policy Page in Libero SoC Software**



4.5 Permanent Locks

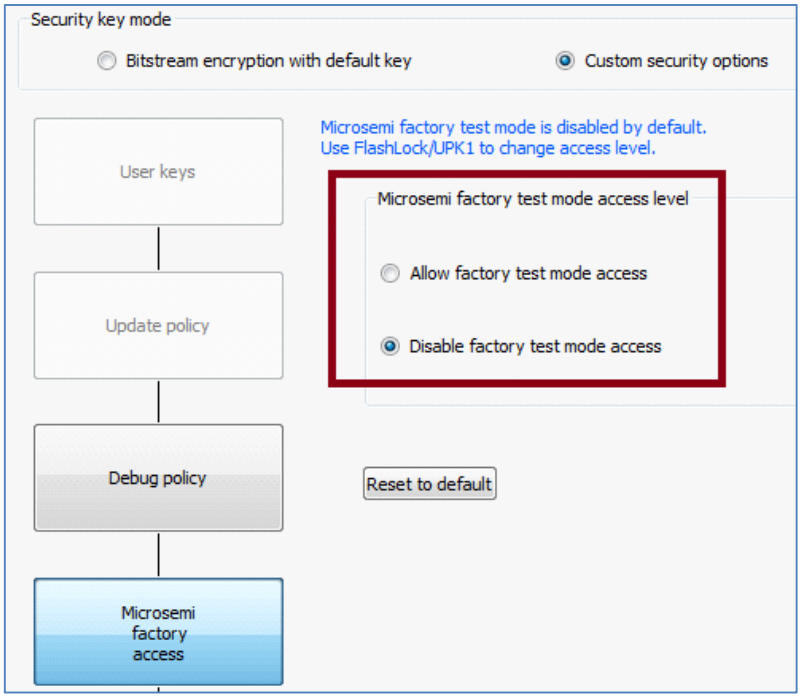
The FlashLock passcode (UPK1), the user passcode key2 (UPK2), the debug passcode (DPK) and Factory Passcode (FPK) can be permanently locked out using these user-set lock bits.

The user can permanently lock (disable) the factory passcode (FPK).

To permanently lock FPK, open **SPM** in the Libero SoC software and select **Permanently protect factory test mode access using FlashLock/UPK1** option under "Microsemi factory access" page, as shown in Figure 10, page 28.

The default is for the factory passcode to be automatically disabled when the user keys are programmed, but this is reversible if the user matches the FlashLock passcode and changes the setting to re-enable the factory passcode. However, if the factory passcode is permanently disabled, it is impossible for the user to allow factory test mode access anytime thereafter. Microsemi requires access to perform failure analysis, therefore, if the factory passcode is permanently disabled, any future failure analysis requested on that device is not possible.

Figure 10 • **Microsemi Factory Access Page in Libero SoC software**

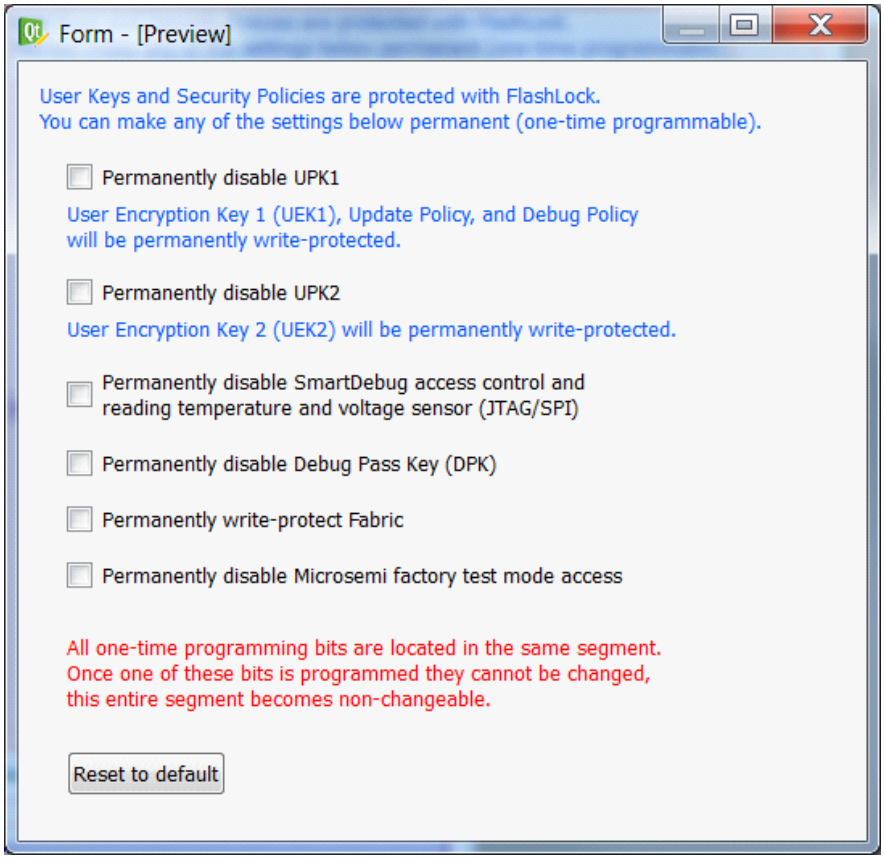


4.6 One-Time Programming (OTP) Locks

The One-Time Programming (OTP) mode in PolarFire devices, enables additional optional, layered, security settings. To set up OTP mode, open the OTP SPM tool and configure the security settings. The OTP lock bits enable the following:

- permanently disable UPK1 passcode matching
- permanently disable UPK2 passcode matching
- permanently disable all debug functions including all user debug and live probe
- disable debug passcode (DPK) matching
- disable modifications to the fabric. If the device is zeroized the fabric cannot be subsequently reprogrammed
- disable factory passcode matching to prevent future factory access to the device

Figure 11 • **OTP Settings in Libero SoC Software**



Note: The permanent locks including OTP locks cannot be unlocked by passcodes. Permanent locks can only be written once as a set, and are immune to zeroization operations.

5 Supply Chain Assurance

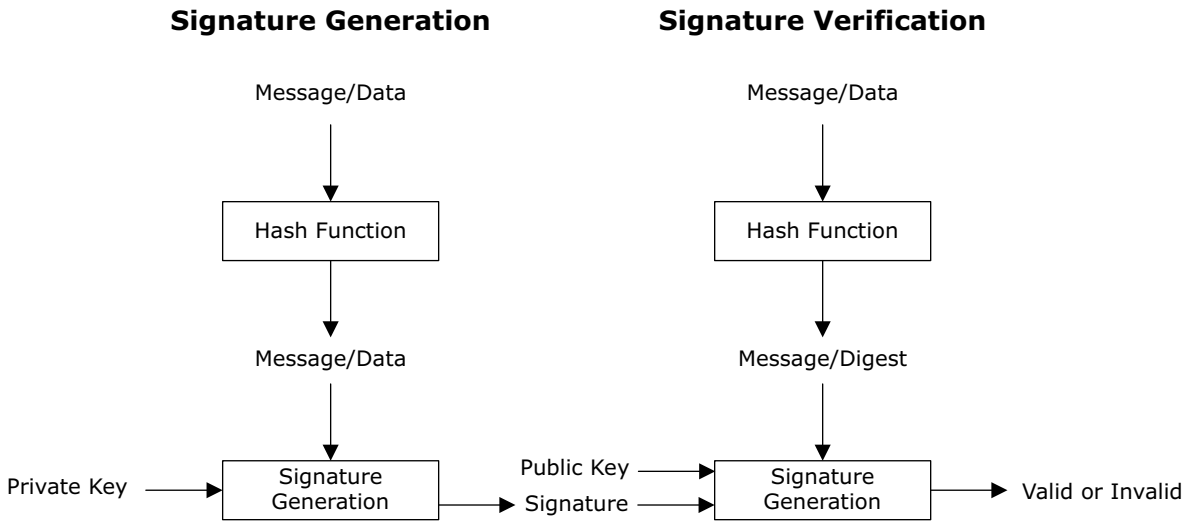
This chapter describes the security measures incorporated in each stage of the PolarFire FPGA device life cycle, from manufacturing to deployment and field upgrades, and finally, to decommissioning.

5.1 Supply Chain Assurance Certificate

Counterfeiting in electronic parts takes various forms, including copying designs at the transistor level, black-topping, re-marking devices to misrepresent used devices as new, changing date codes, improving the speed grade or the temperature grade, increasing the alleged screening level, among others.

To prevent counterfeiting and fraud, every PolarFire FPGA device incorporates an X.509-compliant supply chain assurance certificate (SCAC), stored in the device sNVM. The digital certificate includes a digital signature, and an electronic analog of a written signature. The digital signature can be used to provide assurance that the claimed signatory signed the information. In addition, a digital signature can be used to detect whether or not the information is modified after it is signed. The following figure shows the standard digital signature processes.

Figure 12 • **Digital Signature Processes**



The digital certificate in PolarFire devices cryptographically binds the device serial number and date code, enabled features, the device's secret factory key, and a digital signature from Microsemi in a way that can be validated internally by the device and externally by the user. Any mismatch between how the device is represented by its shipping paperwork, or the label printed on its surface and the digital certificate indicates the possibility of counterfeiting fraud.

Most devices ship with a certificate that may indicate a super-set for the temperature range and/or speed grade. If an exact match between the certificate and actual model number values for these parameters is desired, please contact Microsemi Sales.

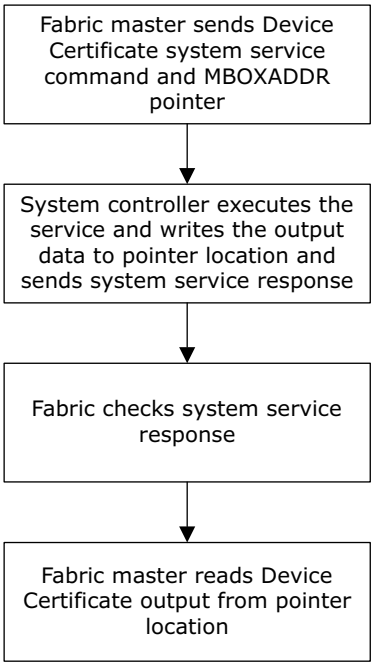
5.1.1 Device Certificate System Service

You can run the digital certificate system service and fetch the device supply chain assurance certificate (SCAC) from the pNVM. The system services are System Controller actions initiated by asynchronous events from the user design via the System Controller's system service interface (SSI). For more information, see Design System Services, page 53. When the SCAC is exported, the DSN and public key are checked for consistency against the actual values encoded in the device. Internally regenerating the public key from the private key adds an additional layer of protection against cloning, since the encrypted value of the private key and its authentication tag depend on the PUF, thus deeply binding the private key to that particular physical device. The Microsemi signature on the certificate is also checked using the

immutable trusted Microsemi public key (CPK) stored in the private NVM for this purpose. The exported certificate data is 1024-bytes, but the actual certificate size may be smaller. Any extra bytes should simply be discarded by the user. In the event of an error, the certificate content is still returned for inspection.

The following figure shows the main steps for running the Device Certificate system service. For more information about running system services, see Design System Services, page 53

Figure 13 • **Device Certificate System Service Flows**



The following table lists SSI system service descriptors for device certificate service requests.

Table 1 • **System Service Descriptor for Device Certificate Service Request**

System Service Descriptor	
15:7	6:0
MBOXADDR[10:2]	03H

The following table specifies the device certificate service mailbox format.

Table 2 • **Device Certificate Service Mailbox Format**

Offset	Length (bytes)	Parameter	Direction	Description
0	1024	CERTIFICATE	Output	Device Certificate

The following table lists and describes the device certificate service states.

Table 3 • **Device Certificate Service Status**

STATUS	Description	Note
0	Success	Certificate is valid and consistent with device
1	Device mismatch	Public key or FSN do not match device
2	Signature invalid	Certificate signature is invalid
3	System error	PUF or storage failure

The certificate is validated by checking its signature using the HSM public key, certificate public key (CPK).

5.1.1.1 Anti-Cloning Protection

The supply chain assurance certificate (SCAC) provides protection against simple re-marking of devices. An actual clone, however, would not be detected since the certificate, which is public information, could be copied from a genuine device onto the clone.

To make life more difficult for the cloner, the user can require the device to provide proof of possession (PoP) of the private half of the ECC key pair, which is certified by the Microsemi-signed X-509-compliant certificate This is done either by employing a challenge-response protocol (see the key verification JTAG or SPI protocols), or by having the device digitally sign something fresh and then verifying the signature using the public key (see the digital signature system service).

This would then require the clone to have knowledge of the device's private key, which is protected by PUF encryption and stored in private flash; a digital copy of the public certificate alone would no longer be sufficient to prove the device's identity. This protocol thus improves confidence in the authenticity of a device.

5.1.2 Device Integrity Protection

It should be possible to distinguish new PolarFire FPGA devices from a previously used or tampered device. The used device has obvious implications for device quality and endurance. Attempts may be made to extract the device's unique factory keys with the intention of later intercepting or forging communications with the device. To mitigate this class of attacks, PolarFire devices employ a mechanism to mark used devices. This requires reading the device integrity bits.

When the user receives a new device, he/she may examine the device integrity bits and check that they are still intact. The device integrity bits are invalidated in the following events:

- loading a bitstream in Program mode,
- failed execution of the factory passcode protocol

Device integrity bits cannot be modified by any user operation. Zeroization may change the state of the device integrity bits, but cannot restore them to their pristine state.

A future version of this user guide will have more information about this topic.

5.1.3 Certificate of Compliance (C-of-C)

As new devices can be programmed by whoever possesses them, the devices must either be initialized using user keys in a trusted facility with vetted personnel, or another method should be used to ensure that the correct keys, security settings, and user-supplied design are programmed into the devices. The best practice would be to bring the fully-assembled and programmed systems to a trusted facility where the programming could be verified, before they are put into any sensitive applications. Either of these approaches—using a trusted facility to pre-load keys, or afterwards, to verify programming—requires extra time and expense. This may include the cost of maintaining such facilities and staff, and the inconvenience of not being able to put otherwise finished systems into service until additional steps have been performed.

Microsemi PolarFire FPGAs offer an alternative approach that uses cryptographic techniques to provide assurance that they are programmed correctly, and not with some malicious entity's keys instead of the user's keys, or with (intentionally) wrong security settings, or with a different design than intended (perhaps containing a Trojan Horse).

During programming, the device can generate a short message called a certificate of conformance (C-of-C). This includes keyed digests (message authentication code tags) for each bitstream component programmed. The pre-image data includes the data programmed by that bitstream component, and the device serial number. This ensures that the C-of-C tag from each device is unique, even if the programmed data is the same. The key makes the tag impossible to forge.

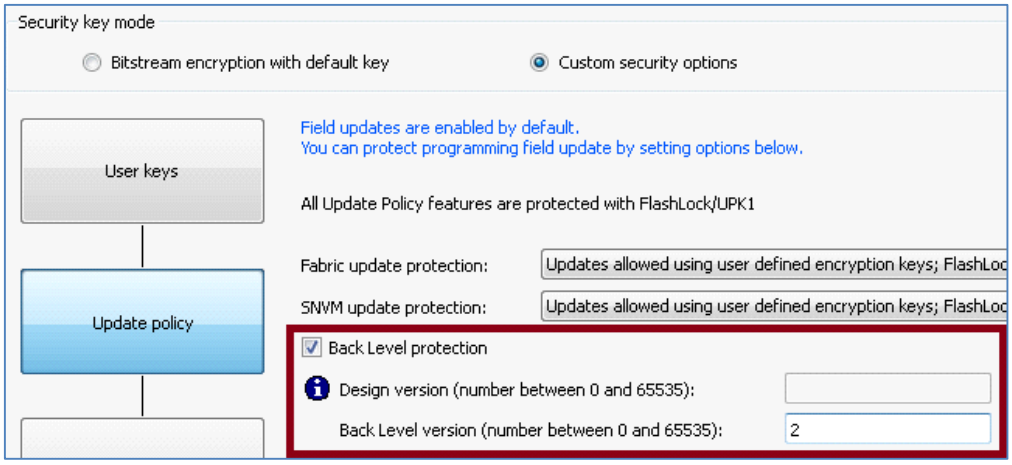
The SPPS software can validate the returned C-of-C messages from each device and report this in secure log files. This is one aspect of keeping tight accounting control over the number and identity of the parts produced, the scrapped parts, and so on. The C-of-C proves that each component is programmed with the expected data. Assuming the C-of-C is read from a device that is write protected where it cannot be overwritten without the use of secret passcodes or keys, C-of-C provides strong assurance that the device is programmed correctly.

The advantage of the C-of-C approach is that it provides this assurance minus the expense of shipping parts or systems around the world between less-trusted assembly facilities and more-trusted facilities where additional programming or verification steps must be performed, and it may even eliminate the need for the more expensive facilities. Programming can be performed in a less expensive facility without the risk of undetected tampering of the programming data. Generating and confirming the C-of-C is very efficient, and adds almost nothing to the programming time. It is completely automated by Microsemi programming software tools and HSMs. The C-of-C check needs to be requested by the user, and is the default option for the HSM-assisted secure production programming solution.

