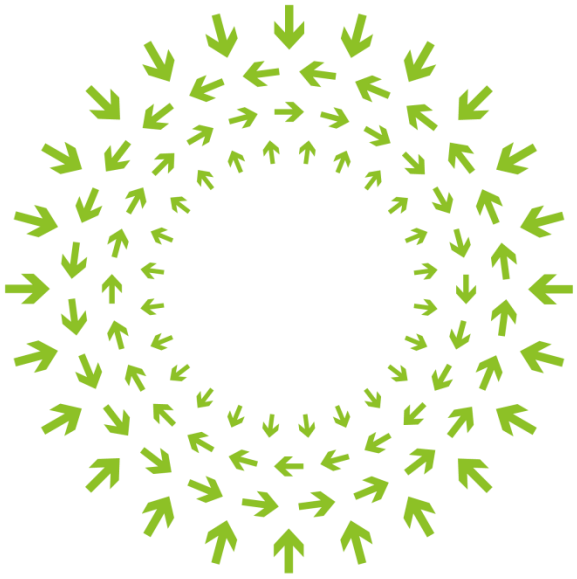


OPEN

Compute Project

Project Cerberus Processor Cryptography Specification

Author:
Bryan Kelly, Principal Firmware Engineering Manager, Microsoft

Revision History

Date	Description
08/28/2017	Version 1.0

As of November 1, 2017, the following persons or entities have made this Specification available under the Open Web Foundation Final Specification Agreement (OWFa 1.0), which is available at http://www.openwebfoundation.org/legal/the-owf-1-0-agreements/owfa-1-0

Microsoft Corporation.

You can review the signed copies of the Open Web Foundation Agreement Version 1.0 for this Specification at Project Olympus License Agreements, which may also include additional parties to those listed above.

Your use of this Specification may be subject to other third party rights. THIS SPECIFICATION IS PROVIDED "AS IS." The contributors expressly disclaim any warranties (express, implied, or otherwise), including implied warranties of merchantability, non-infringement, fitness for a particular purpose, or title, related to the Specification. The entire risk as to implementing or otherwise using the Specification is assumed by the Specification implementer and user. IN NO EVENT WILL ANY PARTY BE LIABLE TO ANY OTHER PARTY FOR LOST PROFITS OR ANY FORM OF INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER FROM ANY CAUSES OF ACTION OF ANY KIND WITH RESPECT TO THIS SPECIFICATION OR ITS GOVERNING AGREEMENT, WHETHER BASED ON BREACH OF CONTRACT, TORT (INCLUDING NEGLIGENCE), OR OTHERWISE, AND WHETHER OR NOT THE OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

CONTRIBUTORS AND LICENSORS OF THIS SPECIFICATION MAY HAVE MENTIONED CERTAIN TECHNOLOGIES THAT ARE MERELY REFERENCED WITHIN THIS SPECIFICATION AND NOT LICENSED UNDER THE OWF CLA OR OWFa. THE FOLLOWING IS A LIST OF MERELY REFERENCED TECHNOLOGY: INTELLIGENT PLATFORM MANAGEMENT INTERFACE (IPMI); I^2C IS A TRADEMARK AND TECHNOLOGY OF NXP SEMICONDUCTORS ; EPYC IS A TRADEMARK AND TECHNOLOGY OF ADVANCED MICRO DEVICES INC.; ASPEED AST 2400/2500 FAMILY PROCESSORS IS A TECHNOLOGY OF ASPEED TECHNOLOGY INC.; MOLEX NANOPITCH, NANO PICOBLADE, AND MINI-FIT JR AND ASSOCIATED CONNECTORS ARE TRADEMARKS AND TECHNOLOGIES OF MOLEX LLC; WINBOND IS A TRADEMARK OF WINBOND ELECTRONICS CORPORATION; NVLINK IS A TECHNOLOGY OF NVIDIA; INTEL XEON SCALABLE PROCESSORS, INTEL QUICKASSIST TECHNOLOGY, INTEL HYPER-THREADING TECHNOLOGY, ENHANCED INTEL SPEEDSTEP TECHNOLOGY, INTEL VIRTUALIZATION TECHNOLOGY, INTEL SERVER PLATFORM SERVICES, INTEL MANAGABILITY ENGINE, AND INTEL TRUSTED EXECUTION TECHNOLOGY ARE TRADEMARKS AND TECHNOLOGIES OF INTEL CORPORATION; SITARA ARM CORTEX-A9 PROCESSOR IS A TRADEMARK AND TECHNOLOGY OF TEXAS INSTRUMENTS; GUIDE PINS FROM PENCOM; BATTERIES FROM PANASONIC. IMPLEMENTATION OF THESE TECHNOLOGIES MAY BE SUBJECT TO THEIR OWN LEGAL TERMS.

