AN12601
On-The-Fly AES Decryption(OTFAD) in i.MX 7ULP

Rev. 0 — October 2019

Application Note

Contents

1 Introduction

1.1 Purpose

In embedded processor design, there is an increasing need for hardware support of cryptographic calculations required for system security. In particular, there are emerging customer requirements to protect application code and data stored in external memories in an encrypted form, and have the embedded processor provide hardware support for "on-the-fly" decryption. As the name suggests, the external memory image of the code and data is always in an encrypted format, and in response to processor references of the memory space, the memory image is decrypted on the fly, returning the original values to the requesting bus master.

In i.MX 7ULP, the module that provides the aforementioned capability is called OTFAD (On-The-Fly AES Decryption). The on-the-fly decryption engine is combined with the QuadSPI external flash memory controller. The QuadSPI external flash device provides a cost-effective non-volatile memory solution that supports eXecute-in-Place (XiP) capabilities. In other words, the combined access speed of the external flash memory, coupled with internal processor local cache memories, allows the application code to be executed directly from the external memory without the need to copy the code into another (faster) memory. For example, an attached SDRAM or on-chip SRAM.

This document gives an overview of the steps required in the preparation of an encrypted image that can be programmed in QSPI memory to boot the CM4 part using the OTFAD module present with the QSPI interface. For more details on the OTFAD module, refer to the i.MX 7ULP Security Reference Manual.

1.2 Audience

This document is intended for those who:

- Need to boot CM4 part of i.MX 7ULP with an encrypted image.
- Need an explanation about the procedure involved in encrypting the CM4 boot image.

It is assumed that the reader is familiar with the basics of the AES cryptographic algorithm.

1.3 Scope

This document provides an explanation of various methods in which a CM4 boot image can be encrypted. It requires usage of a few tools and a user-friendly interface to encrypt the boot image. The tools target the implementation of OTFAD module in i.MX 7ULP and therefore can only be used with this chip. M4 ROM supports encrypted boot on QSPI NOR flash using OTFAD hardware.

1.4 Definitions, Acronyms, and Abbreviations

Table 1. Cryptographic Terms

Term	Definition
Block Cipher	A family of functions and their inverse functions that is parameterized by cryptographic keys; the functions map bit strings of a fixed length to bit strings of the same length.
Block Size	The number of bits in an input (or output) block of the block cipher. For OTFAD, it is 128 bits (16 bytes)
Ciphertext	Encrypted data
CTR	Counter
Cryptographic Key	A parameter used in the block cipher algorithm that determines the forward cipher operation and the inverse cipher operation.
Data Block (Block)	A sequence of bits whose length is the block size of the block cipher.
Decryption (Deciphering)	The process of a confidentiality mode that transforms encrypted data into the original usable data.
ECB	Electronic Codebook
Encryption (Enciphering)	The process of a confidentiality mode that transforms usable data into an unreadable form.
Initialization Vector (IV)	A data block that some modes of operation require as an additional initial input.
Input Block	A data block that is an input to either the forward cipher function or the inverse cipher function of the block cipher algorithm.
Key Encryption Key (KEK)	Cryptographic key used to wrap and unwrap key data. Key Wrap Key Wrap constructions are a class of symmetric encryption algorithms designed to encapsulate (encrypt) cryptographic key material. The constructions are typically built from standard primitives such as block ciphers and cryptographic hash functions. (http://en.wikipedia.org/wiki/Key_Wrap)
Image Encryption Key (IEK)	Cryptographic key used to encrypt the boot image. There can be 4 separate image encryption keys for encrypting boot image in 4 different blocks.
Mode of Operation (Mode)	An algorithm for the cryptographic transformation of data that features a symmetric key block cipher algorithm
Nonce	A value that is used only once.
Output Block	A data block that is an output of either the forward cipher function or the inverse cipher function of the block cipher algorithm.
Plaintext	Usable data that is formatted as input to a mode.

Table continues on the next page...

Table 1. Cryptographic Terms (continued)

Term	Definition
RFC3394	The Advanced Encryption Standard (AES) Key Wrap Algorithm, defined to implement a key wrapping and unwrapping algorithm.

1.5 References

These documents provide specific references to the cryptographic functions supported by the OTFAD:

1. FIPS Publication 197, "Advanced Encryption Standard (AES)", U.S. Department of Commerce, National Institute of Standards and Technology (NIST), November 26, 2001.