5.1.4 Back-Level Protection (Versioning)

PolarFire devices allows protection against a replay attack, where an earlier form of a bitstream (one perhaps with security vulnerabilities) may be reintroduced to gain information about a system. The user can assign a version number to each configuration bitstream, and add a back-level version, using the **Security Policy manager** > **Update Policy** page in the Libero SoC software tool, as shown in the following figure.

Figure 14 • **Back Level Protection**



The back-level version value restricts the design version that the device accepts as an update. The back-level version is set at, or higher than, all of the (old) versions that the user wishes the device to reject. So, only (new) programming bitstreams with a Design Version strictly > the current Back Level Version previously stored on the device are allowed for programming. The new back-level version programmed

along with an accepted bitstream affects any future bitstreams, and can be higher or lower than the back-level it overwrites, at the user's discretion. Back-level protection is secured by FlashLock/UPK1, which can be used to bypass it. Digests

Digests are used for protecting data integrity. In the factory and user security segment, each logical page contains an automatically generated digest calculated dynamically at the time of programming the data to be written. For the FPGA fabric, the digest includes an overall value covering the data to be programmed. In addition, digests are calculated and stored for the sNVM pages marked as ROM. The digests can be verified on-demand by the user, either internally using a system service, or externally using a programming instruction. In addition, the user can automatically run digest checks on each power-up. The following section describes the various options to run the digest check.

5.1.5 Power-On Digest

The PolarFire FPGA device may be configured to perform automatic digest checks while powering up the user design (at system boot or after programming) to check the integrity of the selected memory. The user can specify which digest to check. If any of the selected digest checks fails, a tamper event is generated.

The pNVM and sNVM can be read as many times as desired without any noticeable effect on data reliability. If, for example, the first-stage boot code for a soft CPU is stored in the sNVM, then the power-up integrity check could be used to automatically provide—on each power-up event—a high level of assurance that the code had not been changed, either through a natural or malicious event, since the digest was stored.

Note: A read-endurance limit specifies how many times a digest of the FPGA fabric can be run before the long-term reliability of the FPGA configuration data could be affected.

See the PolarFire FPGA data sheet for more information about the FPGA configuration memory endurance limits. Therefore, depending upon how the system is deployed and used (for example., how often it is powered-up), the on-demand digest check (see below) may be more appropriate for testing the integrity of the FPGA fabric.

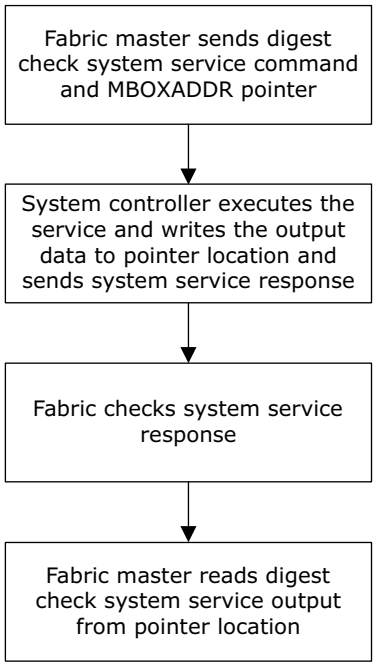
A future version of this user guide will have information on how to set up the power-on digest check.

5.1.6 On-Demand Digest Check

This digest check recalculates and compares digests of selected non-volatile memories. If the fabric digest is to be checked, then the user design must follow all prerequisite steps for the Flash*Freeze service before invoking this service.

The following figure shows the main steps for running the digest check design system service.
For more information about running system services, see Design System Services, page 53.

Figure 15 • **Digest Check Design System Service System Service Flows**



The following table lists the digest check design system service.

Table 4 • **Digest Check Request**

15:7	6:0
MBOXADDR[10:2]	47H

The following table lists the digest check design system service mailbox format.

Table 5 • **Digest Check Service Mailbox Format**

Offset	Length (bytes)	Parameter	Direction	Description
0	2	OPTIONS	Input	Digest options

The following table lists digest check design system service options.

Table 6 • **OPTIONS[15:0]**

OPTIONS[15:0]	Name	Description
0	FABRIC	Fabric digest
1	CC	UFS Fabric Configuration (CC) segment
2	SNVM	ROM digest in SNVM segment
3	UL	UFS UL segment
4	UKDIGEST0	UKDIGEST0 in User Key segment (includes SRAM PUF activation code and device Integrity bit)
5	UKDIGEST1	UKDIGEST1 in User Key segment
6	UKDIGEST2	UKDIGEST2 in User Key segment (UPK1)

Table 6 • **OPTIONS[15:0] *(continued)***

OPTIONS[15:0]	Name	Description
7	UKDIGEST3	UKDIGEST3 in User Key segment (UEK1)
8	UKDIGEST4	UKDIGEST4 in User Key segment (DPK)
9	UKDIGEST5	UKDIGEST5 in User Key segment (UPK2)
10	UKDIGEST6	UKDIGEST6 in User Key segment (UEK2)
11	UPERM	User Permanent lock (UPERM) segment
12	SYS	Factory Key Segments and Factory Key Segments

If the fabric digest (FABRIC) is '1,' then the FPGA fabric is placed in Flash*Freeze state, but the System Controller continues to operate as normal. On completing the fabric digest, Flash*Freeze is automatically exited and the user design is re-enabled using the same procedure described for the Flash*Freeze service. If the fabric digest is '0' then there is no need for a Flash*Freeze shutdown and the fabric continues to operate as normal during the requested digest calculations.

A failure of any digest results in the DIGEST tamper flag being triggered. The following table lists the Digest Check status.

Table 7 • **Digest Check Status**

STATUS	Description
DIGESTERR	See Table 8, page 36

If a digest mismatch occurs, DIGESTERR indicates which of the selected digests are in error. The following table lists and describes the DIGESTERR error codes.

Table 8 • **DIGESTERR Error**

DIGESTERR[i]	Name	Description
0	FABRICERR	Fabric digest error (0 if CHECKFABRIC is '0')
1	CCERR	UFS Fabric Configuration (CC) segment error
2	SNVMERR	ROM digest in SNVM segment error (0 if CHECKSNVM is '0')
3	ULERR	UFS UL segment error
4	UK0ERR	UKDIGEST0 in User Key segment error (includes SRAM PUF activation code and device Integrity bit)
5	UK0ERR	UKDIGEST1 in User Key segment error
6	UK2ERR	UKDIGEST2 in User Key segment error (UPK1)
7	UK3ERR	UKDIGEST3 in User Key segment error (UEK1)
8	UK4ERR	UKDIGEST4 in User Key segment error (DPK)
9	UK5ERR	UKDIGEST5 in User Key segment error (UPK2)
10	UK6ERR	UKDIGEST6 in User Key segment error (UEK2)
11	UPERR	User Permanent lock (UPERM) segment digest error
12	SYSERR	Factory Key Segments and Factory Key Segments digest error

5.1.7 Exporting Digests

The following sections describe how to export the stored digests via a design system service or the JTAG or SPI-slave interface.

5.1.7.1 Exporting Digests (Externally)

The stored digests can be exported via the JTAG or SPI-slave interface. A future programming software will have this feature.

5.1.7.2 On-Demand Read Digests

This read digest service returns the stored digests for the device.

Table 9 • **Read Digests Service Request**

15:7	6:0
MBOXADDR[10:2]	04H

Table 10 • **Read Digests Service Mailbox Format**

Offset	Length (bytes)	Parameter	Direction	Description
0	416	DIGESTS	Output	Digest Array

Table 11 • **Read Digests Service Status**

STATUS	Description	Note
0	Success	

The following table shows the digest output.

Table 12 • **Digest Output**

Offset (byte)	Size (bytes)	Value	Note
0	32	CFD	Fabric digest
32	32	CCDIGEST	Fabric Configuration segment digest
64	32	SNVMDIGEST	SNVM Digest digest
96	32	ULDIGEST	User lock segment
128	32	UKDIGEST0	UKDIGEST0 in User Key segment (includes SRAM PUF activation code and device Integrity bit)
160	32	UKDIGEST1	UKDIGEST1 in User Key segment
192	32	UKDIGEST2	UKDIGEST2 in User Key segment (UPK1)
224	32	UKDIGEST3	UKDIGEST3 in User Key segment (UEK1)
256	32	UKDIGEST4	UKDIGEST4 in User Key segment (DPK)
288	32	UKDIGEST5	UKDIGEST5 in User Key segment (UPK2)
320	32	UKDIGEST6	UKDIGEST6 in User Key segment (UEK2)
352	32	UPDIGEST	User Permanent lock (UPERM) segment
384	32	FDIGEST	Factory Key Segments and Factory Key Segments
Total	416 bytes		

The following table lists the various device Integrity test functions via JTAG/SPI and system services in PolarFire devices.

Table 13 • **PolarFire FPGA Integrity Test Functions Commands and Services**

	JTAG/SPI Command	System Service
Bitstream, IAP, and UIC Authentication Services (external SPI Flash)		X[1]
Export C-of-C tags (during bitstream pgm'g)	X[2]	
Export Digests Stored During Programming (on demand)	X[3]	X[3]
Compute/Export Fresh Digests (on demand)	X[4]	X[10]
Compute/Report Fresh Status Flags (on-demand)	X[4]	X[5]
Compute/Report Fresh Tamper Flag (after Power-on-Reset)		[6]
Export Zeroization Proof (after zeroization)	X[7]	
Device Integrity Flag (for new devices)	X[8]	
sNVM Authentication (when page is read)		X[9]

Notes:

1 The Bitstream and IAP Authentication system services check the validator and other header data (version) of a bitstream stored in the external SPI Flash. The UIC Bitstream Authentication system service checks the validator of a UIC script stored in the external SPI Flash.

2 As part of bitstream programming (FRAME_DATA JTAG/SPI commands)—Fresh C-of-Cs (MACs) of the newly programmed data are exported for each included bitstream segment (BITS, KEYS, FPGA, ENVM, EOB) as they are loaded. The user can then also run READ_DIGESTS (see 3) for un-keyed verification.

3 READ_DIGESTS JTAG/SPI commands and READ DIGEST System Service—On demand, exports digests that were previously computed and stored during programming for the Fabric, sNVM, various user locks, keys & passcodes, and factory segments (13 total digests reported).

4 CHECK_DIGESTS JTAG/SPI commands—Fresh flags for the sNVM, various user locks, keys & passcodes, and factory segments, and optionally for the Fabric (12 or 13 total flags reported). Immediately after running the CHECK_DIGESTS command, all of the fresh digests can be exported.

5 DIGEST CHECK system service: Computes fresh flags on demand for the Fabric, sNVM, various user locks, keys & passcodes, and factory segments (up to 13 total flags reported, depending on request); also sets DIGEST tamper flag if any errors

6 POWER-ON-RESET DIGEST: At boot of user design, a tamper flag upon failure. Can use DIGEST CHECK service to see which digest failed (see 5).

7 READ_ZEROIZATION RESULT JTAG/SPI commands provide digitally-signed expected digest response to a challenge (i.e., nonce) on a zeroized device.

8 READ_DEVICE_INTGRITY JTAG/SPI commands provide digitally-signed expected bit pattern response to a challenge (i.e., nonce) on a new (never user-programmed), un-tampered device.

9 If an sNVM page was written with the authentication option enabled, then the authentication tag is checked when the page is read

10 The correct digests can be determined using the READ DIGEST system service (see 3), and freshly validated using the DIGEST CHECK system service (see 5).

5.2 Information Services

Some of the device data are considered public and may be exported internally using system services from a fabric master. The information services return information about the device or current user design. The requested information is copied to a location in the mailbox RAM, the address of which is included in the service descriptor. The size of the data returned is service dependent. Overflows result in the return data wrapping around to the start of the mailbox. In addition, some device information can be exported from PolarFire devices using an external interface, such as the JTAG or SPI-slave programming interfaces.

The following table lists the public data that can be exported using the system service and external JTAG or SPI-slave programming interfaces.

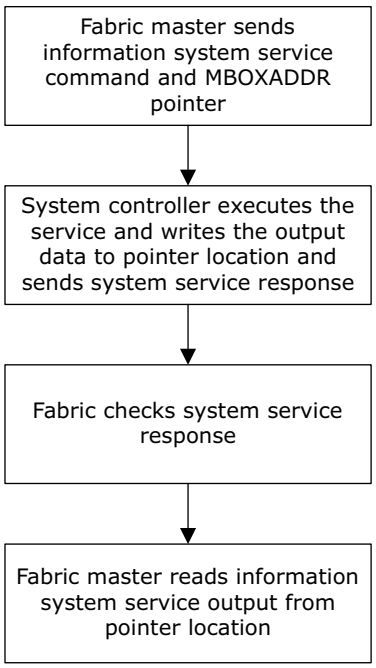
Table 14 • **Public Information Accessible**

Information	Internal (System Service)	External (JTAG/SPI)	Comment
Serial Number	x	x	Unique-per-device, 128 bits
IDCODE		x	Std. IEEE 1149.1 JTAG IDCODE value, 32 bits
USERCODE	x	x	Std. IEEE 1149.1 JTAG USERCODE value, 32 bits
Design Info		x	256 bits, set by user (Normally the top level component name)
Device Certificate	x	x	Returns 1024 bytes containing the freshly verified X-509-compliant certificate
Read digest	x	x	416 bytes
Query security	x	x[1]	9Bytes
Read Debug Info		x	64 bytes

Note: 1.Separate JTAG/SPI-Slave commands for the various security checks.

The device information system service is run using a fabric master. The following figure shows the device and design Information system service flow. The user must use the `CoreSystemService_PF` IP core to run these services. For more information about the `CoreSystemService_PF` IP core, see Design System Services, page 53.

Figure 16 • **Information System Service Flows**



5.2.1 Serial Number Service

Each PolarFire FPGA device has a unique, publicly readable, 128-bit device serial number (DSN). The DSN can be used in cryptographic protocols to uniquely identify the device. The DSN comprises two 64-bit fields: FSN and SNM.

- **FSN**—The first (most significant) field is the factory serial number (FSN). FSN is a pseudo-random per-device unique value assigned during Microsemi's manufacturing test, and persists for the lifetime of the device.
- **SNM**—The second component is the serial number modifier (SNM). SNM is initialized during factory test and is destroyed during the "recoverable" zeroization action. If the device is subsequently recovered, a new SNM is assigned such that each SNM generated for a given FSN, is unique.

The serial number design system service fetches the 128-bit Device Serial Number (DSN).

Table 15 • **Serial Number Service Request**

15:7	6:0
MBOXADDR[10:2]	00H

Table 16 • **Serial Number Service Mailbox Format**

Offset	Length (bytes)	Parameter	Direction	Description
0	16	DSN	Output	Device Serial Number

5.2.1.1 USERCODE Service

The user code design system service fetches the 32-bit USERCODE.

Table 17 • **USERCODE Service Request**

15:7	6:0
MBOXADDR[10:2]	01H

Table 18 • **USERCODE Service Mailbox Format**

Offset	Length (bytes)	Parameter	Direction	Description
0	4	USERCODE	Output	Device USERCODE

Table 19 • **USERCODE Service Status**

STATUS	Description	Note
0	Success	

5.2.1.2 Design Info Service

The design info design system service returns the design information.

Table 20 • **Design Info Service Request**

15:7	6:0
MBOXADDR[10:2]	02H

Table 21 • **Design Info Service Mailbox Format**

Offset	Length (bytes)	Parameter	Direction	Description
0	32	DESIGNID	Output	256-bit user-defined design ID, set in Libero SoC software
32	2	DESIGNVER	Output	16-bit design version, set in Libero SoC software
34	2	BACKLEVEL	Output	16-bit design back-level, set in SPM

Table 22 • **Design Info Service Status**

STATUS	Description	Note
0	Success	

5.2.2 Query Security

The query security design system service reads non-volatile states of user security locks.

Table 23 • **Query Security Service Request**

15:7	6:0
MBOXADDR[10:2]	05H

Table 24 • **Query Security Service Mailbox Format**

Offset	Length (bytes)	Parameter	Direction	Description
0	9	LOCKS	Output	Lock Array

Table 25 • **Query Security Service response**

STATUS	Description	Note
0	Success	

The following table shows the Lock output. This is identical to the data returned by the QUERY_SECURITY instruction from JTAG/SPI-SLAVE.