Contents

Table of Figures

1 Summary

Throughout this document, the term “Processor” refers to any and all Central Processing Unit (CPU), System On Chip (SOC), Micro Control Unit (MCU), and Microprocessor architectures. The document details the required ROM boot loader Secure Boot and Flash integrity protection. The Processor must implement all required features to establish a hardware based Root of Trust to authenticate and protect code and data that executes within the Processor.

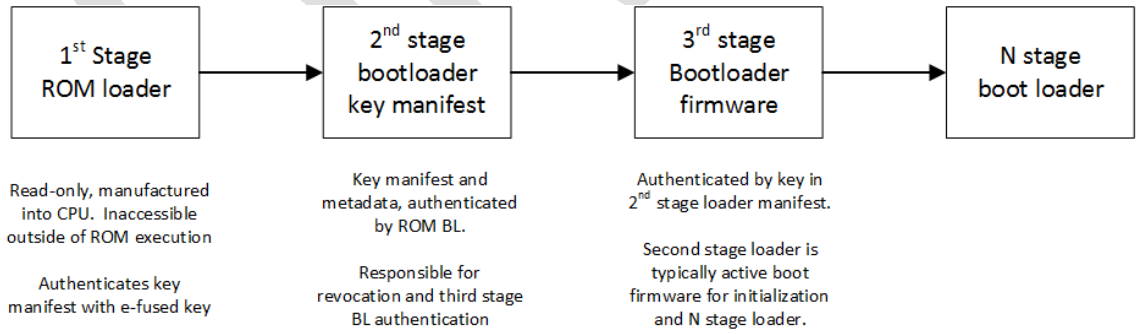
Processors that intrinsically fail to meet all of these requirements, must implement the interposed Cerberus RoT described in this document.

2 Secure Boot, Confidentiality and Cryptography Signing Requirements

2.1 Secure Boot

Secure boot adds cryptographic checks to the boot process. The purpose is to verify the integrity and authenticity of firmware or firmware boot loaders that follow the immutable first stage ROM boot loader. This firmware is typically mutable and referred to as Second Stage boot loader. They immediately follow the execution of the ROM boot loader, with a key manifest being the second stage loader, followed by the third stage boot firmware

Figure 1 Firmware Boot Loader



Secure boot establishes a chain of trust for the boot process. Starting with the ROM as the root of trust, all subsequent firmware components get authenticated. The secure boot process uses an RSA public key signature algorithm whereby the private key is used to generate an image signature/digest and the

public key typically e-fused into the processor is used to verify the signature. The public key does not need to be kept confidential, but the initial ROM bootloader public key should not be replicable.

2.2 Flash Encryption

Flash encryption adds obfuscation and confidentiality to firmware and the data stored within the flash media. Flash encryption is separate from secure boot, but enhances security by obfuscating the data contained on persistent firmware load store. This additional mechanism is useful for added obfuscation when the processor uses external flash storage.

2.3 Project Cerberus Security Requirements

There are two Processor security models supported in Project Cerberus:

1. High Security (HS), which includes e-fused public key or hash of public key and fully encrypted firmware images for secure boot and secure runtime firmware update. Authentication is established through RSA public/private key pair digest comparison during boot, while confidentiality is enforced through AES encryption of the entire firmware package.

   HS processor should support ROM crypto for verification of a RSA digest of a key manifest providing the keys to authenticate the secure first boot loader that subsequently authentications and decrypts firmware into secure memory to ensure the integrity of the execution environment. The ROM accessible RSA public key is fused into OTP memory.

