AN13711

i.MX RT1170 Keyblob Encapsulation and Decapsulation

Rev. 0 — 4 August 2022 Application note

Document information

Information	Content
Keywords	RT1170, Keyblob encapsulation and decapsulation, caam
Abstract	This AN is for RT1170 to use caam built-in blob encapsulation and decapsulation.

1 Introduction

1.1 Purpose

Protecting sensitive data on the device is a critical requirement. This goal can be achieved by storing the data in an encrypted form, Cryptographic Acceleration and Assurance Module (CAAM) uses blobs, which are cryptographic data structures, in order to protect data. It provides both confidentiality and integrity protection.

1.2 Audience and scope

This document is for i.MX RT1170 users who want to understand:

• CAAM's built-in blob protocol for protecting user-defined data across system power cycles

• How to encapsulate and decapsulate red or black key blobs in RAM

• How to encapsulate and decapsulate red or black key blobs in secure RAM

The reader must be familiar with the basics of the Advanced Encryption Standard (AES) cryptographic algorithm.

1.3 Acronyms and abbreviations

Table 1. Acronyms and abbreviations

Acronym	Definition
AES-ECB	AES Electronic Codebook Mode
AES-CCM	the Counter with Cipher Block Chaining-Message Authentication Code Mode
BKEK	Blob-Key Encryption Key
BK	Blob Key
Blob	Data structure used by CAAM to protect data
CAAM	Cryptographic Acceleration and Assurance Module
MAC	Message Authentication Code
JDKEK	Job Descriptor Key Encryption Key
TDKEK	Trusted Descriptor Key Encryption Key
RAM	Random Access Memory
RNG	Random Number Generator
KDF	Key-Derivation Function
OEM	Original Equipment Manufacturer - an organization that makes devices from component parts bought from other organizations

2 Overview

CAAM can protect data in a cryptographic data structure, which provides both confidentiality and integrity protection. The CAAM's built-in blob protocol provides a method for protecting user-defined data across system power cycles. The data to be protected is encrypted so that it can be safely placed into non-volatile storage before the chip is powered down. Each time that the blob protocol is used to protect data, a different, randomly generated key is used to encrypt the data. This random key is itself encrypted using a Key-Encryption Key (BKEK), and the resulting encrypted key is then stored along with the encrypted data. The BKEK is derived from the chip's master secret key, so the BKEK can be recreated when the chip powers are up again. The combination of an encrypted key and encrypted data is called a blob.

2.1 Blob conformance considerations

In the context of CAAM, a blob is encrypted data that is bound to a specific device by virtue of using a secret non-volatile, device-specific master key. This master key is used only for creating and extracting blob data, and the value of this key itself cannot be extracted from a device. To protect data that requires a high level of security, blob creation is performed in hardware using 256-bit security strength. AES-256 is used as the encryption algorithm and SHA-256 is used for key derivation.

CAAM blobs provide both confidentiality and integrity protection for the encapsulated data. Because a blob protects both confidentiality and integrity, it may be stored in external long-term storage such as flash. Counter with cipher block chaining-message authentication code (AES-CCM) is used as the bulk encryption algorithm.

Note: *The MAC associated with a blob provides integrity protection not only for the encrypted data that the blob contains, but also for all intermediate keys used in the creation of a blob.*

Many different blobs can exist at the same time, used for different purposes and subject to different security policies. To guarantee that blobs are not inadvertently or intentionally swapped, CAAM encrypts different blobs with different keys. Two mechanisms are used to guarantee that a single key is not used to encrypt unrelated data and to ensure that each key is used to encrypt as little data as possible. One of these mechanisms is random key generation. Each time that a blob is created, CAAM generates a different, random 256-bit key using CAAM's internal hardware Random Number Generator (RNG). This blob key is used to encrypt the blob data using AES-CCM, which provides both confidentiality and integrity protection. The second mechanism is key derivation using a device-unique non-volatile master key as the key-derivation key. The volatile random blob key is encrypted with the non-volatile key derived from the master key and then stored with the blob so that the blob data can be decrypted during subsequent power-on cycles. Different types of blobs are encrypted using different keys derived from the master key. The derived keys are further differentiated by a key modifier supplied by software that can be used to guarantee that one blob cannot be inadvertently or maliciously substituted for another blob. Software can use these key modifiers to differentiate specific data, or to prevent replay attacks (replacing the current blob with an out-of-date version of the blob).