Table 26 • **LOCK Output**

Byte	Bit	Lock	Description
0			
	0	UL_DEBUG	Debug instructions disabled
	1	UL_SNVM_DEBUG	SNVM debug disabled
	2	UL_LIVEPROBE	Live probes disabled
	3	UJTAG_DISABLE	User JTAG interface disabled
	4	JTAG_BS_DISABLE	JTAG boundary scan disabled
	5	UL_TVS_MONITOR	External access to System TVS monitor disabled
	6	JTAG_MONITOR	JTAG fabric monitor enabled
	7	JTAG_DISABLE	JTAG TAP disabled
1	0	UL_PLAINTEXT	Plain text passcode unlock disabled
	1	UL_FAB_PROTECT	Fabric erase/write disabled
	2	UL_EXT_DIGEST	External digest check disable
	3	UL_VERSION	Replay protection enabled
	4	UL_FACT_UNLOCK	Factory test disabled
	5	UL_IAP	IAP disabled
	6	UL_EXT_ZEROIZE	External zeroization disabled
	7	SPI_DISABLE	SPI port disabled
2-8	0	Other Locks	Factory use only

5.2.3 Read Debug Info

This system design system service reads out useful debug information for programming, user initialization and IAP.

Table 27 • **Query Security Service Request**

15:7	6:0
MBOXADDR[10:2]	06H

Table 28 • **Query Security Service Response**

STATUS	Description	Note
0	Success	

Table 29 • **READ_DEBUG_INFO Shared Buffer Usage**

Size (Bytes)	Byte Offset	Parameter	Description
32	0	Reserved	
4	32	TOOL_INFO	Reflects the TOOL_INFO passed in during ISC_ENABLE prior to programming. IAP sets this to 0.
1	36	TOOL_TYPE	Tool type used to program device 1=JTAG, 2=IAP, 3=SPI_SLAVE
4	37	Reserved	
7	41-44	Reserved	
1	48	UIC_STATUS	User device initialization command (UIC) status.
1	49	UIC_SOURCE_TYPE	UIC source type when execution finished or halted.
2	50	Reserved	
4	52	UIC_START_ADDRESS	UIC source address when execution finished or halted.
4	56	UIC_INSTR_ADDRESS	UIC instruction count from start of UIC execution.
4	60	CYCLECOUNT	Programming cycle count.

6 Device-Level Anti-Tamper Features

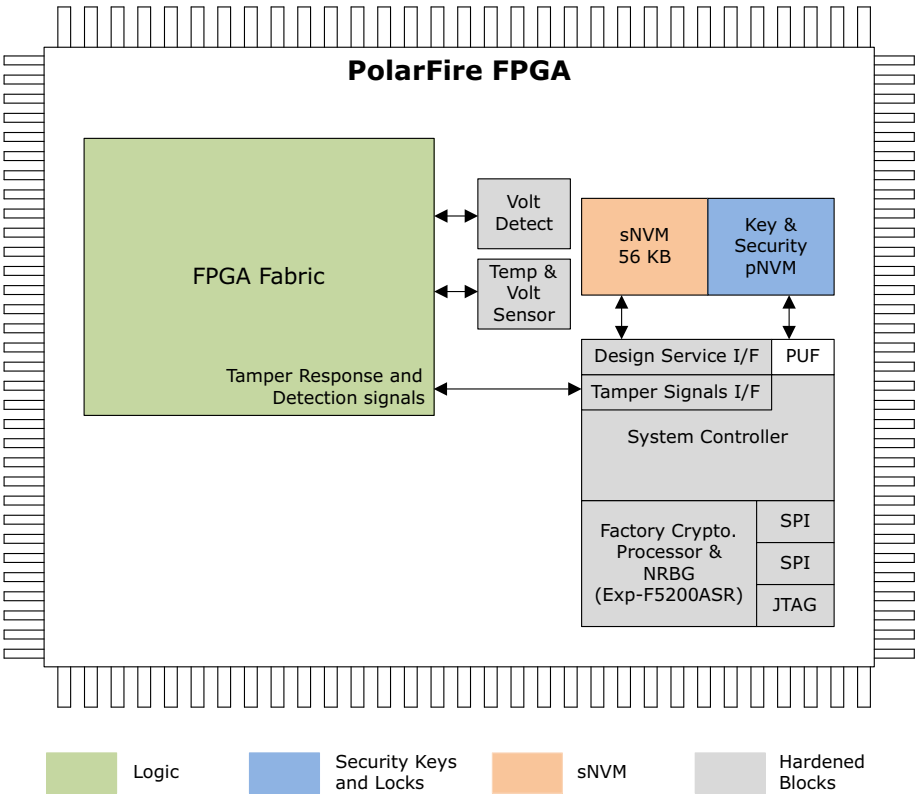
PolarFire FPGAs include a number of built-in tamper detection and response capabilities that can be used to enhance the security of the device. These countermeasures are intended to address various types of attacks that include non-invasive, semi-invasive, and invasive attacks. PolarFire devices can detect a number of conditions that may indicate an attempt to tamper.

When a tamper condition is detected, a notification is sent to the fabric via one of many dedicated control lines. The Tamper macro can be used to expose those flags to the user design. On receiving a tamper event, the fabric design may either choose to ignore the event or take defensive action. If the user ignores the event, the operation of the user design is not impacted. If the device is in Flash*Freeze state when a tamper event occurs, the event causes an immediate Flash*Freeze exit.

Note: Tamper monitors are only active during Flash*Freeze if they were enabled at the time Flash*Freeze was entered.

In addition to tamper detection and tamper response, all built-in uses of design security protocols in PolarFire devices have protection against differential power analysis (DPA) and related side-channel monitoring attacks. All cryptographic algorithms are implemented in PolarFire FPGA are DPA-resistant. Similar to SmartFusion2 and IGLOO2 FPGAs, PolarFire FPGAs also have upgraded integrity check mechanisms compared to earlier generation devices that can optionally be used to check the reliability and security of a device automatically upon power-up, during programming or on-demand.

Figure 17 • **Tamper Detection and Response Interface to the FPGA Fabric**



6.1 Tamper Detection and Tamper Response

PolarFire devices have built-in tamper detection output flags and tamper response input commands that are available to the FPGA fabric. The tamper detection flags allow the user to monitor tamper events and then trigger the built-in or custom tamper responses using the tamper response command input signals. The built-in tamper detection output flags and tamper response inputs are exposed to the FPGA fabric, as shown in Figure 18, page 45.

A Tamper macro is provided in the Libero SoC software Catalog to access tamper flags and response inputs from fabric. The following figure shows the Tamper macro and its configurable options.

Figure 18 • **Tamper Macro**

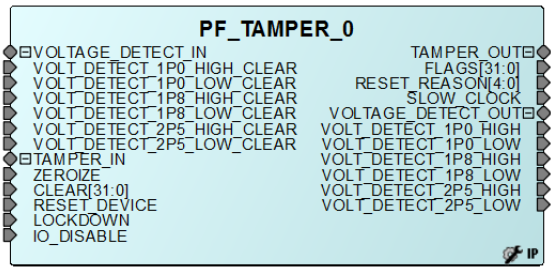


Figure 19 • **Tamper Configurator - Tamper**

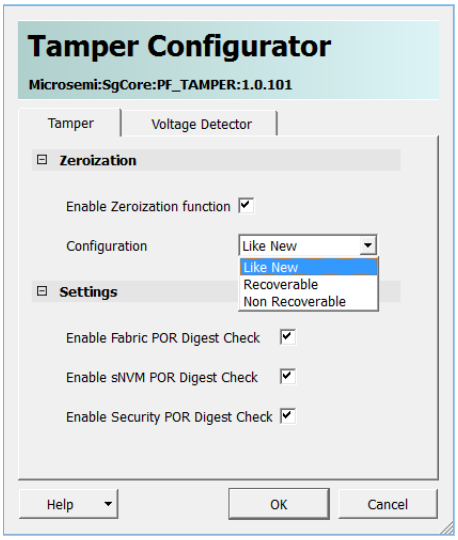
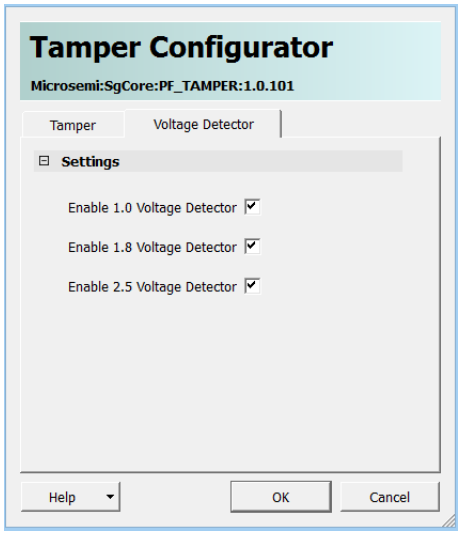


Figure 20 • **Tamper Configurator - Voltage Detector**



In the Libero SoC software tool, the user needs to instantiate the tamper macro in their design to expose the tamper detection output flags and tamper response inputs to/from the FPGA fabric.

6.1.1 Tamper Detection Flags

Tamper-detection flags inform the user about tampering activity. Each tamper event is signaled over a dedicated wire and the tamper signals are active low. On receiving the tamper detection flags, the user can choose to use the appropriate tamper response (see Tamper Response, page 48) or ignore/clear the flags. Tamper events can only be cleared by asserting the associated fabric clear signal or a system reset.

In the event of a fatal tamper event, the user design is powered down and an automatic reboot is executed. The shutdown sequence guarantees a minimum of 10µs after the tamper alarm fires before the shutdown begins, allowing the user design to perform internal clean up tasks. The design system services may not be used during this time.

The following table lists and describes the tamper detection flags.

Table 30 • **Tamper Detection Flags**

Flags[0:31]	Flag	Description
0	JTAG_ACTIVE	This signal is triggered whenever the JTAG TAP controller enters the Run-Test-Idle state. This event can be cleared by sending a JTAG_ACTIVE_CLR signal from the fabric.
1	MESH_ERROR	This signal is generated whenever the active mesh observes a mismatch between the actual metal mesh output and the expected output. This allows protection against invasive at-tacks, such as cutting and probing of traces using focused ion beam (FIB) technology with an active metal mesh on one of the higher metal layers. This event can be cleared by sending a MESH_ERROR_CLR signal from the fabric.
2	CLOCK_DETECTOR_GLITCH	This signal occurs each time the clock glitch detector detects a pulse width violation. This event can be cleared by sending a CLOCK_DETECTOR_GLITCH_CLR signal from the fabric.

Table 30 • **Tamper Detection Flags *(continued)***

Flags[0:31]	Flag	Description
3	CLOCK_DETECTOR_FREQUENCY	This signal is asserted each time the clock frequency detector observes a frequency mismatch between the 160 MHz and 2 MHz RC oscillators. This flag is set if the discrepancy over a number of clock cycles is too high. The PolarFire device includes an additional tamper detection mechanism compared to SmartFusion2 and IGLOO2. This is set if the 160 MHz clock source is totally disabled between clock edges on the 2 MHz clock. In this case, the system asynchronously asserts the fabric tamper flags. This event can be cleared by sending a CLOCK_DETECTOR_FREQUENCY_CLR signal from the fabric.
4	LOW_1P05	This flag indicates that the 1.05 V supply (V_{DD}) is below the low threshold of the System Controller 1.05V detector. The tamper event is continuously generated until the supply returns to a level above the low threshold. This event can be cleared by sending a LOW_1P05_CLR signal from the fabric.
5	HIGH_1P8	This flag indicates that the 1.8V supply (VPP) is above the high threshold of the System Controller 1.8V detector. The tamper event is continuously generated until the supply returns to a level below the high threshold. This event can be cleared by sending a HIGH_1P8_CLR signal from the fabric.
6	HIGH_2P5	This flag indicates that the 2.5V supply (VPPA) is above the high threshold of the System Controller 2.5V detector. The tamper event is continuously generated until the supply returns to a level below the high threshold. This event can be cleared by sending a HIGH_2P5_CLR signal from the fabric.
7	GLITCH_1P05	This signal is asserted whenever the System Controller 1.05 V glitch detector observes a voltage glitch on 1.05 V supply (V_{DD}). This event can be cleared by sending a GLITCH_1P05_CLR signal from the fabric.
8	SECDED	Asserted when a 2-bit error occurs in the System Controller's main memory. This is a fatal condition which results in a system reboot. This event can be cleared by sending a SECDEC_CLR signal from the fabric.
9	SCB_BUS_ERROR	This event occurs when an error has been detected on an System Controller bus. This event can be cleared by sending an SCB_BUS_ERROR_CLR signal from the fabric.
10	WATCHDOG	Asserted when the watchdog reset is about to fire. This is a fatal condition that results in a system reboot. This event can be cleared by sending a WATCHDOG_CLR signal from the fabric.
11	LOCK_ERROR	This signal is asserted when a single- or double-bit error is detected in the continuously-monitored user security locks. This event can be cleared by sending a LOCK_ERR_CLR signal from the fabric.
12	Reserved	Not used
13	DIGEST	A requested digest check failed
14	INST_BUFFER_ACCESS	Read/Write of software IO buffer
15	INST_DEBUG	Debug instruction executed
16	INST_CHECK_DIGESTS	An external digest check has been requested
17	INST_EC_SETUP	Elliptic Curve slave instructions have been used

Table 30 • **Tamper Detection Flags *(continued)***

Flags[0:31]	Flag	Description
18	INST_FACTORY_PRIVATE	Private factory instruction executed
19	INST_KEY_VALIDATION	Key validation protocol requested
20	INST_MISC	Uncategorized slave instruction executed
21	INST_PASSCODE_MATCH	Attempt to match a passcode
22	INST_PASSCODE_SETUP	OTP passcode protocol initialization
23	INST_PROGRAMMING	An external programming instruction has been used
24	INST_PUBLIC_INFO	Request for device public information
25	INST_ZEROIZATION_RECOVERY	Zeroization recovery has been attempted
26	INST_PASSCODE_FAIL	Passcode match failed
27	INST_KEY_VALIDATION_FAIL	Key validation failed
28	INST_UNUSED	Unused instruction opcode executed
29	BITSTREAM_AUTHENTICATION_FAIL	Bitstream authentication failed
30	IAP_AUTO_UPDATE	Device has executed an auto update
31	IAP_AUTO_RECOVERY	Device has executed auto recovery

6.1.2 Tamper Response

PolarFire devices have four built-in tamper responses that can be triggered from the FPGA Fabric. The Tamper macro shown in Figure 18, page 45, exposes these tamper response inputs to the FPGA fabric. The tamper responses are normally initiated after receiving a tamper event described in the previous section, but they may also be initiated on-demand by the user.

6.1.2.1 IO Disable

The IO disable response allows the user design to immediately disable FPGA I/Os. Each I/O can be statically configured to be either disabled, preventing any further communication, or remain active and operate as normal. A disabled I/O has a tristated output. The IO Disable response persists until the associated fabric control signal is negated, the disabled I/Os then return to user control.

Note: The IO disable feature does not disable SerDes I/Os. Hence, user must put the SerDes blocks in Reset before asserting the IO_Disable signal to disable SerDes I/Os.

6.1.2.2 Security Lockdown

The security lockdown response activates all the user lock bits to behave as if they are locked. This is irrespective of the underlying user defined lock state. Any passcodes that were unlocked become immediately locked.

As long as this signal is asserted, the device does not respond to any programming command, or perform any cryptographic service. The Security Lockdown response persists until the associated fabric control signal is negated. When that happens, lock states return to their pre-lockdown state, but passcodes are locked.

6.1.2.3 Reset

This reset response signal sends a reset request to the system controller. The system controller immediately powers down the device and re-executes its normal power-up sequence. This is equivalent to asserting the device reset pin.

6.1.2.4 Zeroization

PolarFire devices have a built-in tamper response capability that can zeroize (clear and verify) any or all configuration storage elements as per the user setting. Most volatile storage is also cleared and verified. The user can monitor the built-in tamper detection flags or other system events and then decide to trigger one of the three types of built-in zeroization requests and zeroize the device. The user needs to enable zeroization and set the chosen zeroization option (using the Tamper macro configurator) in the static design configuration, and then at run-time, send a zeroization request from FPGA fabric to the system controller. Zeroization may be also triggered via a JTAG or SPI slave instruction. Zeroization is immune to the security lockdown response, which essentially means that asserting a security lockdown does not prevent zeroization from initiating or completing.

Like the SmartFusion2/IGLOO2 FPGA families, PolarFire FPGAs have the following three zeroization modes:

- **Like New**—All user data is destroyed. The device is effectively returned to its original factory state, allowing it to be programmed like a new device. User permanent locks are not affected.
- **Recoverable**—All user data is destroyed. Additionally, the device certificate and factory keys are also destroyed. The device is placed in a state which prevents any further use until a recovery procedure is executed. For the recovery procedure to be executed, the device must prove to a Microsemi factory HSM that all data has indeed been zeroized. A programming file is then generated which assigns the device new keys and certificate, returning the device to the Like New state. The factory serial number (FSN) half of the device serial number (DSN) stays the same, but the serial number modifier (SNM) half of the device serial number is changed so that a recovered device can be told apart from a new one.
- **Unrecoverable**—All user data is destroyed. Factory data is also destroyed. Upon completion of zeroization in the Unrecoverable mode, the only allowed access to the device is retrieval of the zeroization certificate. The device may not otherwise be used again.

The following table lists the status of the various FPGA components during the three zeroization modes.