   The processor should support a minimum of two One Time Programmable (OTP) Memory e-fuses. At a minimum: 1 x 256 bit hash of public key. 1 x 256 bit AES symmetric key.

   The processor should support additional OTP memory for the revocation of the key manifest or equivalent functionality.

2. General Security (GS), which includes encrypted hashed checksum signature.

   There are no requirements on the ROM boot loader and Hardware secure boot.

   Runtime updates are validated by Firmware without e-fuses.

   Firmware is structured to have writable regions separate from the executable and code regions. The executable and code regions should be structured to have a consistent digest.

   These controllers is less secure than HS above. If external flash is involved, flash writes need protection with an external controller that enforces HS compatibility on the GS level SoC. See section on Cerberus RoT.

3 High Security (HS) Processor Requirements and Boot Flow:

The following section describes the Processor requirements and boot flow for the HS classification Processor in Project Cerberus. The Processor ROM boot loader should support the following capabilities:

1. Support flash encryption and secure boot
2. One-Time Programmable (OTP) Memory for key storage in non-volatile memory.
    a. Minimum of 256-bit OTP memory for public key SHA2 hash.
    b. Minimum of 256-bit OTP memory for AES key.
    c. Keys must be inaccessible outside of secure memory.
3. Support minimum SHA2 2048 bit public key authentication
4. Support AES 256 bit encryption/decryption
5. Support additional OTP memory for revocation of key chain manifests or digests.
6. Support Device Identifier Composition Engine (DICE)
    a. Measurement of first mutable (BL after ROM).
    b. HW protected Unique Device Secret (UDS)
    c. Compound Device Identifier (CDI)
        i. HMAC of UDS and first mutable code (CDI)
7. Support FIPS 140-2 compliant tamper protection.
8. Support Power on Self-Test.
9. Source Code and utility for programming memory and sealing (blowing) e-fuses.
10. Source Code and utility for signing and encrypting firmware images.
11. ROM has native ability to calculate digest of key manifest, revoke key manifest and authenticate additional boot loaders against a key manifest.
12. ROM level asymmetric public key verification and symmetric crypto for firmware decryption

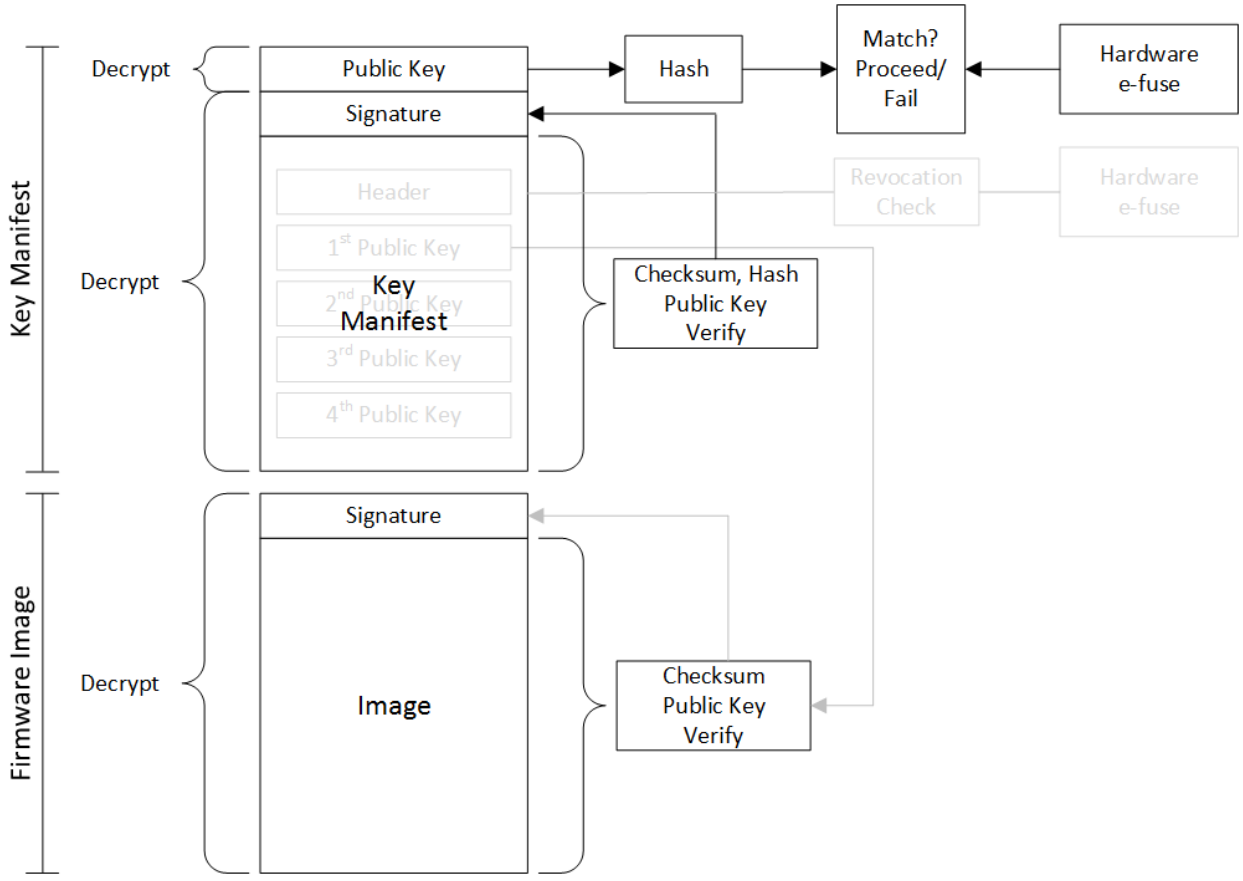
OTP memory consumes considerable silicon area. The HS Processor OTP memory area is required to have support for key revocation, typically a cryptographic hash public key manifest or numeric version in the manifests header that is compared against the revocation area in OTP memory.