2. NIST Special Publication 800-38A, "Recommendation for Block Cipher Modes of Operation: Methods and Techniques", U.S. Department of Commerce, National Institute of Standards and Technology (NIST), December 2001.

3. http://www.ietf.org/rfc/rfc3394.txt, "Advanced Encryption Standard (AES) Key Wrap Algorithm (RFC3394)", J. Schaad, R. Housley, September 2002.

4. i.MX 7ULP Security Reference Manual (Available at www.nxp.com)

2 Overview

2.1 OTFAD operation

The On-The-Fly AES Decryption (OTFAD) module provides an advanced hardware implementation that minimizes any incremental cycles of latency introduced by the decryption in the overall external memory access time. It implements a block cipher mode of operation supporting the counter mode (CTR). The CTR mode provides a confidentiality mode that features the application of the forward cipher to a set of input blocks, called counters, to produce a sequence of output blocks that are exclusive-ORed with the plaintext to produce the ciphertext, and vice versa.

The OTFAD engine includes complete hardware support for a standard AES key unwrap mechanism to decrypt a key BLOB data instruction containing the parameters needed for up to 4 unique AES contexts. Each context has a unique 128-bit key, 64-bit counter and 64-bit memory region descriptor.

The OTFAD supports three modes of operation as defined by input configuration and control signals:

1. Logically Disabled Mode (LDM)
2. Security Violation Mode (SVM)

   In both LDM and SVM mode, the OTFAD registers are inaccessible and the input data from the QuadSPI is passed through the AHB RAM buffer and no data decryption is performed.

3. Normal Mode (NM)

In Normal mode, the OTFAD can perform data decryptions and/or data bypasses as it responds strictly to the memory transactions on its connected system buses.

In this document, we study the functionality of OTFAD engine in Normal mode. While in this mode, OTFAD engine can perform key unwrapping and data decryptions allowing the encrypted image to boot. OTFAD engine can essentially perform three operations in normal mode:

1. **Key scrambling**: This mechanism allows to scramble the OTFAD key (KEK) responsible to unwrap the image encryption keys (IEK).
2. **Key Unwrapping**: The OTFAD key (KEK) is used to unwrap the key blobs to extract the IEKs of each context.
3. **Data decryption**: The image encryption keys are used by OTFAD engine to decrypt the encrypted image blocks of maximum 4 contexts.

This document guides through the tools that have been created to help with the operation of key scrambling, key wrapping and data encryption which could then be used to create an encrypted boot image. The OTFAD module takes the encrypted image and performs key scrambling, key-unwrapping, and data decryption operations to boot the encrypted image securely.

2.1.1 Key Scrambling

Key scrambling is an encryption algorithm designed to obfuscate the secret key information. It is an optional feature. When enabled, the OTFAD engine implements a key scrambling algorithm, wherein, the input OTFAD key is scrambled and then utilized to unwrap the key blobs containing the image encryption keys. The OTFAD tool takes three inputs to scramble the OTFAD key:

1. **OTFAD key**: An OEM programmed input 128-bit key used to wrap/un-wrap the image encryption keys.
2. **Key Scramble**: An OEM programmed input 32-bit key used to scramble the input OTFAD key.
3. **Key Scramble Align**: An OEM programmed 8-bit key aligns value used along with Key Scramble in scrambling algorithm.

Once these values are provided to the OTFAD tool, it then scrambles the input OTFAD key (according to the key scrambling algorithm, refer to i.MX 7ULP SRM) and results in a scrambled OTFAD key (KEK). This scrambled OTFAD key is then used by the OTFAD tool to wrap the IEKs of each context into a key blob. In the case when key scrambling is disabled, the OTFAD key is directly used to create the key blobs of IEKs. During boot, the OTFAD engine in i.MX 7ULP CM4 core then scrambles the fused OTFAD key internally using the same key scrambling algorithm implemented in hardware and unwraps the key blobs to extract the IEKs of each context. Pictorial representation in Figure 3.

2.1.2 Key wrap/unwrap

Key blob processing is a mechanism where a preloaded data structure wrapped using the RFC3394 standard[3]. It is fetched from the external flash memory after a system reset event, unwrapped by the OTFAD hardware, and automatically loaded into the four memory context programming model registers. In this manner, the sensitive context data, for example, the 128-bit key and 64 bits of the counter, is unwrapped and handled entirely within the OTFAD.