Table 31 • **Status of Various FPGA Components During the Three Zeroization Modes**

Zeroization Modes	Description	FPGA	Factory and user re-configurable lock bit segment	Factory Lock segment	User Lock segment	Factory Parameter Segment in pNVM	User Key in pNVM	Factory Key in pNVM	sNVM
Like New	**Zeroize user data and keys**	✓		✓	✓		✓		✓
Recoverable	**Zeroize user data and keys and Factory Keys**	✓		✓	✓		✓	✓	✓
Unrecoverable	**Zeroize everything**	✓		✓	✓	✓	✓	✓	✓

6.1.2.4.1 Zeroization of Volatile Memories

All volatile user memories (SRAM or registers) are zeroized. Any System Controller memories containing secrets are also zeroized.

- **Fabric Memories**—Fabric volatile memories are zeroized using an improved method in PolarFire FPGA. Thus all fabric LSRAMs and µRAMs can be zeroized faster.
- **Fabric Registers**—Fabric registers are zeroized using an improved method in PolarFire FPGA.

6.1.2.4.2 Zeroization of NVM

All user NVM (FPGA fabric and uPROM) is zeroized with an enhanced method, to a verifiable state in a short period of time. The content is destroyed and the NVM cells "scrubbed" to ensure there are no usable remnants left in the memory that could be used to identify the prior content.

6.1.2.4.3 Zeroization of pNVM

Like the NVM, the pNVM is zeroized by applying cycles of erase and program pulses.

6.1.2.4.4 Zeroization of UserCrypto and Factory Crypto Core

Both, Athena user and factory cores support a purge function to clear all internal memory.

6.1.2.4.5 Zeroization of Intrinsic-ID PUF

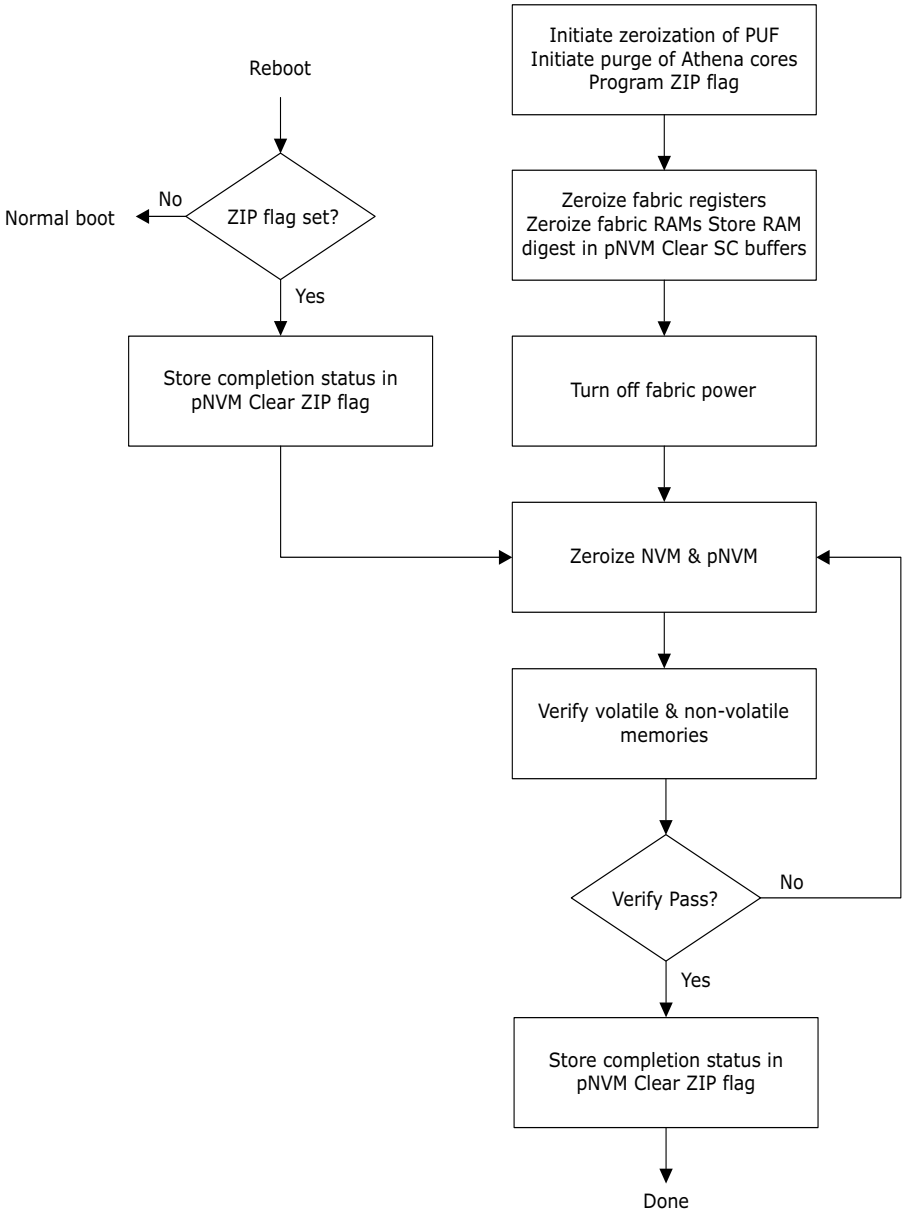
The PUF has its own zeroization function, which clears all internal secrets.

6.1.2.5 Zeroization Flow

The zeroization procedure in PolarFire FPGA includes several erase and programming operations to reduce any data remnants in the flash array to undetectable levels (a process known as "scrubbing"). This section briefly describes the zeroization procedure. Additional zeroization documentation is available under NDA. When zeroization is initiated, it always runs to completion, even if interrupted by a device reset or loss of power.

After the activation of zeroization command or request from the fabric or JTAG or SPI Slave, the system controller programs a Zeroization-In-Progress (ZIP) flag that act as status flags during the zeroization process. The ZIP flag is checked during device boot and, if set, the zeroization procedure is restarted or resumed. The zeroization flow is shown in Figure 21, page 51. On completing the zeroization, the device must generate a certificate proving that all requested data has been destroyed. The ZIP flag is cleared after generating the certificate. Once zeroization is complete, the zeroization result (proof of zeroization) can be read from the device via the JTAG or SPI slave interfaces in response to a challenge from the user, proving the response was fresh and not just replayed from another device or time.

Figure 21 • **Zeroization Flow**



6.1.2.6 Zeroization Verification

Zeroization is not complete until all verification operations are successful. If a verification step fails, the zeroization sequence is repeated until verification is successful. Thus, completing zeroization is itself an indication that verification was successful. Once complete, the Zeroization-Done flag, ZDONE, is set and programmed into pNVM along with the zeroization mode, ZMODE. A future version of this user guide will have detailed information about how to verify Zeroization-Done flag, ZDONE.

6.2 JTAG Security Monitor

In SmartFusion2 and IGLOO2 devices, the user JTAG (UJTAG) interface provided a mechanism for extending TAP controller functionality, which was also used in security applications for limited monitoring of JTAG activity. This interface is enhanced in PolarFire devices with additional features. The UJTAG macro allows a user-defined security monitor implemented in the FPGA fabric to observe all JTAG signal activity. It also enables control of the System Controller TDI input and its TRSTB input.

Note: When the JTAG Security Monitor interface is enabled, the maximum TCK frequency for the device may be limited by delays through the user logic. TCK frequency is not monitored by the device as no security sensitive logic operates in this clock domain. When the FPGA fabric is powered down, the Security Monitor interface is disabled.

A future version of this user guide will have more information about this topic.

6.3 Voltage Detectors

Each PolarFire FPGA supply voltage is equipped with a voltage detector, which triggers an alarm when the supply voltage falls below a minimum level and/or raises above a maximum level. The alarm condition is passed directly to the fabric. The alarm condition is cleared by asserting the event clear signal to the fabric. The threshold levels on the detectors are set in the factory. Use Tamper macro to access the voltage detector inputs and outputs. See Figure 20, page 46.

6.4 Temperature and Voltage Sensor

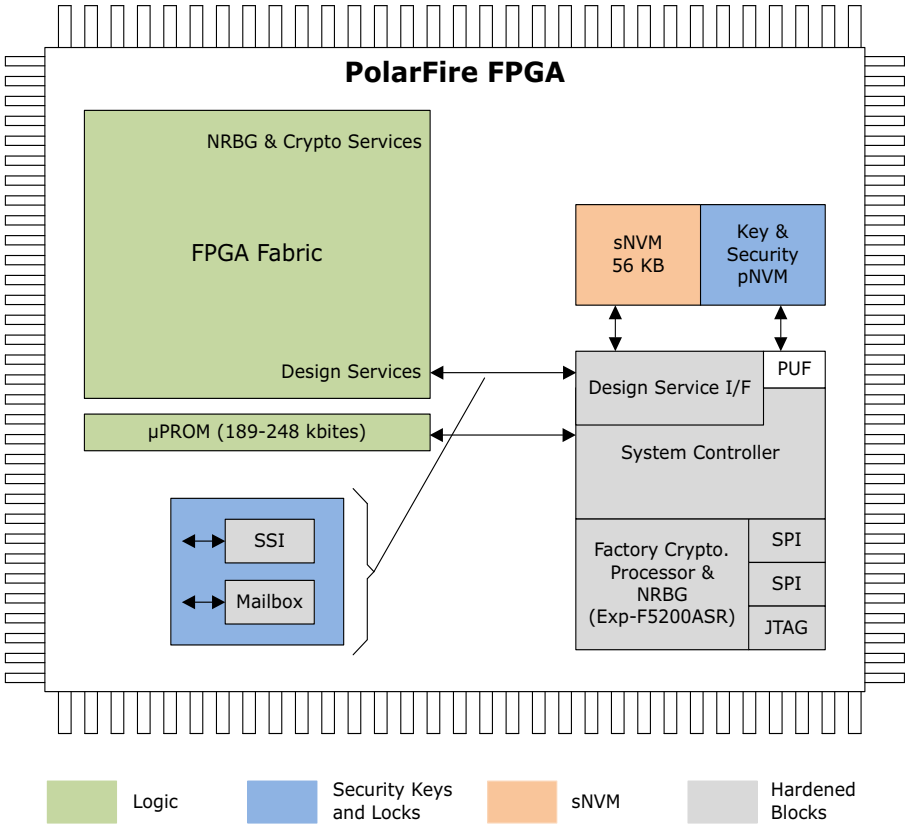
Each PolarFire device is equipped with an ADC based Temperature and Voltage Sensor. It provides die temperature and the value of the supply voltages available on the chip in digital form to the FPGA fabric. The output of the sensor is a calibrated 10-bit value. The temperature and voltage information is translated into standardized temperature and voltage values and are used by the programming circuits. The output of the temperature sensor is also used by a hardware logic widow-comparator residing within the system controller that raises an alarm if the temperature is not between the maximum and minimum threshold levels specified by the customer.

A future version of this user guide will have more information about temperature and voltage sensors.

7 Design System Services

System services are system controller actions initiated by asynchronous events from the user design via the system controller's design service interface. The design service interface includes the system service interface (SSI) and a mailbox interface. The SSI operates in conjunction with the mailbox interface, which is a synchronous interface operating on a user-supplied clock. The mailbox interface is used for passing data related to the system service between the fabric and the system controller (in both directions). The mailbox interface consists of a set of address, data and control signals for a 2kB dual-port RAM within the System Controller. The following figure shows the design service interface (comprising the SSI and mailbox interface) between the FPGA fabric and system controller.

Figure 22 • **Design Service Interface Between Fabric and System Controller**



7.1 SSI Interface Signals

The following table lists the SSI interface signals.

Table 32 • **SSI Interface Signals**

Signal Name	Direction	Description
FAB_SS_REQ	Input	Service Request. The fabric sets this high and must keep it high until SS_ACK is asserted. The controller ignores when SS_BUSY is asserted.
FAB_SS_ACK	Output	Indicates that the controller has received the request and latched the command data. At this time, the fabric can de-assert SS_REQ.
		Is de-asserted after SS_REQ is de-asserted.
FAB_SS_BUSY	Output	Indicates that the controller is busy processing the request, is set the same time as SS_ACK. This remains set until the controller has completed the service request and requested data is available in the mail box memory.
FAB_SS_CMD[15:0]	Input	Service Request command/data. Must be valid before SS_REQ is asserted and held until SS_ACK is asserted
FAB_SS_STATUS[15:0]	Input	Service Request status. The controller sets this before de-asserting SS_BUSY.
FAB_SS_ABORT	Input	Service request abort. Instructs the controller to abort the current service request. Once set, it must be held set until SS_BUSY is de-asserted and must be de-asserted before the next SS_REQ assertion. It may be asserted before the SS_SCK is asserted.

7.2 Mailbox Interface Signals

The following table lists the interface signals from the fabric to the mailbox.

Table 33 • **Interface Signals From the Fabric to the Mailbox**

Signal Name	Direction	Description
FAB_MBOX_CLK	Input	Clock to Mailbox RAM
FAB_MBOX_WRITE	Input	Write strobe to all 32-bits
FAB_MBOX_READ	Input	Read strobe
FAB_MBOX_ADDR[8:0]	Input	Address (Word based)
FAB_MBOX_WDATA[31:0]	Input	Write data
FAB_MBOX_RDATA[31:0]	Output	Read data, is generated four clock cycles after the read strobe.

Table 33 • **Interface Signals From the Fabric to the Mailbox *(continued)***

Signal Name	Direction	Description
FAB_MBOX_ECC[1:0]	Output	Provides ECC status when the mail box is read. 2'b00: No ECC errors detected, data is correct 2'b01: Indicates exactly one bit error and has been corrected 2'b10: Indicates exactly two bit error, no correction performed 2'b11: Indicates more than two bits are erroneous, no correction performed

System services are invoked by writing a 16-bit system service descriptor to the SSI, which triggers a service request to the system controller. The system controller inspects the descriptor to determine the service to be executed. The lower seven bits of the descriptor specify the service to be performed, and the upper nine bits are used to provide additional information, typically the address of a location in the 2KB mailbox RAM containing additional parameters that must be pre-written to the mailbox before requesting the service. The following table shows the SSI system service descriptors.

Table 34 • **SSI System Service Descriptor**

SSI System Service Descriptor	Descriptor Name	Description
15:7	MBOXADDR[10:2]	Mail box address where additional information is provided.
6:0	SERVICEID	Service ID.

The following figures show the system services request signal timing.