The e-fused initial public key hash is used for the authentication of the key manifest and revocation of manifests. This public key or hash of the public key should not be accessible outside the secure bootloader environment. In the hashed initial public key mode, the key should not be revocable, if a digest of multiple public keys is stored in OTP, individual key digests represented by the OTP digest may be revocable.

3.1 Boot Process:

- ROM initialization code in secure execution mode
- Decompress first boot loader code (ROM)
- Decrypts second stage key manifest, authenticates with OTP key, moves flash into secure on-chip SRAM
- Verifies image digest signature with e-fused public key
- Verifies authorizes digest against revocation e-fuses.
- Extracts third-stage public key from key manifest
- Authenticates third-stage boot loader.
- Secure third-stage boot loader initialization from SRAM.
- Decrypts/Authenticates other firmware components using key manifest.

Figure 2 Boot Process



Note: The initial image is a key manifest of hashes of public keys uses to verify subsequent firmware images.

4 OTP Programming Process

- Microsoft RSA Public and AES key programmed into e-fuses.
- Keys are sealed/blown. [Process reviewed]
    - Component design should permit sealing e-fuses.
- There are two processes for fusing the key:
    - OTP Memory programmed at dock in Microsoft or Integration facility, [*] after firmware repaving has secured the platform.
    - OTP Memory programmed at manufacturer, whereby the part is fused with the Microsoft public key by the manufacturer. This creates a Microsoft specific part number for the device.

[*] Keys fused at Microsoft facility by provisioning software. Until Microsoft provisions the devices, they are untrusted. Negative and Positive testing is performed to verify the fusing process.

5 Signing Process

Microsoft uses RSA asymmetric keys (public/private) for signing and symmetric AES keys for encryption.

Multiple keys maybe required to support different boot stages device firmware. The third-stage boot loader, or "firmware block" should be protected by the public key in the key manifest (second-stage), subsequent keys can be stored in the firmware image, provided they are part of the key verified digest.

5.1 Manifest Signature Process:

The integrity of the key manifest is verified by the ROM boot loader using the public key fused in OTP memory. The manifest header contains version identification information which is verified against the revocation area of OTP memory.