3 Blob types

CAAM supports different types of blobs, and a coded value of the blob type is used as an input to the Key Derivation Function (KDF). It prevents a blob that was exported as one type from being imported as another type as it would decrypt improperly and fail the MAC tag check. The Table 2 below lists the types of blobs that CAAM supports. The type categories are orthogonal, that is, a blob has one type from each type category. For instance, one blob may be a normal format / black key / general memory blob, while another blob may be a test format / general data / Secure Memory blob. In addition, black key blobs are differentiated by encryption mode and encryption key, so one black key blob may be an AES-ECB / TDKEK type and another black key blob may be an AES-CCM / JDKEK type.

Table 2. Blob types

Type Category	Type
Formats	Normal format
	Test format
	Master key verification format
Contents	General data (that is, red blobs)
	Black keys (that is, black blobs) • Encryption modes: AES-ECB and AES-CCM • Encryption keys: JDKEK and TDKEK
Memory types	General memory
	Secure memory

3.1 Blob types differentiated by format

CAAM supports three different formats for blobs, usable for all blob content types, all blob memory types, and all blob security state types. The figure below describes the blob formats and how they work.

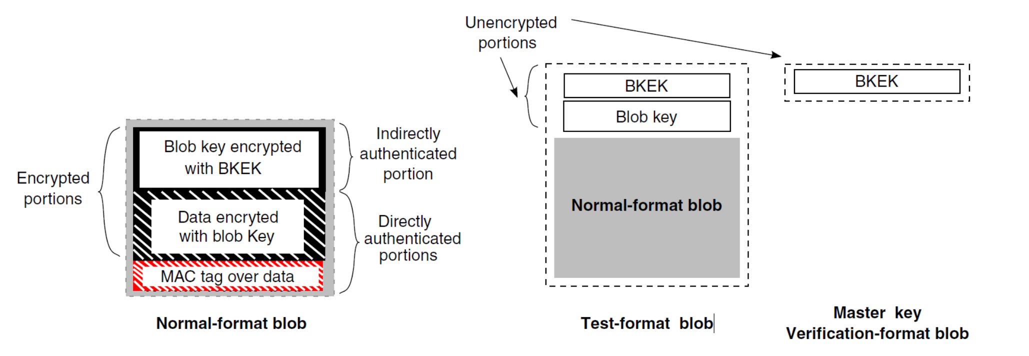

Figure 1. Formats of CAAM blobs

A normal-format blob consists of the encrypted blob key, the encrypted data, and a message authentication code (MAC) tag, as shown on the left side of the figure. A randomly generated, 256-bit blob key is used to encrypt the data using the AES-CCM cryptographic algorithm. AES-CCM encrypts the data and also yields a MAC tag that is used to protect the data's integrity. The blob key itself is encrypted in AES-ECB mode using a 256-bit BKEK. Checking the MAC directly authenticates the data encapsulated in the blob. The blob key is indirectly authenticated because substitution or corruption of the encrypted blob key yields an incorrect plaintext blob key. It causes the blob content to be decrypted incorrectly, which is detected by the MAC check. As a normal-format blob is used to protect actual data, BKEK that is used to encrypt the blob key for a normal format blob is secret, having been derived from the secret master key.

As shown in the middle of the figure, a test-format blob consists of a normal-format blob, with the unencrypted BKEK and unencrypted blob key prepended. As the purpose of a test-format blob is to facilitate testing blob encapsulation and decapsulation, BKEK for a test-format blob is derived from a known test key. CAAM permits test-format blobs to be encapsulated or decapsulated only when CAAM is in non-secure mode.

As shown on the right side of the figure, a master key verification format blob consists of only the unencrypted BKEK. As the purpose of a master key verification format blob is to verify that the master key has been properly programmed, BKEK for a master key verification format blob is derived from the secret master key. To ensure the secrecy of BKEKs used for normal format blobs, the derivation is different from the derivation used for normal format blobs. It ensures that BKEKs used to protect data cannot be exposed by examining the BKEK values in master key verification format blobs.

Note: *The test format blob and the master key verification format blob are not supported in i.MX RT1170 SDK example, in order to keep OEM’s secret data safe.*

3.2 Blob types differentiated by content

Unencrypted data that should be protected is sometimes referred to as "red data". Thus, the type of blob intended for general data (which is left unencrypted when the blob is decapsulated) is called a red blob. When CAAM is instructed to encapsulate data as a red blob, it assumes that the data to be encapsulated is unencrypted, and proceeds to encrypt the data with the blob key. Likewise, when CAAM is instructed to decapsulate a red blob, it assumes that the data that is decapsulated is to be left in memory unencrypted. Other mechanisms, such as an operating system or hypervisor acting with a memory management unit, may be used to protect the data before it is encapsulated into a blob and after it is decapsulated from a blob. CAAM's secure RAM can also be used to protect unencapsulated data.