Figure 23 • **System Services Request Signal Timing**

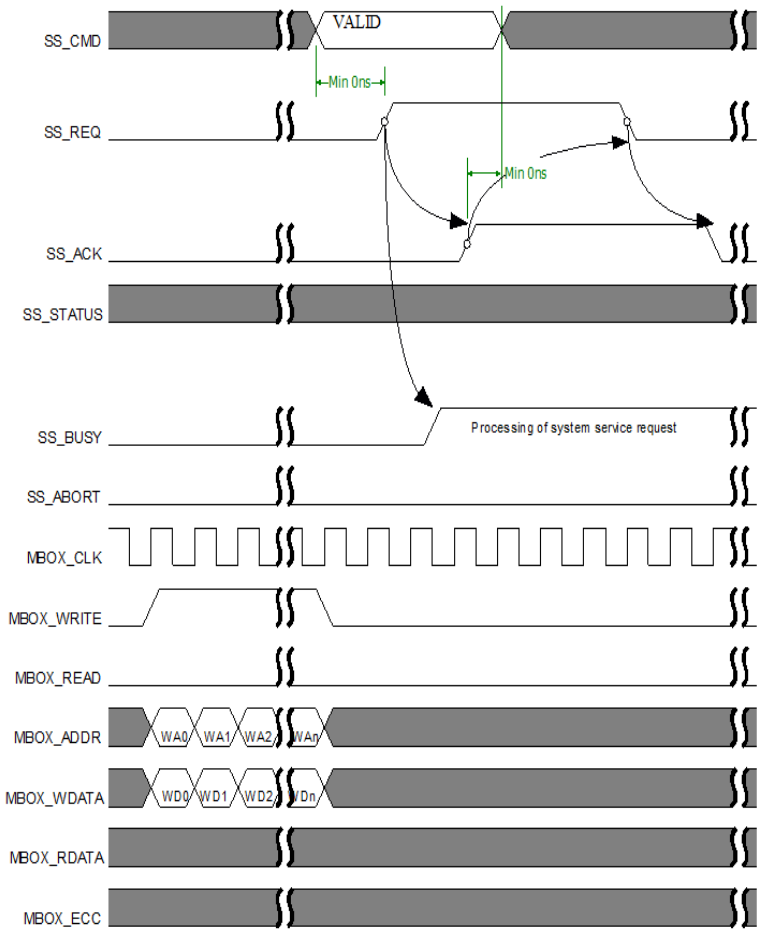


Figure 24 • **System Services Status Signal Timing**

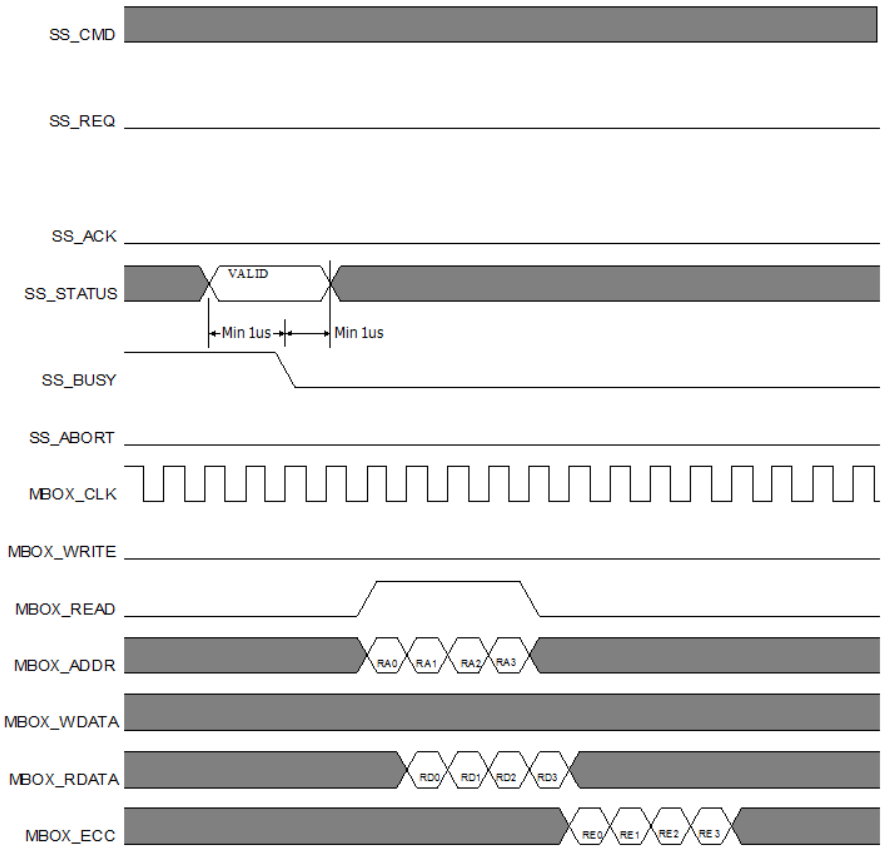


Figure 25 • **System Service Interface Abort Timing**

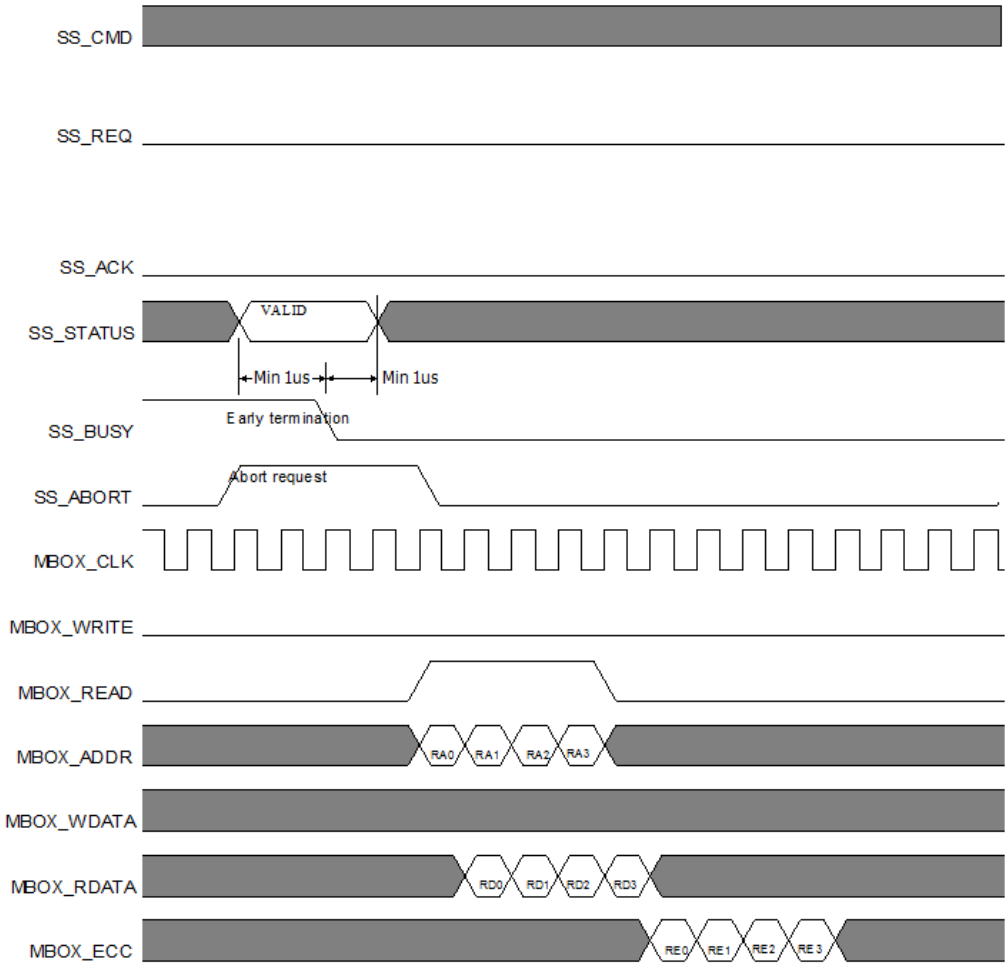
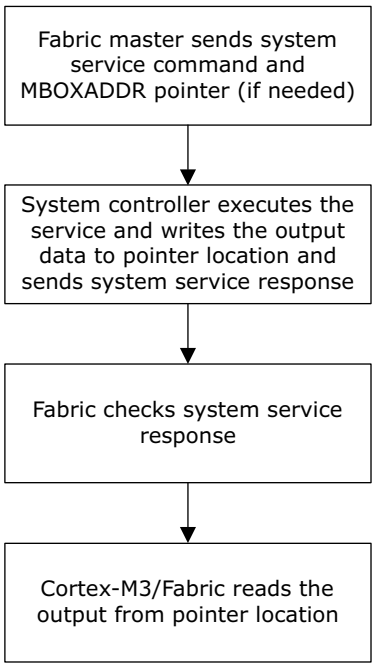


Figure 26 • **Device Certificate System Service Flows**



7.3 Secure NVM Write Service

The Secure NVM write service provides write access to pages in the sNVM region of the pNVM. Data may be stored as encrypted and authenticated cipher text, authenticated plain text, and non-authenticated (ordinary) plain text.

For authenticated plain text and authenticated cipher text, a 512-bit an sNVM master key (SMK) is the primary key used, with 256-bits allocated for authentication and 256 bits for encryption. SMK is protected as an encrypted/authenticated key code by the PolarFire FPGA PUF, and stored in the pNVM in protected memory. For crypto-enabled options, the system controller uses AES-256 in synthetic initialization vector (SIV) mode, which supports authenticated-encryption with additional authentication-only data, providing a 256-bit security strength. In SIV mode, the IV used for the encryption function is computed from the data, preventing IV misuse, and doubles as the authentication tag. The computed 128-bit IV/tag is stored in the same page as the user data, reducing the available space for user data by 16 bytes compared to the non-authenticated plain text-only option.

Besides the user-supplied plain text data, PolarFire FPGAs also submit additional metadata for authentication that effectively provide a "tweak" to the encryption and authentication functions. Some of the data included are the page address and the page write-counter. This means that the cipher text and the authentication tag are different even if the same data is written to two different sNVM page addresses, or even if the same data is written to the same page again (since the page-write counter advances).

In addition, the sNVM system services allow the user to protect each sNVM page independently. The user must include a user sNVM key (USK) when writing (or reading) each sNVM page. The USK is used as another element in the "tweak." Without the same 96-bit (12 byte) USK as was used during the write command, the read command fails authentication (and could not possibly decrypt correctly, either). The user can choose to set this key differently for each page, or for groups of pages, or the same for all pages—either as a secret key for added security, or to a frivolous value such as all zeroes if this feature is not needed.

Table 35 • **Secure NVM Write Request**

15:7	6:0	Description
MBOXADDR[10:2]	10H	Non-authenticated plain text
MBOXADDR[10:2]	11H	Authenticated plain text
MBOXADDR[10:2]	12H	Authenticated cipher text

Table 36 • **Secure NVM Write Service Mailbox Format (10H)**

Offset	Length (bytes)	Parameter	Direction	Description
0	1	SNVMADDR	Input	SNVM module
1	3	RESERVED		For alignment
4	252	PT	Input	Data to write to SNVM

Table 37 • **Secure NVM Write Service Mailbox Format (11H,12H)**

Offset	Length (bytes)	Parameter	Direction	Description
0	1	SNVMADDR	Input	SNVM module
1	3	RESERVED		For alignment
4	236	PT	Input	Data to write to SNVM
240	12	USK	Input	User Secret Key

SNVM modules marked as ROM cannot be overwritten by this service. The service cannot be used to create ROM modules (write-protected pages). ROM is declared when a bitstream is generated, and a page's ROM status can only be changed with a new bitstream, and not at run-time.

Table 38 • **SNVM Write Service Status**

STATUS	Description	Note
0	Success	
1	Invalid SNVMADDR	Illegal page address
2	Write failure	PNVM program/verify failed
3	System error	PUF or storage failure
4	Write Not Permitted	ROMFLAG is set

7.4 Secure NVM Read Service

The Secure NVM read service provides access to the data stored by the secure NVM write service or data programmed via a bitstream. If the data was programmed using authentication, then the USK key used at the time of programming must also be provided.

Table 39 • **Secure NVM Read Request**

15:7	6:0
MBOXADDR[10:2]	18H

Table 40 • **Secure NVM Read Service Mailbox Format (18H)**

Offset	Length (bytes)	Parameter	Direction	Description
0	1	SNVMADDR	Input	SNVM module
1	3	RESERVED		For alignment
4	12	USK	Input	User Secret Key (ignored if page is plain text)
16	4	ADMIN	Output	Page admin data
20	236/252	PT	Output	Data read from SNVM

Table 41 • **SNVM Read Service Status**

STATUS	Description	Note
0	Success	
1	Invalid SNVMADDR	Illegal page address
2	Authentication Failure	Storage corrupt or incorrect USK
3	System error	PUF or storage failure

Reading data that has been tagged for authentication or enciphered by a debugger tool requires the value of USK to be supplied.

A future version of this user guide will provide more information about debugging authenticated or authenticated-encrypted sNVM data.

7.5 PUF Emulation Service

The PUF emulation service provides a mechanism for authenticating a device, or for generating pseudo-random bit strings that can be used for many different purposes. The service accepts a 128-bit challenge and an 8-bit optype, and returns a 256-bit response unique to the given challenge, optype, and device.

Table 42 • **PUF Emulation Service Request**

15:7	6:0
MBOXADDR[10:2]	20H

The data for the service is specified by the MBOXADDR field which defines the location of a 14-word parameter block defined in the following table.

Table 43 • **PUF Emulation Service Mailbox Format**

Offset	Length (bytes)	Parameter	Direction	Description
0	1	OPTYPE	Input	Challenge OPTYPE
1	3	RESERVED	-	Reserved for memory alignment
4	16	CHALLENGE	Input	Challenge input
20	32	RESPONSE	Output	Response output

Table 44 • **PUF Emulation Service Status Format**

STATUS	Description
0	Success
1	Internal error

```
RESPONSE = KeyTree(PEK, OPTYPE, CHALLENGE)
```

Where PEK is the factory-defined PUF emulation key and KeyTree is a function that uses the 8-bit OPTYPE concatenated with the 128-bit CHALLENGE to navigate a binary key tree with the 256-bit secret PEK at its root. The leaf of the tree that is computed as a result of the 136 internal hashing operations (one for each level in the binary tree), is a 256 bit secret. The root key, PEK, the result, RESPONSE, and the intermediate results are protected against side-channel attacks due to the nature of the protocol, plus the fact that the SHA algorithm implemented in the system controller's TeraFire cryptographic core also has strong DPA countermeasures. The OPTYPE and CHALLENGE are not protected against side-channel leakage. The OPTYPE allows the user to conceptualize there are 256 different 128-bit key trees, each with 2^{128} possible output responses, which can be put to different uses without much danger of collision.

The function emulates a "strong PUF", which means that it takes a cryptographically large challenge space and computes a pseudo-random repeatable output response from it, but in this implementation, it does not use unclonable physical properties developed during the manufacturing of the device for the challenge-response calculation, instead using classical cryptographic algorithms; thus the "emulation" disclaimer. The root key PEK is, however, protected as an encrypted/authenticated PUF key code, so the unclonable physical properties of the PolarFire device do enter into the reconstruction of the PUF secret and decryption of the key code to unwrap PEK for use in this function.

There are many uses in cryptography for such a per-device unique, pseudo-random function.
One use is to identify a particular chip by first recording (possibly several) challenge-response pairs, then later seeing if the target chip provides the same response as expected for one of the recorded challenges-response pairs. Another application is deriving many keys from one.Nonce Service

The nonce service provides the user with the ability to strengthen the DRBG of the Athena F5200B TeraFire random bit generator by providing an alternate entropy source to use as additional seed data in its DRBG functions.

Table 45 • **Nonce Service Request**

15:7	6:0
MBOXADDR[10:2]	21H

Table 46 • **Nonce Service Mailbox Format**

Offset	Length (bytes)	Parameter	Direction	Description
0	32	NONCE	Output	Generated nonce

NONCE = KeyTree256(PUK, 0, PUFSEED)

Where PUFSEED is a 256-bit conditioned true random output of the PUF.

Table 47 • **Nonce Service Status**

STATUS	Description
0	Success completion (exit)
1	Error fetching PUK
2	Error generating seed

To generate maximum entropy and forward and backward resistance, the PUF is automatically power-cycled before generating the seed.

Note: Since the seed is used as the path parameter to the KeyTree256 function, its value will leak.

The generated nonce itself is still a secret since PUK is a 256 bit randomly-distributed secret and is resistant to leakage

7.6 Digital Signature Service

The digital signature design system service takes a user-supplied SHA-384 hash and signs it with the device's 384-bit private "factory" elliptic-curve-cryptography key, FEK, which is the private half of the key pair whose public key (CPK) is certified by Microsemi in the device's X.509-compliant supply chain assurance certificate (SCAC). The resulting P-384 ECDSA signature can either be formatted using ASN.1 DER or simply returned in a raw format compatible with the user TeraFire cryptoprocessor. Since ECDSA requires the use of a nonce, the service returns a different result each time, even if the hash input is the same.

The factory cryptoprocessor does not directly support generating a nonce with the required numerical range required for ECDSA. It is therefore possible that the generated nonce is rejected, in which case a new nonce is automatically generated until a good value is found. This makes the execution time of this service non-deterministic, however, the probability of an out-of-range nonce being initially generated is extremely low and the probability of a second bad nonce is infinitesimal.

Table 48 • **Digital Signature Service Request**

15:7	6:0	Description
MBOXADDR[10:2]	19H	Raw Format
MBOXADDR[10:2]	1AH	DER Format

Table 49 • **Digital Signature Service Mailbox Format**

Offset	Length (bytes)	Parameter	Direction	Description
0	48	HASH	Input	SHA-384 hash to be signed
48	96 (Raw)	SIGNATURE	Output	ECDSA signature (r, s)
	104 (DER)			

SIGNATURE = ECDSA(FEK, HASH).

Table 50 • **Digital Signature Service Status Format**

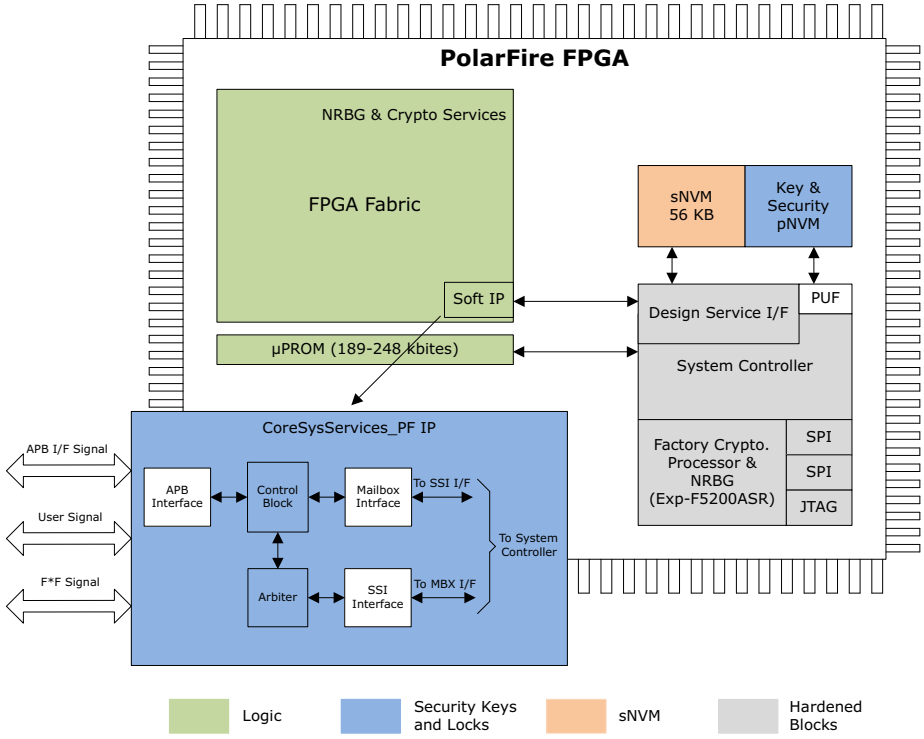
STATUS	Description	
0	Success	
1	FEK Failure	Error retrieving FEK
2	DRBG Error	Failed to generate nonce
3	ECDSA Error	ECDSA failed

If the Raw format is selected, the SIGNATURE field contains two unsigned little-endian 12-word (48 byte) values compatible with the Athena F5200B.