The OTFAD tool uses AES key wrap algorithm (RFC3394) to wrap the secrets with OTFAD key. The secret contains the Image encryption key, Counter, start and end address of the image being encrypted and CRC32 of the previous information. The IV used is constant ```0xa6a6a6a6_a6a6a6a6``` as per RFC3394. As OTFAD engine implements 4 context, there could exist 4 different Image encryption keys and corresponding input data. The output is 48 bytes of keyblob generated by the tool.

NOTE

As the OTFAD engine supports 4 AES contexts, it requires all memory contexts to exist even in the case when only 1 context is enabled. Thus, the key blobs must be created for all 4 contexts (dummy blobs for disabled contexts).

2.1.3 Image encryption/decryption

The OTFAD tool uses AES-128-CTR encryption algorithm to encrypt the boot image based on the input parameters provided. As OTFAD supports 4 contexts, the input image can be divided up to 4 different images which can be encrypted using their own image encryption key (IEK) and counter values.

The boot image for QSPI boot medium in MX 7ULP CM4 starts at an offset 0x1000, therefore the boot image can only be encrypted from 0x1000 offset.

A typical input image looks as below (with offsets):

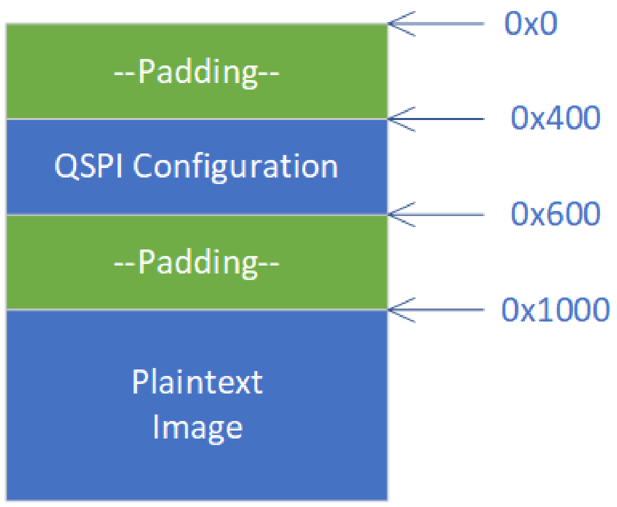

Figure 1. Input image

The output looks as below (with offsets) when complete image is encrypted with one IEK:

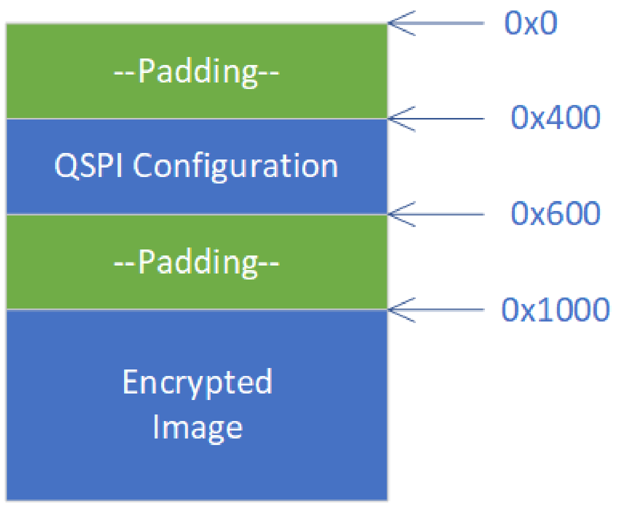

Figure 2. Output encrypted image

The OTFAD operation can be depicted as Figure 3:

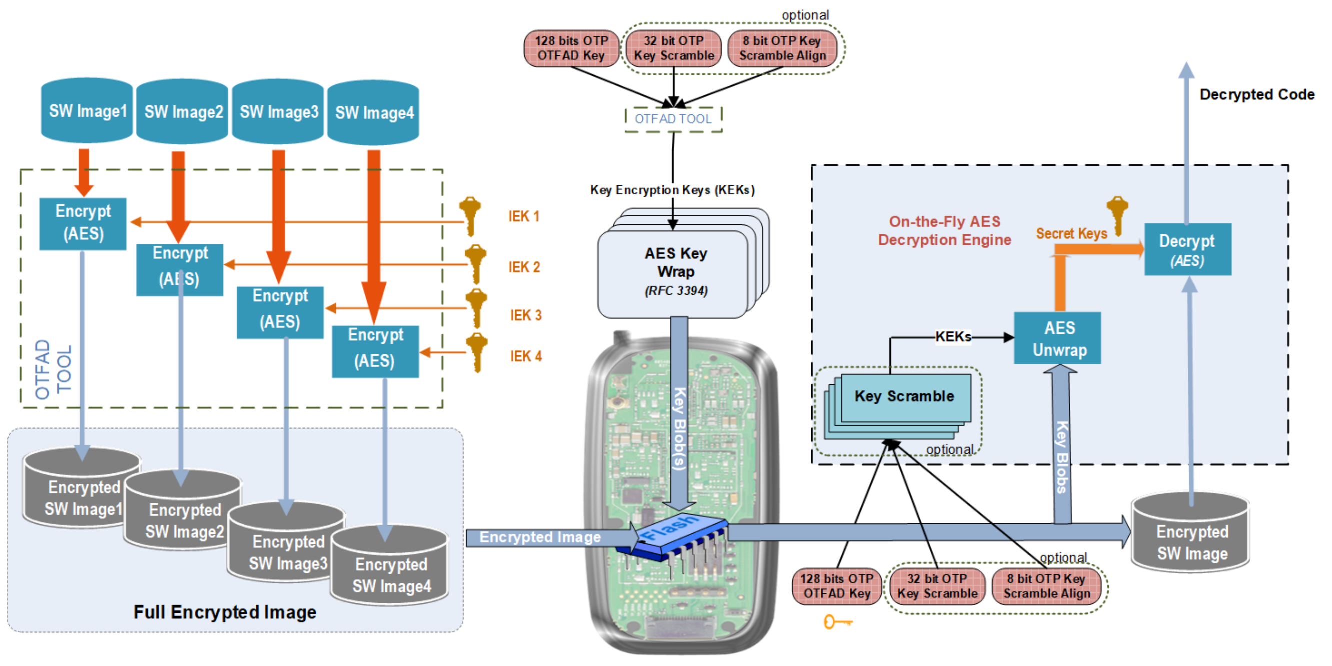

Figure 3. OTFAD operation

2.2 CM4 Boot ROM

During initial boot up of the i.MX 7ULP chip, it checks the eFuses for enablement of OTFAD. If the OTFAD is enabled, the CM4 BootROM configures OTFAD engine's control register (CR) and enables the following:

1. Key blob processing is enabled (KBPE) and key blob processing is initiated (SKBP)
2. Decryption is enabled in OTFAD engine, which fetches data as defined by the hardware configuration (GE)

If key scrambling is enabled through eFuses, the field CR[KBSE] is enabled accordingly. More information on these register fields can be found in i.MX 7ULP SRM document.

3 OTFAD guide

3.1 Enablement

The On-The-Fly-AES-Decryption (OTFAD) engine present in CM4 of i.MX 7ULP can be enabled using eFuses or the register interface. Some data required by the OTFAD engine to perform decryption of the encrypted boot image is programmed in eFuses as follows:

Table 2. OTFAD eFuses

eFuse	Bits	Description
OTFAD_KEY	128	Key used for key blob unwrap
KEY_BLOB_EN	1	Enable key blob unwrap
KEY_SCRAMBLE_EN	1	Enable Key Scramble
KEYn_SCRAMBLE[1:0]	2	Key scramble align select for key blob unwrap (n=0..3)

Table continues on the next page...

Table 2. OTFAD eFuses (continued)

OTFAD_KEY_SCRAMBLE[31:0]	32	Key to scramble OTFAD_KEY

3.2 Implementation

There are multiple tools available to instrument the encrypted image to boot the CM4 part. As mentioned in Section 2.1, the OTFAD engine performs 3 steps to boot an encrypted image. This encrypted image is generated using the same method but in reverse, on a host machine, as follows:

1. **Key Scramble**: If enabled, Input OTFAD key is scrambled using input key scramble and input key scramble align values, generating a Key Encryption Key (KEK).
2. **Key Wrap**: KEK is used to IEKs using RFC3394 algorithm.
3. **Image encryption**: IEKs are used to encrypt the boot image per context.

In addition to these tools, there exists a python script which is used to build the final encrypted image using the inputs provided in a YAML configuration file. OTFAD key, Key Scramble and Key scramble align values in the configuration file are also programmed in the chip for OTFAD engine to process the encrypted image.