- OTP memory is fused to a Microsoft public key.
- Key manifest is generated containing hashes of firmware authentication public keys and manifest identification.
- The keys contained in the manifest have designated purposes and unique Ids. The key id and firmware region/area are coupled in multi-stage firmware boot loaders.
- Manifest signature is generated with the private key corresponding to the public key in device OTP memory.
- The manifest is stored on the device internal memory and is verified by the ROM boot loader during boot.
- Upon manifest signature version, the manifest identification stored in the manifest header is compared against the revocation e-fuses. Revoked manifests fail integrity verification and prevent boot.

5.2 Image Signature Process:

- Microsoft reviews code, compiles code and generates a digest signature checksum of signed binary image using RSA private key for the corresponding image component.
- The public key for the firmware image, must match the key stored in the key manifest on the device. The key manifest is protected by the OTP memory e-fused key.
- The signature digest, public key and key identify are combined with image in a special image format.
- Entire firmware image, header and signature can be optionally encrypted with AES 256 bit symmetric key for additional obfuscation.
- Signed/Encrypted firmware copied to processor firmware load store through firmware update application interfaces.

Figure 3 Image Signature Preparation

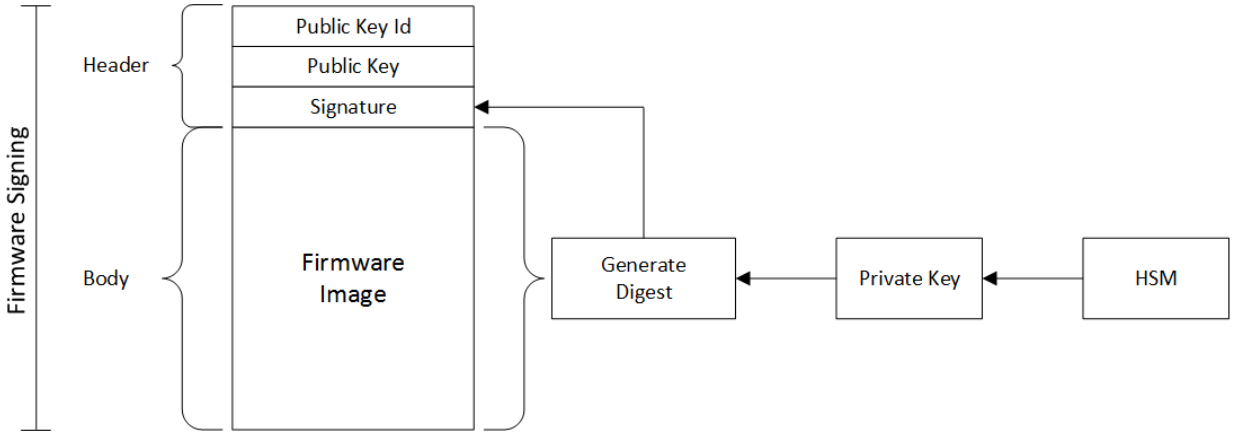


Figure 4 Encrypted Firmware Image

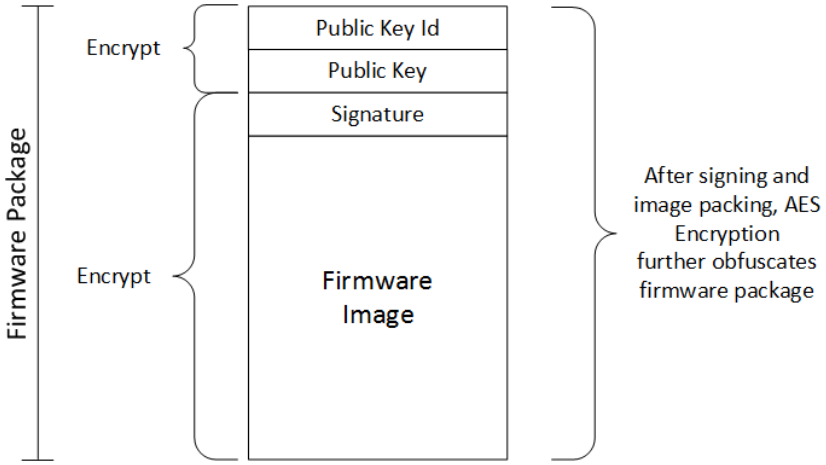
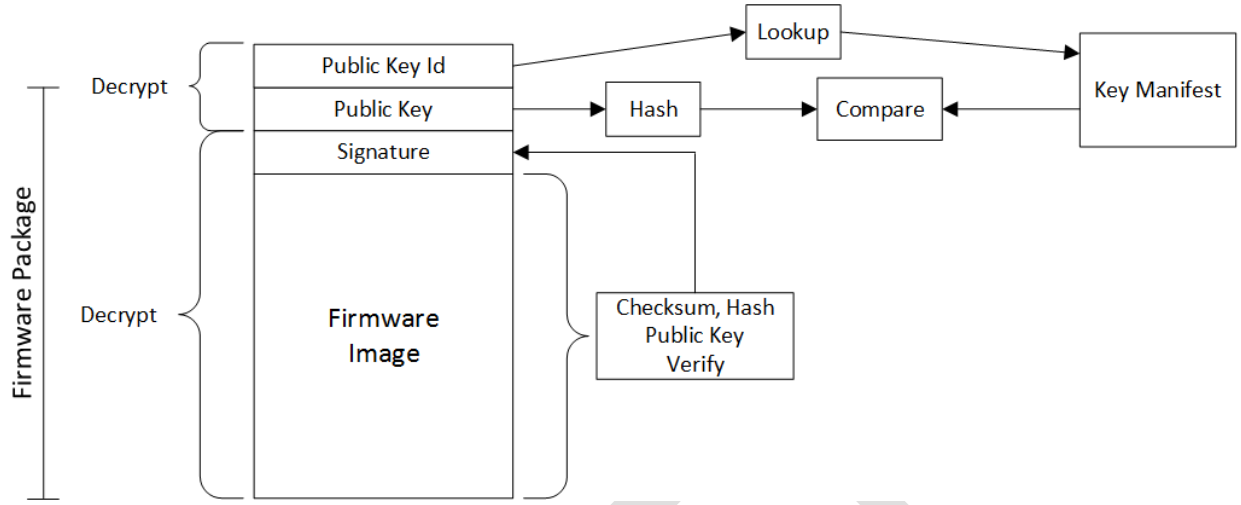


Figure 4 Secure Boot



5.3 Firmware Manifest Runtime Update Process:

The firmware manifest can optionally be updated as part of a firmware update, or independently updated as a separate firmware update. In both instances the upgrade process has similarity.

- An image containing the manifest with a header checksum is MD5 integrity verified by the update utility.
- The header includes the public key, image signature and firmware image identification metadata.
- Processor receives an update initiate request, if updates are permitted to the region (unlocked), bytes are received by the processor and held in the firmware staging area.