If the DER format is selected, the SIGNATURE field is returned in a minimal length DER encoding using a maximum of 104 bytes. If the encoded signature is less than 104 bytes, then the output is padded with zeroes. The extra bytes, if any, must be deleted by the user.Using CoreSysServices_PF Soft IP.

The CoreSysServices_PF soft IP provides an easy user interface to run the design system services. It provides an AMBA APB interface to the user on one side. On the other side, it is connected to SSI and mailbox interfaces. The SSI interface is used for command and status protocol. The mailbox interface is used to provide additional information, typically the address of a location for the specified service or data for the requested service or returns the response data related to the requested service.

Figure 27 • **CoreSysServices_PF Block Diagram**



For more information about CoreSysServices_PF, see *CoreSysServices_PF Handbook* (To be Published).

8 Internal Security Features

8.1 Single-Event Upset Robustness

In complementary metal-oxide-semiconductor (CMOS) technology, memories such as SRAM and flip-flops can have their state changed by radiation particles. These particles can come from high-energy particles hitting atmospheric molecules, causing them to emit sub-atomic particles such as neutrons that can create charged particles when they strike the silicon. Another source of alpha radiation is from device packaging materials such as epoxy. The net result is that radiation can cause the state of a memory cell to flip.

The effect, called single-event upset (SEU) is worse at some latitudes (due mainly to the shape of the Earth's magnetic field) and altitudes. In general, it gets worse from sea-level to approximately 60,000 feet, and then drops up to the radiation belts circling the Earth where the radiation is quite high. Even at sea-level, it is usually by far the largest single source of failures in SRAM FPGAs, and the effect can go up several orders of magnitude at altitudes that commercial airlines or military aircraft reach.

Configuration memory, which is supposed to hold its state static for the entire operation of the device, is especially susceptible due to its large size (number of memory cells), long and constant exposure, and the persistence and severity of the effects of a fault. Until corrected, the fundamental operation of the FPGA routing and logic may be affected. Unless special measures are taken to detect and correct such configuration memory faults, the firm errors may persist for long periods by affecting a large amount of data or even completely destroying the function of the FPGA. Since triple redundancy is considered too expensive on such a large amount of memory, a compromise approach is used, where it takes some time for a scrubber to detect and correct a firm error. In some devices, the entire configuration memory may need to be reloaded from an external source, causing more down-time than is acceptable in many applications.

This is generally more severe and effective than when a flip-flop in the data-path is affected. In this case, the problem is often transient and usually very quickly self-corrected. When the flip-flop is next enabled and clocked, it latches a new, correct value, which often happens within nanoseconds. The distance false value propagates down the data-path is subject to the algorithm design. Also, because there are far fewer flip-flops than configuration memory cells (by two to three orders of magnitude), the probability of such an upset is much lower. In high-reliability applications, single flip-flops can be automatically replaced with triple-redundant flip-flops in the user programmable design by tools provided by Microsemi EDA partners. The two unaffected flip-flops automatically vote away and correct an SEU in the affected flip-flop. PolarFire FPGAs incorporate flash configuration memory that is virtually SEU immune.

8.2 SRAM Scrubbing

All System Controller SRAMs have hardware single-error correction, double-error detection (SECDED) support. Single-bit errors in each location are automatically corrected whenever the location is accessed. A location that is not accessed for a long time may thus accumulate an uncorrectable, and fatal, double-bit error. To prevent this, a periodic scrubbing function is executed, which simply reads all SRAM locations. Whenever a location containing a single-bit error is detected, the corrected data is automatically written back to the SRAM by the memory controller.

The factory cryptoprocessor also contains SRAMs with SECDED support. There is no persistent data maintained within the factory cryptoprocessor memories and data need only survive for a few milliseconds, thus scrubbing is not required. Similarly, the data in the Mailbox SRAM can also be considered ephemeral, and does not require scrubbing.

8.3 System Controller Suspend Mode

For a few high-reliability applications, such as avionics applications, the system controller can be suspended once the initial power-on boot sequence is finished. In general, the system controller goes to sleep and waits for interrupts such as may be generated by system service requests. When in suspend mode, it does not wake up and respond to any such interrupts. Hence, none of the system services can be used. The services that are disabled during suspend mode include (for example, the cryptographic services) the tamper flags or tamper responses such as the zeroization service, and the Flash*Freeze service. The suspend operation is controlled by a user configuration lock bit, and that is set or cleared via the programming tool.

A future version of this user guide will have more information about the Internal Security feature.

9 PolarFire Data Security

All PolarFire "S" grade devices incorporate enhanced data security features, making them highly secure programmable logic devices. Data security protects application data—stored, communicated, or computed at run-time—from being copied, altered, or corrupted. PolarFire devices have a dedicated crypto processor, referred as UserCrypto processor, for data security applications.

9.1 UserCrypto Processor

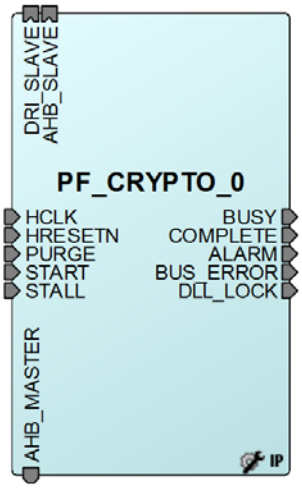
The UserCrypto processor is an Athena TeraFire EXP-F5200B cryptography microprocessor. It provides complete support for the Commercial National Security Algorithm (CNSA) Suite1 and beyond, and includes side-channel analysis (SCA) resistant cryptography using patented leakage reduction countermeasures. These countermeasures provide strong resistance against SCA attacks such as differential power analysis (DPA) and simple power analysis (SPA). The UserCrypto processor is available in PolarFire "S" grade devices.

The UserCrypto processor supports numerous cryptographic algorithms, including the following:

- AES with 128-, 192-, and 256-bit key sizes in ECB, CBC, CFB, OFB, CTR, and GCM modes
- AES key wrap and unwrap
- SHA1, SHA2-224, SHA2-256, SHA2-384, and SHA2-512
- AES-CMAC and AES-GMAC
- HMAC-SHA
- True random number generation (non-deterministic random bit generator plus NIST SP800-90A deterministic random bit generator)
- RSA, DSA, and modular exponentiation (Diffie-Hellman) with key sizes up to 4096-bits
- EC key pair generation, point validation, point multiplication (EC Diffie-Hellman), and ECDSA for
  - NIST P-curves: P-192, P-224, P-256, P-384, and P-521
  - Brainpool curves: P-256, P-384 and P-512
- Key-tree function

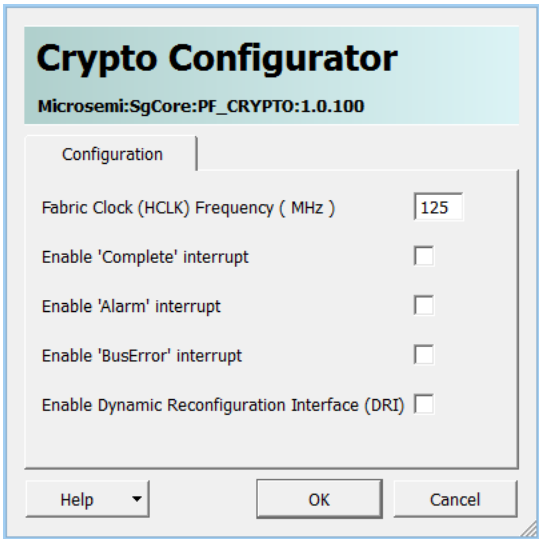
The UserCrypto processor is a hard block in PolarFire FPGAs and its operating frequency ranges from 125 MHz to 189 MHz. It is accessible to a soft processor in the FPGA fabric via the AHB-Lite slave interface for control and primary data input and output. The UserCrypto processor has built-in DMA to offload main processor of doing data transfers between the UserCrypto processor and user memory. The DMA functionality is accessible via an AMBA AHB-Lite master interface. The Libero® SoC design suite provides a Crypto macro in Catalog, which must be integrated with the user design in order to use the UserCrypto processor. The following figure shows the input and output ports of the UserCrypto macro.

Figure 28 • **UserCrypto Macro**



The following figure shows the UserCrypto macro configurator. You can set the frequency of the UserCrypto processor using the configurator.

Figure 29 • **UserCrypto Configurator**



The following table lists and describes the UserCrypto ports. The control and status signals initiate action and obtain status. Corresponding control and status signals are accessible via the dynamic reconfiguration interface (DRI). Contact Microsemi tech-support for information on how to use DRI.

Table 51 • **UserCrypto Port List**

Port Name	Direction	Description
AHB_SLAVE		AHB-Lite slave interface.
AHB_MASTER		AHB-Lite master interface.
HCLK	Input	AHB bus clock.
HRESETN	Input	AHB bus reset. Asserts the functional reset of the UserCrypto block and zeroizes all the internal RAM and registers as PURGE signal.
PURGE	Input	When the signal is set to '1', it initializes the Zeroization of UserCrypto internal RAM and registers. For normal operation, this signal must be tied low. The PURGE input is level sensitive, and if the PURGE pin is still asserted when a purge operation completes, another purge operation is initiated.
START	Input	External execution initiation input when the UserCrypto processor operates in standalone configuration without a host processor connected to the bus interface. Asserting START signal causes the UserCrypto processor to initiate execution. During execution, the status of the UserCrypto processor is reflected by the BUSY and COMPLETE ports. This signal must be tied low when the UserCrypto processor is used as a coprocessor.
STALL	Input	Stalls the UserCrypto processor for a clock cycle, to introduce variance in the external signatures. The STALL input is expected to be generated by a LFSR circuit in the fabric and asserted randomly for a single cycle to achieve the required stall rates. The STALL input must not be asserted until at least three clock cycles after the HRESETN is de-asserted and the DLL has indicated LOCK for three cycles.
BUSY	Output	Execution status signal.

Table 51 • **UserCrypto Port List**

Port Name	Direction	Description
COMPLETE	Output	Computation complete flag.
ALARM	Output	Alarm output. The alarm pin is asserted to indicate an uncorrectable memory error condition.
BUS_ERROR	Output	Reports AHB bus errors. It is set when a HRESP response error is detected by the Athena AHB master. Once set, a reset is required to clear.
DLL_LOCK	Output	DLL lock status.
DRI_SLAVE		Dynamic reconfiguration bus interface to control and monitor UserCrypto functions.

Microsemi provides an Athena TeraFire cryptographic applications library (CAL) to access UserCrypto functions. TeraFire CAL is a C language library that provides functions to access symmetric key, elliptic curve, public key, hash, random number generation, and message authentication code algorithms. It is available for download in the Microsemi Firmware Catalog software. The user application running on the main processor must include CAL APIs to perform the cryptographic operations on the UserCrypto processor.

For information about UserCrypto processor, supported cryptographic functions and their CAL API descriptions, see *Athena TeraFire Cryptographic Algorithm Library (CAL) Users Guide*.

For more information about how to use UserCrypto for data security applications, see *AC464: PolarFire FPGA: Implementing Data Security using UserCrypto Processor Application Note*.

10 Security Glossary

A

Advanced Encryption Standard (AES)

Advanced Encryption Standard (AES) is a 128-bit block cipher with a choice of a 128-bit, 192-bit, or 256-bit key.

AES is based on a state-of-the-art algorithm originally called Rijndael chosen in an international competition and standardized (with selected key sizes) by the United States National Institute of Standards and Technology on October 2, 2000 as FIPS-197. Although selected, it was not officially "approved" by the US Secretary of Commerce until Q2 2001.

AES

See Advanced Encryption Standard (AES), page 70

ANSI

American National Standards Institute (ANSI) is one of the main organizations responsible for furthering technology standards within the USA. ANSI is also a key player with the International Standards Organization (ISO).

Authentication

Authentication refers to the verification of the authenticity of either a person or of data. An example is a message authenticated as originating from its claimed source. Authentication techniques usually form the basis for all forms of access control to systems and data.

Authorization

Authorization is the process whereby a person approves a specific event or action. In companies with access rights hierarchies, it is important that audit trails identify both the creator and the authorizer of new or amended data. It is often an unacceptably high-risk situation for one to have the power to create new entries and then to authorize those same entries oneself.

B

Block Cipher

A block cipher is a type of cipher that works on a block of data. For example, the DES block cipher works on a block size of 64 bits and the AES block cipher works on a block size of 128 bits.

Most block ciphers operate by alternately performing a reversible ("affine") non-linear transformation on groups of bits in the block (often using a small carefully designed look-up table), then permuting bits or small groups of bits and then mixing in key information all in a series of "rounds" that are repeated a number of times with different parts of the key or with sub-keys derived from the key.

Block Cipher Modes of Operation

Since block ciphers only work on relatively small blocks of data such as 64 or 128 bits, some form of unambiguous padding is required for messages that are not exact multiples of the block size, and a scheme for handling multiple blocks is needed.

One way to pad is to add a one to the end of the message, and then fill with zeroes until the next block boundary.

The simplest mode for handling multiple blocks of data is just to encrypt each block individually using the same secret key. This is called Electronic Codebook (ECB) mode, since it is equivalent to using a hypothetical (albeit humongous) code book with 2^{128} input-output pairs recorded in it (for the case of a

128-bit block cipher like AES). Though this efficiently scrambles the contents of each block, it is unsuitable for use in most cases because repeated message blocks are encrypted exactly the same way; a situation that is all too common in real messages.

Popular modes of operation that overcome this problem include:

- Cipher Block Chaining (CBC) mode. In this mode, the output cipher text of each block is used to randomize the input to the next block using a bit-wise XOR operation.
- Counter (CTR) mode increments and then encrypts an ever increasing count value, and then uses the result as keying material that is XORed with the plain text, as in a stream cipher.

The NIST recommended block cipher modes are documented in Special Publication (SP) 800-38 parts A, B, C, D, and E:

- SP 800-38A-Electronic Codebook (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), Output Feedback (OFB), and Counter (CTR) modes
- SP 800-38B-A block cipher-based Message Authentication Code (CMAC)
- SP 800-38C-Counter with Cipher Block Chaining Message Authentication Code (CCM) mode
- SP 800-38D-Galois/Counter Mode (GCM) and Galois Message Authentication Code (GMAC)
- SP 800-38E-XEX Tweakable Block Cipher with cipher text Stealing (XTS) mode, for use with storage devices

C

CERT

The Computer Emergency Response Team (CERT) is recognized as the Internet's official emergency team. It was established in the USA by the Defense Advanced Research Projects Agency (DARPA) in 1988, following the Morris computer Worm incident, which crippled approximately 10% of all computers connected to the Internet.

CERT is located at the Software Engineering Institute (SEI), a US government funded research and development center operated by Carnegie Mellon University, and focuses on security breaches, denial-of-service incidents, providing alerts, and establishing incident-handling and avoidance guidelines. CERT also covers hardware and component security deficiencies that may compromise existing systems.

CERT is the publisher of Information Security alerts, training, and awareness campaigns. The CERT website is www.cert.org.

Checksum

Checksum is a technique whereby the individual binary values of a string of storage locations on your computer are totaled, and the total retained for future reference. On subsequent accesses, the summing procedure is repeated, and the total compared to the one derived previously. A difference indicates that an element of the data has changed during the intervening period. Agreement provides a high degree of assurance (but not total assurance) that the data has not changed during the intervening period.

A checksum is also used to verify that a network transmission has been successful. If the counts agree it is assumed that the transmission was completed correctly.

A checksum refers to the unique number that results from adding up every element of a pattern in a programmable logic design. Typically either a four- or eight-digit hex number, it is a quick way to identify a pattern, since it is very unlikely any two randomly selected patterns ever have the same checksum. Because they are linear functions, checksums are virtually useless in the face of a malicious adversary who can easily find two messages with the same checksum.

See also, Cyclic Redundancy Check (CRC), page 73, Hash Function, page 78 and Message Digest, page 79.

Cipher

A cipher is the generic term used to describe a means of encrypting data. In addition, the term cipher can refer to the encrypted text itself (cipher text, as opposed to the unencrypted plain text). Encryption ciphers use an algorithm, which is a complex mathematical calculation required to scramble the text and a key. Knowledge of the key allows the encrypted data to be decrypted.