CAAM's black blob mechanism is a means for translating between black key encapsulation and blob encapsulation without exposing the key during the translation process. A black blob is simply a blob whose input during blob encapsulation is assumed to be a black key. Its output during blob decapsulation is either a black key that is written into memory or an unencrypted key that is placed directly into a Key Register. CAAM supports the protection of cryptographic session keys by encrypting these keys in a "black key" encapsulation format when storing them in memory via a FIFO STORE command and then decapsulating them "on-the-fly" as they are referenced by a Job Descriptor with a descriptor KEY command. Black key encapsulation or decapsulation is very quick, but black keys are intended only for protection during the current SoC power-on session. Black keys encapsulated during one chip power-on session cannot be decapsulated on subsequent power-on sessions. It happens because the key-encryption key (JDKEK or TDKEK) is erased during power-down and is replaced by a new randomly generated key encryption key at power-up. To protect a key so that it can be recovered on subsequent power cycles, the key must be encapsulated as a blob. A key could be encapsulated as a red blob, but it would require exposing the key in memory in unencrypted form. To avoid exposing keys in unencrypted form, CAAM supports the concept of black blobs. (Data that is not sensitive to disclosure, either because it is inherently non-sensitive or because it always remains encrypted, is sometimes referred to as "black data".)

3.3 Blob types differentiated by memory

CAAM supports blobs for use with different types of memory:

- General memory blobs contain data that originated from any memory accessible to CAAM.
- Secure memory blobs contain data that originated from CAAM's Secure Memory.

All of the blob-format types and blob-content types are available for use with either general memory blobs or Secure Memory blobs. The input data for Secure Memory blob encapsulation must all come from a single Secure Memory partition, and a Secure Memory blob can be decapsulated only to a single Secure Memory partition. If not, the encapsulation or decapsulation process is terminated with an error indication before reading or writing the second partition. CAAM immediately aborts any attempt to decapsulate a secure memory blob into memory other than CAAM Secure Memory, because it would bypass the access controls implemented by Secure Memory.

Another important way that Secure Memory blobs differ from general memory blobs is in the derivation of the blob-key encryption key (BKEK). The BKEK for general memory blobs is derived from the master key or the test key, in non-secure mode, the 128-bit key modifier value within the blob descriptor, and a blob type identifier that includes a constant specific to general memory blobs. The BKEK for Secure Memory blobs is also derived from the master key or the test key, in non-secure mode, but uses a 64-bit key modifier value within the blob descriptor, and a blob type identifier that includes a constant specific to Secure Memory blobs, and the values in the PSDID, SMAPJR, and SMAG2/1JR registers of the partition that the blob is being exported from, or imported to. The reason that the PSDID, SMAPJR, and SMAGR contents are included in the BKEK derivation for Secure Memory blobs is to bind cryptographically the access permissions to the blob. It ensures that a Secure Memory blob can be imported only into a partition with the same access permission settings (and Security Domain ownership) as the partition from which the blob was exported.

Note: *For more information about Secure Memory, see i.MX RT1170 SRM, chapter 6.11.8.*

4 Blob encapsulation

A data blob is encrypted using a blob key (BK), which is a random number used as an AES-CCM key. The NIST AES-CCM specification states that for any key, all invocations must use distinct nonces and counter blocks. Although CAAM uses the same nonce and initial counter block values for all data blobs, CAAM satisfies the AES-CCM requirement because each encryption operation uses a different key (that is, a random number generated by the RNG). The nonce is given as all zeros, and so the initial block B0 = 3B00_0000_0000_0000_0000_0000_0000_xxxxh, where xxxx is the number of bytes of plaintext (maximum length is 65535 bytes), while the initial counter value Ctr0=0300_0000_0000_0000_0000_ 0000_0000_0000h. These values are automatically generated during the encapsulation operation.

Figure 2 shows the entire blob-encryption operation. B0 is generated internally and stored in the Class 1 Context DWords 0 and 1, while Ctr0, also generated internally, is stored in Class 1 Context DWords 2 and 3. The random BK value is stored in the Class 1 Key Register, and the operation mode is set to AES-CCM.

If the plaintext data is in Secure Memory and the data is exported as a Secure Memory blob, then all data must be from the same partition. At the blob pointer, the first 32 bytes contain the key blob, which is the encrypted value of the random blob key. Output ciphertext data (data blob) is stored at the blob pointer + 32. The generated message authentication code (MAC, the signature over the data blob) is stored in the final 16 bytes of the blob.