- Once the manifest header is received and AES decrypted, the image identification metadata are used to identify the type of firmware update and region offset.
- The manifest identification is extracted from the header and compared against the revocation list in OTP memory. If the manifest identification is older than the latest e-fused revoked id, the update is not permitted.
- If the manifest is not revoked, the public key in the header is hashed in secure SDRAM and compared against the hashed OTP key.
- Upon matching the Public key hash, a digest of the firmware image is generated.
- The firmware signature is decrypted with the public key, and the newly generated digest is then compared against the decrypted digest of the firmware image.
- Upon matching the digest, the firmware metadata informs the move from staging memory/flash to the active area on non-volatile storage.

Note: Manifest revocation requires that recovery areas in firmware updated with the unrevoked manifest. This is discussed in the revocation process.

5.4 Firmware Image Runtime Update Process:

- An image containing the firmware image with a header checksum is MD5 integrity verified by the update utility.
- Processor receives an update request, if updates are permitted to the region (unlocked), bytes are received by the processor and held in firmware staging area.
- Once the header is received and AES decrypted, the image identification metadata are used to identify the type of firmware update and region offset.
- The public key and key id are extracted from the header
- The key id must correspond to the firmware region being flashed. Multi-stage boot loaders have multiple keys in the manifest. The key Id must correspond to the firmware region.
- The Public key is then hashed and verified against the key manifest corresponding to the key Id.
- Upon matching the Public key hash, a digest of the remaining firmware image area is generated.
- The firmware signature is decrypted with the Public key, and the newly generated digest is then compared against the decrypted digest of the firmware image.
- Upon matching the digest, the firmware metadata informs the move from staging memory/flash to the active area on non-volatile storage.