Ciphers scramble bits or digits or characters or blocks of bits, whereas codes replace natural language words or phrases with another word or symbol. Modern block ciphers like AES use alternating non-linear substitutions and permutations repeated for a number of "rounds" to encrypt the data. AES, for example, does byte-wide operations on the contents of a 16-byte data block for 10, 12, or 14 rounds, depending upon the key size chosen. Modern ciphers such as AES can be very resistant to mathematical cryptanalysis, requiring an infeasible number of messages encrypted under the same key and a practically infinite amount of computing power to break them.

Code

Codes are a technique for encrypting data, usually in a natural language such as English, by substituting each word or phrase with a secret word or symbol. Because codes require the cumbersome distribution of large code books (essentially a dictionary-like look-up table) to all the participants they are seldom used today. Ciphers are used instead; they work at the alphabet or binary level and require only a relatively short (256-bit) key to be shared by the users.

Codes can be broken through the use of word frequency analysis, and by correctly guessing plain text words from the message. For example, it may be known that a weather report is sent at a certain time each day, and by examining several of these messages from known locations the code for "rain" can be guessed. Codes were traditionally used both for confidentiality, and to make telegraph messages, which were charged by length, shorter. Sometimes codes are cascaded with a cipher, a weak form of double-encryption.

Cloning

In FPGAs, cloning is the act of copying a design without making any changes. No understanding of the design or the ability to modify the design is required.

Configuration

The act of programming an FPGA. For SRAM-based FPGAs this must be done at each system power-up to make it functional. Configuration of SRAM FPGAs require the use of an external configuration device, which is typically a PROM (see the entry for PROM) or other type of nonvolatile memory which must be present in the system.

Since they are nonvolatile, flash- and antifuse-based FPGAs only require configuring once, usually during the system assembly process. Flash FPGAs have the option of being reconfigured, but antifuse FPGAs are intrinsically one-time programmable.

Corrupt Data

Corrupt data is data that has been received, stored, or changed, such that it cannot be read or used by the program that originally created the data.

CPLD

A complex programmable logic device is usually a simple low density programmable logic solution. It typically contains macrocells that are interconnected through a central global routing pool. This type of architecture provides moderate speed and predictable performance. CPLDs are traditionally targeted towards low end consumer products.

CRC

See Cyclic Redundancy Check (CRC), page 73.

Cryptography

Cryptography is primarily concerned with maintaining the privacy of communications and modern methods use a number of techniques to achieve this. Encryption is the transformation of data into another usually unrecognizable form. The only means to read the data is to decrypt the data using the secret key. Other common cryptographic services include ensuring data integrity, authentication of data sources, and digital signatures.

Cyclic Redundancy Check (CRC)

A class of algorithms for computing a short digest value from an arbitrarily long message, similar to a checksum or hash. CRC may also refer to the resulting digest value itself. The "cyclic" in CRC refers to the underlying cyclic codes describing the mathematics of the algorithm. More precisely, CRC algorithms use linear operations in a Galois Field (usually a binary extension field) which are similar to polynomial division using a generator polynomial.

Common CRC algorithms and their generator polynomials have been standardized for many uses, such as detection of bit errors in data transmission. CRC codes are efficient in detecting large bursts of errors, which suits some types of storage media or transmission channels. Examples of some standardized CRC algorithms are CRC-16-CCITT, which is used by Bluetooth (personal area wireless network), CRC-32-IEEE, which is used in 802.3 (wired Ethernet), and MPEG-2 (video).

Because they are linear operations, they are unsuitable for use in the presence of malicious attacks. An attacker can easily create messages with arbitrary CRC digest values. Cryptographic hash functions must be used instead of a CRC in applications such as digital signatures, data integrity, and authentication where there might be non-random errors (malicious attacks).

See also, Hash Function, page 78.

D

Data Encryption

Data encryption is a means of scrambling data so it can be read only by the person(s) holding the key—a password of some sort. Without the key, the cipher (hopefully) cannot be broken and the data remains secure. Using the key, the cipher is decrypted and the data is returned to its original value or state.

For example, using the DES cipher, a key from approximately 72,000,000,000,000,000 possible key variations is randomly generated and used to encrypt the data. The same key must be made known to the receiver so the data can be decrypted at the receiving end. DES can be broken in a matter of hours using a brute-force search because the number of possible keys is low by today's standards.

See also, Public Key Cryptography, page 80.

Data Encryption Standard (DES)

An unclassified cryptographic algorithm adopted by the U.S. National Bureau of Standards (NBS, now called the National Institute of Standards and Technology, NIST) for public and government use as Federal Information Processing Standard (FIPS) 46. It is a 64-bit block cipher with a 56-bit effective key length.

DES is a data encryption standard for the scrambling of data to protect its confidentiality. It was developed by IBM in cooperation with the United States National Security Agency (NSA) and published in 1974 by NIST. It is extremely popular and, because at the time it was thought to be so difficult to break, with approximately 72,000,000,000,000,000 possible key variations, was banned from export from the USA. However, restrictions by the US Government on the export of encryption technology to the countries of Europe and a number of other countries were lifted in 2000.

DES was cracked by researchers in 96 days in 1997 by the DESSHALL project and again in 41 days by distributed.net, both projects using thousands of distributed personal computers, where they showed that DES was susceptible to brute force attacks. One of the final blows to the short 56-bit key length of DES was in 1998 when the Electronic Frontier Foundation (EFF) and Cryptography Research, Inc. (CRI) discovered several DES keys, first in 56 hours and then later in only 22 hours, using a custom-designed computer called DES Cracker. The industry then turned to Triple DES, which uses DES three times, as a

short term standard to secure transactions. Generally sluggish performance caused an outcry that resulted in a new standard. The NIST has since standardized the Advanced Encryption Standard (AES), based on the Rijndael algorithm, as recommended for all new block cipher applications, although Triple DES is still used extensively in the finance industry for legacy reasons.

Decryption

The process by which encrypted data is restored to its original form in order to be understood/usable by another computer or person.

Denial of Service

Denial of service (DoS) attacks deny service to valid users trying to access a site. Consistently ranked as the single greatest security problem for IT professionals, DoS attack is an Internet attack against a website whereby a client is denied the level of service expected. In a mild case, the impact can be unexpectedly poor performance. In the worst case, the server can become so overloaded that it crashes the system.

DoS attacks are not primarily intended for theft or corruption of data, and are often executed by persons who nurse a grudge against the target organization. The following are the main types of DoS attacks:

- Buffer Overflow Attacks whereby data is sent to the server at a rate and volume that exceeds the capacity of the system, causing errors. This could be just a single long message that exceeds the size of the receiving buffer.
- SYN Attack. This takes places when connection requests to the server are not properly responded to, causing a delay in connection. Although these failed connections eventually time out, they can result in denial of access to other legitimate requests for access should they occur in large volumes.
- Teardrop Attack. The exploitation of TCP/IP protocol features whereby large packets of data are split into bite-sized chunks, with each fragment being identified to the next by an offset marker. Later the fragments are supposed to be reassembled by the receiving system. In the teardrop attack, the attacker enters a confusing offset value in the second (or later) fragment, which can crash the recipient's system.
- Ping Attack. This is where an illegitimate attention request or ‘ping’ is sent to a system, with the return address being that of the target host (to be attacked). The intermediate system responds to the ping request but responds to the unsuspecting victim system. If the receipt of such responses becomes excessive, the target system is unable to distinguish between legitimate and illegitimate traffic.
- Viruses. Viruses are not usually targeted but where the host server becomes infected, it can cause a DoS.
- Physical Attacks. A physical attack may be little more than cutting the power supply, or perhaps the removal of a network cable.

DES

See Data Encryption Standard (DES), page 73.

Differential Power Analysis (DPA)

An analysis technique that relies on multiple measurements of a security device's instantaneous power consumption in order to recreate a secret being manipulated inside the device. Simple and Differential Power Analysis were first reported by Paul Kocher et al in 1989. Generally this class of techniques uses statistical methods to amplify the effects of small unintentional leakages of the secret information in power consumption measurements, buried in large amounts of noise.

For example, if the same secret key is used to process multiple independent blocks of data, a DPA attack might be mounted to determine the secret key using anywhere from a handful of power consumption traces to over a million, depending on the magnitude of the leak, the amount of noise that may be obscuring the secret data, and what countermeasures are used. Systems that handle large amounts of data using the same key, or which can be repeatedly be given random or chosen input data which is then processed using the secret key, are especially vulnerable to DPA.

Diffie-Hellman Key Exchange

The Diffie-Hellman key exchange algorithm, named after Whitfield Diffie and Martin Hellman, was the first public key algorithm ever published, in 1976. The third inventor was Ralph Merkle. With it, they revolutionized the field of cryptology, and made secure communication over the Internet feasible.

It is based upon the difference in difficulty of a particular function and its inverse, namely the ease of exponentiation and the difficulty of computing the discrete logarithm (both) in a finite field. When the numbers involved are large (i.e., over one thousand bits) the difference in difficulty is approximately 30 orders of magnitude, and grows with the size of the numbers.

The Diffie-Hellman protocol allows two entities (computers or people) who do not have nor have ever had a secure channel between them to compute a common secret using public information they send to each other. Anyone eavesdropping on the conversation would find it computationally infeasible to learn the shared secret, even though they see all the messages. This is because each of the parties to the computation holds one secret they do not transmit, but use in the exponentiation formula to compute a value that is practically impossible to reverse; and this is the value that is sent over the insecure channel.

Prior to this invention, secret communications always involved having a shared secret key. This shared key had to be transmitted securely between the parties by a trusted courier or some similar means before encrypted communication over an insecure medium such as radio or telegraph could be done using the shared secret key. Since each possible pair of entities might need a unique shared key, the system did not scale well to large groups where the number of combinations can be exceedingly large. It can thus be claimed that public key cryptography made practical encrypted communications between large numbers of parties, such as shopping with credit cards on the Internet, that was not feasible before.

Digital Signatures

With the advent of public key cryptography a number of new cryptographic services were born, with digital signatures perhaps being the most important.

The concept of digital signatures is that the signer performs a computation using a secret key that only the signer knows, but which can be confirmed by anyone having the matching public key.

Using the RSA cryptosystem, this is done mainly by interchanging the usual role of the private and public keys: In "normal" encryption, any sender encrypts the message using the recipient's public key and the recipient decrypts it using the private key that only the recipient knows. In the RSA digital signature algorithm, the signer "encrypts" the message using the private key that only the signer knows, and any verifier can "decrypt" the signature and verify it is the same as the message using the freely available public key.

Since only the signer has a copy of the private key, it is difficult for the signer to repudiate any valid signatures. This is different from symmetric (shared key) systems where at least two parties must be in possession of a key for it to have any use.

In practice, the whole message is not signed. Because of computational efficiency, and to reduce the size of the signature that has to be transmitted along with the message, a hybrid scheme is used. The message is first hashed; that is, a short digest is computed from the message, and it is this digest that is signed using the private key. The verifier also hashes the received message, and verifies the signature matches the hash using the public key. Standards such as PKCS#1 specify other details important to the security of the system such as how padding is done, and the use of random nonces.

There are variations of this hybrid signature scheme using the ElGamal and elliptic curve cryptosystems.

Disable

Disabling is the process by which hardware or software is deliberately prevented from functioning in some way. For hardware, it may be as simple as switching off a piece of equipment, or disconnecting a cable. It is more commonly associated with software, particularly shareware or promotional software, which has been supplied to a user at little or no cost, to try before paying the full purchase or registration fee. Such software may be described as "crippled", in that certain functions, such as saving or printing files, are not permitted. Some in-house development staff may well disable parts of a new program, so that the user can try out the parts that have been developed, while work continues on the disabled functions.

Disabling is also often used as a security measure. For example, the risk of virus infection through the use of infected floppy diskettes or USB thumb drives can be greatly reduced by disconnecting a cable within the PC, thereby disabling the drive. Even greater protection is achieved by removing the drive altogether.

E

Electromagnetic Analysis (EMA)

A form of side-channel analysis where the unintentional information leakage from the cryptographic system is via electromagnetic (EM) emissions. Electromagnetic emissions have been a well-known source of leakage, prompting the US government to specify EM requirements for secure applications in what are called TEMPEST requirements. In one example of EM leakage, the van Eck radiation of a display terminal is read from a distance of hundreds of meters using simple equipment.

Many power analysis (PA) classifications have an EMA analog where a similar attack can be performed using essentially the same method for EMA as for PA. For instance, differential electromagnetic analysis (DEMA) is the analog of differential power analysis (DPA), and can be used to extract the AES key, for example, from an unprotected device using an RF antenna and amplifier instead of a current monitor. One important difference is that in EMA the usable signal is often more strongly modulated on harmonics of the fundamental frequencies due to the better propagation properties of higher frequencies; therefore demodulation is often used to bring these harmonic-related signals back to baseband before completing the analysis. Often, it is possible to focus the area of the attack by the placement of the antenna, resulting in an improved signal-to-noise ratio (from the analyst's perspective).

Elliptic Curve Cryptography (ECC)

Elliptic curve cryptography is a public key cryptographic system defined using elliptic curve polynomials in finite fields. The important principle is related to the Diffie-Hellman problem of finding discrete logarithms in finite fields, but instead of exponentiation the group operator is scalar point multiplication. Since some of the most efficient (non-quantum) algorithms available for finding discrete logarithms do not work on elliptic curves, the key sizes required for elliptic curves can be much shorter than for the Diffie-Hellman (or RSA) cryptosystems for a roughly equivalent security strength.

As an example, for a security strength of around 128 bits, i.e., requiring an attack with approx. 2128 operations, comparable to brute-force attack on AES-128, ECC requires a key size of 256 bits, whilst RSA requires around 3072 bits. As a result, ECC is generally substantially more computationally efficient than RSA. ECC's "hard problem" is susceptible to Shor's attack using a quantum computer. When suitable quantum computers are available, ECC will become ineffective at providing security.

Encryption

The process by which data is temporarily rearranged into an unreadable or unintelligible form for confidentiality, transmission, or other security purposes.

Entropy

In information theory, entropy is a measure of the uncertainty of a system. For example, if all the bits of an n-bit binary number are unbiased (equal probability of a one or zero) and independent (not correlated with any other bits) and are unknown, then the number "contains" n bits of entropy and is said to have full entropy.

In this case, there would be no better method of guessing the number than a brute force search attempting every possible value (2n values), with an expected match after about half of the values have been tried. However, if the bits were known to be biased (for example: 1/3 were randomly selected as zero, and 2/3 as one), then the entropy would be less than n bits and a more efficient search could be performed that started by guessing more ones than zeroes, with an expected match much earlier than in the unbiased case.

In cryptographic applications it is usually critically important that random numbers, such as those used for secret keys, have full entropy.

There is a beautiful and unexpected relationship between entropy as used in information theory and entropy as used in the physical sciences (such as thermodynamics), but in most practical applications the two uses are distinct.

F

Fault Analysis

Fault attacks attempt to break the security of a cryptographic implementation by injecting energy in the form of voltage glitches on the power supply, or light or concentrated electromagnetic energy to generate a fault in the device. Other ways to induce faults can be to operate the temperature or voltage outside the normal ranges. Faults can be used by an adversary in different ways, depending on the precise fault and the design of the system. For example, if a microcontroller can always be made to take a certain branch, whether or not a passcode is matched properly, the passcode protection may be made ineffective. Another broad class of fault attacks are called differential fault analysis (DFA) where if the correct and one or more faulted outputs of a cryptographic calculation using the same secret key can be obtained, the key may be extracted. Fault analysis a subset of the general class of active side channel analysis.

Firmware

Firmware is a sort of halfway house between hardware and software. Firmware often takes the form of a device that is attached to or built into a computer–such as a ROM chip–which performs some software function but is not a program in the sense of being installed and run from the computer's storage media.

Flash FPGA

A flash-based FPGA uses flash memory technology to control the switching of the interconnect and the operation of the logic elements. Flash-based FPGAs are nonvolatile, live on power-up, and reprogrammable They are and relatively secure from reverse engineering or cloning since the programming bitstream is only required to be loaded once, during the initial configuration. This can be performed either in a trusted location, or using strong cryptographic techniques in less trusted locations.

Most flash FPGAs also allow for secure field upgrades using encrypted bitstream files and a decryption key which was loaded in the nonvolatile memory during the initial configuration process. The discovery of the possibly millions of configuration bit values stored in the internal nonvolatile flash memory cells is considered a very difficult problem, thus contributing to the security of flash FPGAs.

FPGA

A field programmable gate array is a very complex programmable logic device (PLD). The FPGA usually has an architecture that comprises a large number of simple logic blocks, a number of input/output pads, and a method to make the desired connections between the elements. The largest programmable logic devices have gate counts running into the millions, and modern devices often have many ancillary hardware blocks such as microprocessor units (MPUs), phase-locked loops (PLLs), static random access memory (SRAM), specialized digital signal processing (DSP) elements, embedded nonvolatile memory (eNVM).