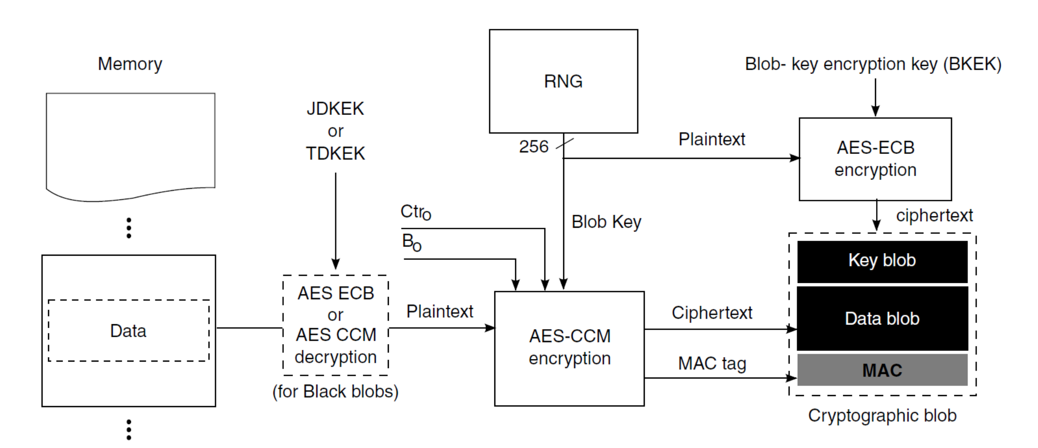

Figure 2. Encapsulating a blob

5 Blob decapsulation

Before decrypting a data blob, the associated key blob must be decrypted to obtain the blob key. The key blob resides at the blob pointer. AES-ECB mode is used to decrypt the key blob using the BKEK. Generation of the BKEK for general memory blobs and Secure Memory blobs is described below.

Ctr0 and B0 are generated internally, and are stored in the Class 1 Context 1 and Context 2 registers, respectively (see Figure 3). AES-CCM mode is used to decrypt the data blob (starting at the blob pointer + 32), using the decrypted blob key. If the decrypted data is from a Secure Memory blob, then the decrypted data must be written into Secure Memory and all of the data must be written into the same partition. If any of the pages are not within the same partition of Secure Memory, the blob decryption process is terminated with an error, but any data written prior to the error overwrites the previous data within the partition.

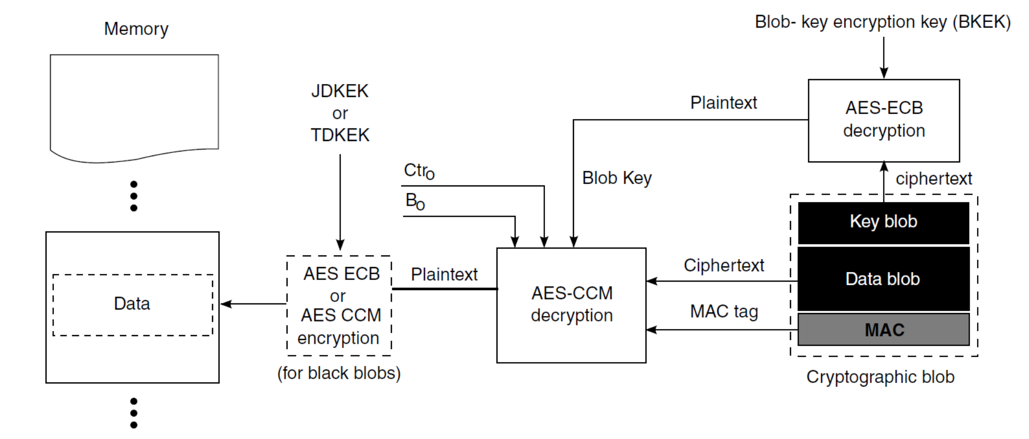

Figure 3. Decapsulating a blob

6 Using Keyblob function

This section describes how to use the blob encapsulation and decapsulation function in a different mode.

When using the attached CAAM driver example, the default settings of the blob example are to generate a red blob and a black blob in AES-ECB mode that are placed in processor's RAM. The 256-bit or 128-bit OEM secret key and key modifier needed for master key derivation are generated by RNG. Figure 4 shows the logs of generating a red blob and a black blob in default settings.