Note: If the Processor supports only a signal stage boot loader, it is still recommended the firmware manifest for revocation.

6 Key Revocation:

The key manifest contains a numeric identification in its metadata that during boot is compared against dedicated OTP memory fuses for revocation. The public key that verifies the authenticity of the key manifest cannot itself be revoked, but authenticate manifests containing compromised keys can themselves be revoked based upon enforcement of the ROM boot loaders verification of the manifest identification against the OTP memory fuses reserved for revocation.

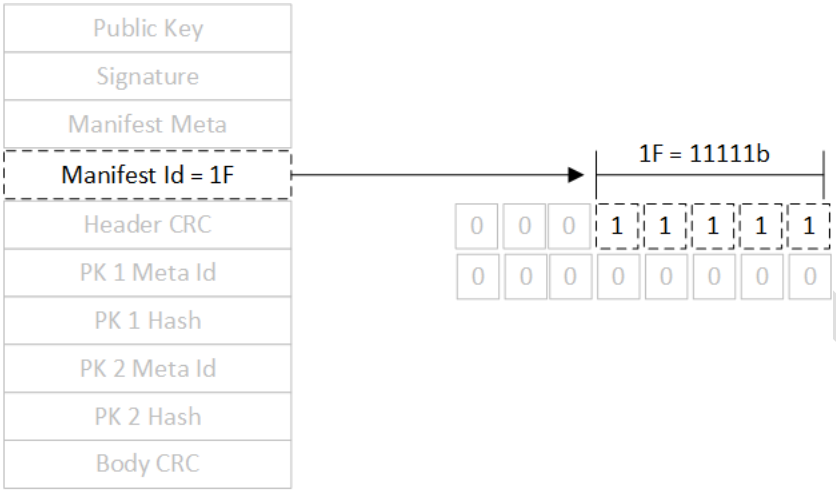
OTP memory consumes considerable silicon area and there is a limitation on the amount of revocations that can be securely fused into the Processor. When the revocation OTP area has become exhausted, it is required that the part permit the last fused manifest id.

This specification requires the ability to revoke key manifests a minimum of 8 times, including the final revocation. If using single byte of OTP memory, 8 revocations should be possible.

Key revocation is performed by setting the key manifest revocation bits in the metadata of key manifest header. Upon boot if the if the key manifest id is incrementally higher than the latest OTP memory fused area, then the next OTP memory area is programmed one increment higher. To load a key manifest on the Processor, the manifest must be digitally signed by the private key corresponding to the

fused pubic key. Revocation can only be achieved with a valid manifest id, one increment higher than the current manifest id, with renovation bits set in the header manifest.

Figure 5 Revocation Process



When the manifest Id is incremented and revocation bits set in a correctly signed manaifest, upon signature verification of the key manifest, the boot loader will compare the manifest id with the revocation OTP memory fuses. If the key manifest id is an increment higher than OTP memory, and OTP memory has not been exhausted, the new manifest id get's fused into OTP memory. If all reservation OTP memory has been fused no further revocation beyond the last valid (e-fused) are permitted. If the manifest id is valid, boot proceeds to verification of the third stage boot loader. If the manifest Id not the current permitted version, boot will halt.

7 General Security (GS) SOC Requirements and Boot Flow:

In Project Cerberus, Processors that do not enforce firmware authentication or have a ROM supporting Secure Boot, must implement the Cerberus RoT between the Processor and the Processors firmware load store. Processors that do not support secure boot with firmware integrity checks, cannot authenticate their first stage boot loader independently and therefore require auxiliary hardware to enforce this minimum level of firmware protection. These Processors should support the following capabilities in firmware:

1. A soft runtime Firmware update mode with public key digest verification.
2. Firmware structure that separates readonly and executable code regions of firmware from mutable regions for logs.
3. Source Code to utility for calculating digest and framing firmware images with signature.

In addition, the Cerberus RoT will sit between the Process and its firmware load store (typically NOR SPI flash) and enforce boot and runtime firmware authentication and signature integrity checks on behalf of the GS Processor. The Cerberus RoT will prevent runtime flash access to regions of firmware deemed locked for runtime access.