These devices are user customizable and programmable on an individual device basis. They are valued by designers for their flexibility.

H

Hacker

A hacker is an individual whose primary aim in life is to penetrate the security defenses of large, sophisticated, computer systems. A truly skilled hacker can penetrate a system right to the core and withdraw without leaving a trace of the activity. Hackers are a threat to all computer systems that allow access from outside the organization's premises, and the fact that most hacking is just an intellectual challenge should not allow it to be dismissed as a prank. Clumsy hacking can do extensive damage to systems even when such damage was not intentional.

In 2015 the US government issued a report from the Defense Cybersecurity Culture and Compliance Initiative (DC3I) that indicated there were around 100,000 attempted malicious network attacks on Department of Defense assets every day.

Hash Function

A cryptographic hash, also called a message digest, is a publicly-known function that takes as its input a message of (almost) any length and compresses it into a random-like short message called a digest or fingerprint. "Hash" may refer to either the function or the output digest value itself.

Commonly used digest output lengths are from 160 to 512 bits. Hash functions are important components of integrity, authentication, and digital signature schemes, amongst other uses.

A good cryptographic hash must have several properties:

1. **Pre-image resistance**—it must be infeasible to determine any part of the input message from the output digest.
2. **Second pre-image resistance**—it must be infeasible to generate any input message with a given output digest.
3. **Collision resistance**—it must be infeasible to find any two input messages with the same output digest.

These imply a strong one-way-ness property for cryptographic hash functions. For a good hash function, if even one bit of the input message is changed, roughly one-half of the output bits changes pseudo-randomly.

Commonly-used hash functions are MD5, SHA-1, and the SHA-2 family of hashes, including SHA- 256, SHA-384, and SHA-512. Though still in widespread use, MD5 is considered broken, and SHA-1 has some serious weaknesses. The US government agency NIST recently completed a competition for a new family of hash functions called SHA-3 that must have better security than the current standard hash functions. An algorithm called Keccak was selected as the winner. It uses different principles than most prior hash functions, and is very efficient in hardware implementations.

Cryptographic hashes are related to, but not the same as hashes used in computer science for creating tables for looking up data by value. Those hash functions do not have the three security properties (above) required for a cryptographic hash and as a result must never be used in a cryptographic (adversarial) setting.

See also, Cyclic Redundancy Check (CRC), page 73 and Security Strength, page 82.

HEX / Hexadecimal

Hexadecimal, or hex, is a base 16 numbering system (as opposed to the usual decimal base 10). Hex is a useful way to express binary computer numbers. A byte is normally expressed as having 8 binary bits. Two hex characters of four bits each, called nibbles, represent eight binary digits, also known as a byte. Nibbles are sometimes represented using the sixteen 8-bit ASCII character set symbols 0-9 and a-f (or A-F) for human consumption such as when displayed or printed.

IIAP

See In-Application Programming (IAP), page 78.

In-Application Programming (IAP)

IAP is the ability of a microcontroller to run an application that reconfigures (reprograms) its own nonvolatile program code storage. Some flash FPGAs having a built-in microcontroller natively support both IAP and ISP.

See also the entries for "In-System Programming (ISP)".

In-System Programming (ISP)

ISP is the ability to program and reprogram an FPGA that is mounted on a circuit as part of a functional system. Flash and SRAM-based FPGA technologies support ISP.

Intellectual Property (IP)

Intellectual property is defined as creative, technical, and intellectual products, often associated with custom circuit designs implemented in ASIC or programmable logic architectures.

Invasive Attack

Invasive attack is an attack on a semiconductor to determine its functionality and requires physical entry to the part. Typical methods include probing, etching, and FIB (focused ion beam) intrusion.

See also the entries for "Noninvasive Attack" and "Semi-Invasive Attack".

ISP

See In-System Programming (ISP), page 78.

M

Malicious Code

Malicious code includes all and any programs (including macros and scripts) that are deliberately coded in order to cause an unexpected (and usually unwanted) event on a PC or other system. However, whereas antivirus definitions (vaccines) are released weekly or monthly, they operate retrospectively. In other words, someone's PC has to become infected with the virus before the antivirus definition can be developed. In May 2000, when the Love Bug was discovered, although the antivirus vendors worked around the clock, the virus had already infected tens of thousands of organizations around the world, before the vaccine became available.

Message Authentication Code

A Message Authentication Code (MAC) is similar to a hash function in that it computes a random-like output digest from any size input message, but unlike a hash, which is a public function that anyone can compute, a MAC uses a secret key so that only those in possession of the secret can correctly create or verify it.

Message Digest

See Hash Function, page 78.

Modes of Operation

See Block Cipher Modes of Operation, page 70.

N

National Institute of Standards and Technology (NIST)

NIST was formerly the National Bureau of Standards (NBS). NIST is the government agency that sets weights and measures for the United States. It is an agency of the Commerce Department. In security and cryptography, NIST works closely with the National Security Agency (NSA), a part of the Defense Department, to set government standards and make recommendations for private sector use.

Nonce

A number used only once. Nonces are an important element of many protocols because they help protect against replay attacks. By incorporating a unique nonce in the protocol the attacker cannot replay data from an earlier run of the protocol that, by definition, used a different nonce. Nonces are also often required for initialization vectors such as those used with some block cipher modes of operation, or stream ciphers. If the same initialization vector is used with the same key on more than one message, the security of the cipher mode can be very seriously compromised. Nonces are also used in some types of digital signatures.

Common ways of generating nonces are by counting, using a time stamp, or using a sufficiently large random number whose chance of repeating is vanishingly small. The best choice depends upon the circumstances, because each of these has its own difficulties and advantages. For instance, in many systems it is very difficult to be sure of a secure time source. With a counter, the issue is to make sure that it is never reset or a count value used twice, even if the power supply is tampered with. In other systems there may not be a good source of entropy with which to create sufficiently large random numbers.

Noninvasive Attack

A noninvasive attack is an attack on a semiconductor to determine its functionality that does not require physical entry to the part. Types of attacks include actively varying voltage levels to gain access, and passive side-channel analysis.

See also the entries for "Invasive Attack", "Semi-Invasive Attack" and "Side-Channel Analysis".

Nonvolatile

A device is nonvolatile if it does not lose its contents when its power is removed. Nonvolatile memory is useful in microcomputer circuits because it can provide instructions for a CPU as soon as the power is applied, before secondary devices, such as disk, can be accessed. Nonvolatile memories include metal-mask read-only memory (ROM), fusible-link programmable ROM (PROM), ultra-voilet- erasable electrically-programmable ROM (UV-EPROM), and electrically-erasable PROM (EEPROM) including "flash" memory, a special type of EEPROM where the memory is erased in large blocks rather than by individual bytes or words, making it much faster and also less expensive.

O

Overbuilding

Unscrupulous contract manufacturers (CM) overbuild on a program or contract and sell the excess on the gray market.

P

Power Analysis

See Side-Channel Analysis, page 82 (a super-set of power analysis), Simple Power Analysis, page 83, and Differential Power Analysis (DPA), page 74 (both sub-types).

Public Key Cryptography

Public key cryptography is based upon the revolutionary principle that instead of using a shared secret key for two or more parties to communicate privately, as in all ciphers and codes before 1976, a key can have two parts: a public part and a secret part. The public part may be communicated to anyone and does not have to be kept secret. It can be used for encryption, thus allowing anyone in the world to encrypt a message intended for a given recipient. Only the recipient, namely the holder of the secret part of the key, can perform the decryption.

The first public key scheme, called the Diffie-Hellman key exchange algorithm, was published by Whitfield Diffie, Martin Hellman, and Ralph Merkle, in which they used mathematics based upon the difficulty of the discrete logarithm problem to generate a shared secret key between two parties that had no prior secret communication. This was later expanded into the ElGamal encryption system for enciphering messages. Shortly after, Ron Rivest, Adi Shamir, and Len Adleman published the now well known RSA encryption scheme named after them, based upon the difficulty of factoring large primes.

Besides greatly simplifying key distribution between anonymous parties, public key cryptography also introduced a new cryptographic service called digital signatures. The holder of the secret key "signs" a message with a message-dependent code only they can generate, and anyone in possession of the public key can verify the integrity of the data and the correctness of the signature. Since only one person

holds the private key (unlike in symmetric key systems where at least two people have the key), it makes it much more difficult for the signer to later repudiate their signature.

Though attributed to the inventors mentioned above who were the first to publish their results, it is now known that public key cryptography had been invented a few years earlier by James Ellis, Clifford Cocks and Malcolm Williamson, employees of the General Communications Headquarters (GCHQ), a British government agency, which kept their results secret and largely failed to recognize the importance of the discoveries.

R

Random Numbers

Random numbers are used extensively in cryptography, for generating secret keys and nonces, for example. In most implementations, they are binary numbers. The random numbers must be unknown and unpredictable to an adversary. An n-bit binary number which is completely unpredictable and unknown to an adversary is said to contain n bits of entropy; if the adversary has a better than 50%- 50% chance of guessing some of the bits, the entropy is reduced.

True random numbers are derived from an unpredictable physical source, most often some form of electrical noise although radiation decay and some other physical processes are also sufficiently random though less practical. If each bit generated by the physical process is unbiased and uncorrelated with all the other bits then it has one bit of entropy. By gathering many such bits, one can accumulate a large amount of entropy.

Pseudo-random numbers are derived from a deterministic computational process. With good algorithms pseudo-random bits can be computationally indistinguishable from true random bits. However, no matter how many such bits are generated, the entropy content is limited by the lesser of the initial true random seed used to initialize the computation process and the number of bits of internal state storage. If an adversary were able to learn the internal state of a pseudo-random generator (by guessing or other means) he could predict all future values, and may even learn something about past values.

Important standards related to random numbers include:

- SP 800-90A-(NIST) Recommendation for Random Number Generation Using Deterministic Random Bit Generators. Part A covers deterministic random bit generators. Parts B and C (still in draft form) cover non-deterministic entropy sources and how to combine them to create hybrid random bit generators
- FIPS 140-2 Annex C-(NIST) Approved Random Number Generators for FIPS PUB 140-2
- SP 800-22-(NIST) A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications
- Test Suite-(BSI) Random number Test Suite
- AIS-31-(BSI) Functionality classes and evaluation methodology for physical random number generators

Reverse Engineering

Reverse engineering is the act of examining a design to understand exactly how it works, perhaps with the intent to copy the design. The design is then altered to differentiate it from the original design for the purpose of improving upon it or to prevent legal action because of the theft, or to insert a "Trojan Horse". Also, reverse engineering is sometime used to determine if any patents are being violated. Some applications of reverse engineering are legal depending on the subject and the legal jurisdiction, while other cases may be considered theft.

S

Security Strength

Security strength is a rough measure of the work effort, log base 2, required to attack a given cryptographic problem. For a well-designed block cipher, the best approach an attacker has is a brute force search over all the possible keys. In this case the security strength, measured in bits, is the same as the length of the key (in bits). For example, AES-128 (the version of AES using a 128-bit key) has an estimated security strength of 128 bits since the best known attack is a brute force search of all 2128 keys.

For a well-designed hash function, the security strength varies depending upon which of the security properties is being depended upon in its usage (see the entry for Hash Function). For pre-image resistance and 2nd-pre-image resistance, the security strength is the same as the digest output size (in bits). For collisions, the security strength is very nearly half the number of bits in the output. The reduced strength is due to the Birthday Attack, which is applicable in this situation.

For public key algorithms, the security strength is a complicated function of the key size but also depends upon the most efficient attack algorithm known. Since the most efficient attacks on RSA or Elgamal do not work on elliptic curve algorithms, shorter keys can be used with elliptic curve cryptography for a given security strength. For elliptic curve algorithms, the keys must be roughly twice as long as for symmetric algorithms such as AES. RSA, Diffie-Hellman, and Elgamal all require comparable (to each other) but much longer keys. For example, a one-thousand bit RSA key is roughly equivalent in security strength to an 80-bit symmetric key and a 160-bit elliptic curve key.

Not all block ciphers and hash functions have the ideal security strength shown above. If some attacks are known that reduce the work factor to find the key (or pre-image, or collision, etc.) caused by a weakness in the algorithm, then the security strength is correspondingly downgraded. For instance, the MD5 hash algorithm design in 1994, which has a digest size of 128 bits, must have a collision resistance security factor of 64 bits (which in itself is marginal), but attacks had been found by 2006 that reduced the work factor to less than 224, (one trillion times easier) making it unsuitable for cryptographic applications since the latest/best attack algorithm known can find an MD5 collision in less than one minute on a standard notebook computer.

Security strength is often equated with the length of time the algorithm or secret data is used. For short term (ephemeral) use, 80 bits may be enough for strong security, but for data that has to last a few years 100 bits or more is recommended, and for data that may have to keep secret for several decades, 128 bits is recommended. This is because attacks only get better, and computing equipment has been getting faster and cheaper due to Moore's Law.

Grover's algorithm, applicable to quantum computing, is expected to reduce the security strength of most cryptographic algorithms by a square-root factor, i.e., by about halving the security strength measured in bits. For example, to maintain a security strength of 128 bits in a post-quantum world, one should use AES-256. Shor's algorithm, which is applicable to many public key algorithms such as ECC, RSA, and DH (but not most block ciphers or hashes) has a more devastating effect on the security strength, making these algorithms next to worthless.

Semi-Invasive Attack

A semi-invasive attack is an attack on a cryptographic device such as an integrated circuit which may involve removing all or part of the package, but does not require internal probing or cutting of circuit lines. Instead, the attack is carried out using optical or electron microscope observations or by injecting (temporary) faults optically or electromagnetically, which do not require the active device to be touched. This family of attacks is generally less expensive to conduct than invasive attacks but more expensive than other types of active fault attack or passive side-channel analysis.

See also, Invasive Attack, page 79, Noninvasive Attack, page 80, and Side-Channel Analysis, page 82.

Side-Channel Analysis

Passive side-channel analysis is a noninvasive (or occasionally a semi-invasive) analysis technique which attempts to break the security of a cryptographic system by monitoring information unintentionally leaked via side-channels. These side-channels could be power consumption, electromagnetic

emissions, optical emissions, thermal signatures, or timing of response times, for example. As all "real world" implementations of cryptographic systems have unintended side-channels, they represent a serious threat to the security provided by these systems.

Active side-channel analysis attempts to break the security by using light, electrons, electromagnetic energy, or other active sources of energy to probe or disrupt the target system. Sometimes active and passive techniques are combined.

See also, Simple Power Analysis, page 83, Differential Power Analysis (DPA), page 74, Electromagnetic Analysis (EMA), page 76, and Fault Analysis, page 77.

Simple Power Analysis

Simple power analysis is a side-channel analysis technique based upon one or just a few measurements of a security device's power consumption. Information about secrets being manipulated inside the device are unintentionally leaked out via the instantaneous power consumption of the device. In some cases, a secret key can be read more-or-less directly from simple observations of a single oscilloscope trace.

SRAM FPGA

An SRAM FPGA is an FPGA that utilizes SRAM (Static Random Access Memory) technology to configure the interconnect and to define the logic. SRAM FPGAs are reprogrammable, volatile, and require a boot-up process to initialize. SRAM FPGAs are generally considered less secure than flash or antifuse technology based FPGAs because the design configuration bitstream has to be loaded from an external component at each power-up cycle.

See also, Differential Power Analysis (DPA), page 74.

T

Tamper Detection

Tamper detection is an alarm set off when any of a number of possible tamper detection sources is triggered. Common tamper detectors for high-end security integrated circuits include voltage, clock and temperature alarms, internal redundancy violations, physical tampering alarms such as a failure of a mesh covering important circuits, etc.

See also, Zeroization, page 84, which is one possible response to a tamper detection alarm.

Tamper Resistant Packaging

Often used in smart card systems, tamper resistant packaging is designed to render electronics inoperable if the product is physically (invasively) attacked. Tamper evident packaging can also be used to deter tampering attacks.

See also, Zeroization, page 84, and Tamper Detection, page 83.

Timing Analysis

Timing analysis uses detectable data dependent variations in the time to perform calculations to determine secrets contained in the data. The time may be detected by monitoring external signals, unintended side channel leakage, network response time, cache hits, or other means. To prevent timing analysis constant time forms of common cryptographic algorithms are used.

V

Volatile

As applied to memory technology, volatile memory loses its data when power is removed. SRAM and DRAM technologies are volatile, while flash, EEPROM, and fuse-type memories are nonvolatile. The inability of an SRAM-based FPGA to maintain its configuration when power is removed is a function of the volatile memory technology upon which it is based. Thus, SRAM-based FPGAs require additional

external nonvolatile memory components, and the sensitive data must be securely transported from the external device to the FPGA at each power-up cycle.

Z

Zeroization

Active zeroization is used to erase critical information, followed by verification that the erase operation was successful. It can be used as one of many possible responses to a tamper detection alarm.

See also, Tamper Detection, page 83.

Passive zeroization is erasure of nonvolatile memory by removal of the power source. Verification may be infeasible in this case.