3.2.1 Input configuration file

The Input configuration file consists of necessary information required to build an encrypted boot image. An example of Input configuration file is as follows:

```
# Global I/O
otfad_key: "otfad_key"
key_scramble: "key_scramble"
key_scramble_align: 0x11
input_image: "ulp_m4.bin"
output_file: "otfad.bin"
# Boot image partitions I/O
boot_image_part1:
image_offset: 0x1000
size: 0x2000
image_enc_key: "enc_key1"
counter: "ctr1"
boot_image_part2:
image_offset: 0x3000
size: 0x2000
image_enc_key: "enc_key2"
counter: "ctr2"
boot_image_part3:
image_offset: 0x5000
size: 0x2000
image_enc_key: "enc_key3"
counter: "ctr3"
boot_image_part4:
image_offset: 0x7000
size: 0x2000
image_enc_key: "enc_key4"
counter: "ctr4"
```

The input image is the image built by CM4 SDK. Other inputs consist of OTFAD key, Key scramble, Key scramble align, IEK and Counter. The input image can be encrypted in 4 contexts, each with its own IEK and counter. Each of those contexts can have their own image offset and size, but the memory locations cannot overlap.

3.2.2 Build script

The encrypted image can be built using the build_otfad_enc_image.py script. The script takes in the Input configuration file and executes the following 3 tools in the OTFAD tool package.

1. **Key scrambler:**

   Key scrambling tool takes the OTFAD key, Key scrambler and Key scramble align from the input configuration file and creates a scrambled OTFAD key (KEK).

2. **Key wrap:**

   Key wrapping tool takes the KEK generated by Key Scrambler tool and the inputs from each boot_image_part from the Input configuration file to generate key blobs for each context.

3. **Encrypt Image:**

   The encrypt image tool takes the Input image, and contents in each boot_image_part of the Input configuration file to encrypt the input boot image.

NOTE

Each tool can be run on its own as well. For more information on how to use the tools can be referred to in the individual tool's README files.

An example of final encrypted image (according to the example Input configuration file above) looks like follows:

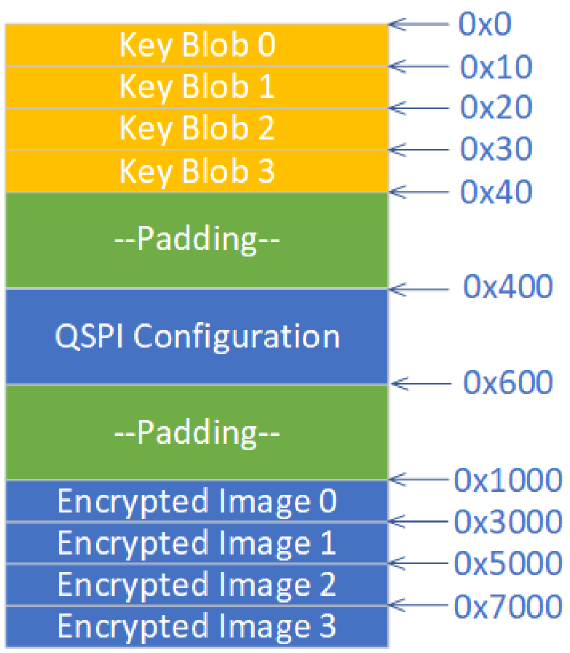

Figure 4. Final encrypted image

4 Final stage

Once the final encrypted image is generated, the image can be then programmed in the QSPI boot medium. Once tested, the OTFAD configuration is recommended to be locked down by setting following fuses:

1. OTFAG_CFG_LOCK – Locks following (OTFAD related) configurations:
   a. KEY_SCRAMBLE_EN
   b. KEY_BLOB_EN

   c. KEYn_SCRAMBLE
   d. RESTRICT_OTFAD_IPS
   e. OTFAD_KEY_SCRAMBLE

2. OTFAD_KEY_LOCK – Lock for OTFAD KEY fuses

Please see the FuseMap document to get full picture of the fuses involved.

5 Revision history

Table 3. Revision history

Revision number	Date	Substantive changes
0	10/2019	Initial release

6 Appendix A. known issues

A. 1. QSPI access with OTFAD

- After enabling OTFAD, it is possible to encounter an issue with probing a QSPI boot media to program a new image. The workaround is to disable OTFAD before probing the QSPI.

#write FLDM bit to enable access to QSPI

mw 0x410A5C00 0xc0000068

Date of release: October 2019
Document identifier: AN12601