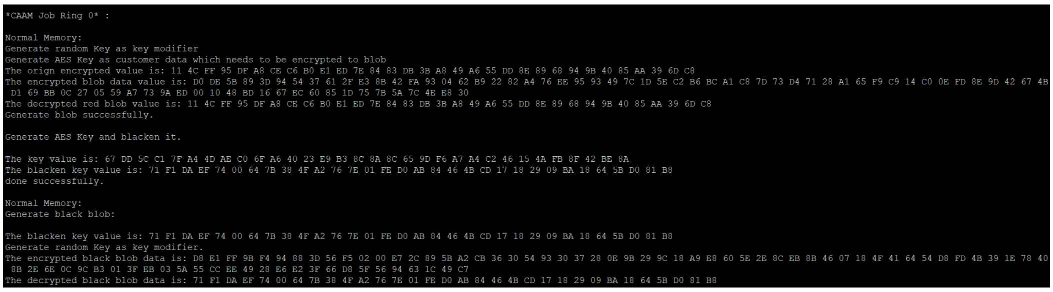

Figure 4. Logs of generating a red blob and a black blob in default settings

OEMs can generate a red blob and a black blob in secure memory by adding the definition of "SECUREMEMORY". The blob input and output buffer are placed in CAAM secure RAM. For secure memory settings, see i.MX RT1170 SRM, chapter 6.11.8.

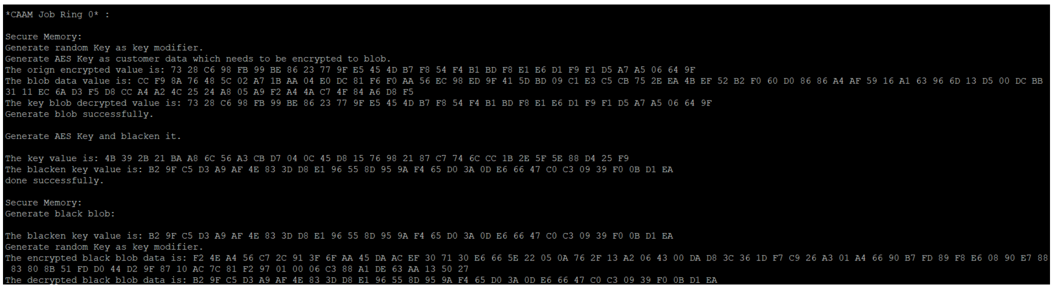

Figure 5. Logs of generating a red blob and a black blob in secure memory

The attached example also supports the AES-CCM mode for black blob generation. The major change is the key length of the blob input. Meanwhile, the job descriptor should change to the AES-CCM mode.

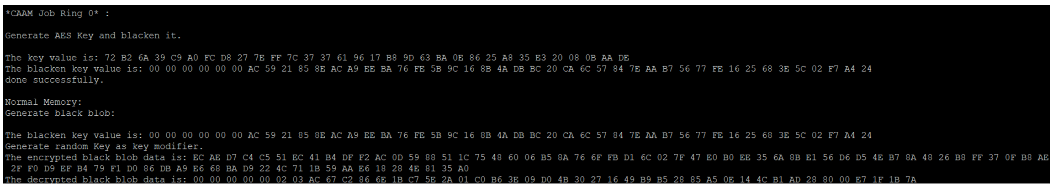

Figure 6. Logs of generating a black blob in normal memory with AES-CCM

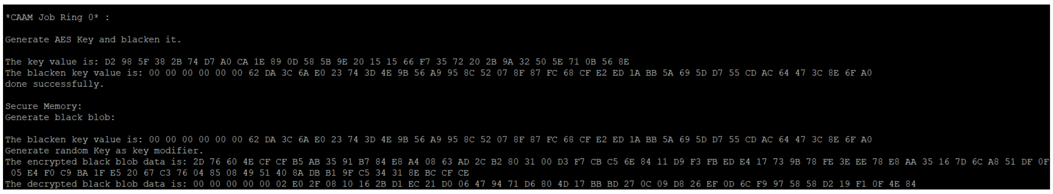

Figure 7. Logs of generating a black blob in secure memory with AES-CCM

7 Software enablement

The keyblob encapsulation and decapsulation functionality can be enabled using the attachment that contains `fsl_caam.c`, `fsl_caam.h`, and `caam.c`. For more information, see the latest AN release available on www.nxp.com.

8 References

The following documents may offer further reference.

- Demo Application to Generate Red/Black Blobs Using CAAM and Encrypt/Decrypt Data (document AN12554)
- Security Reference Manual for the i.MX RT1170 Processor (document i.MX RT1170 SRM)

9 Revision history

The following table summarizes the changes since the initial release.

Revision history

Revision number	Date	Substantive changes
0	04 August 2022	Initial release

10 Legal information

10.1 Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

10.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this data sheet expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

10.3 Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Contents

1 Introduction 2
1.1 Purpose 2
1.2 Audience and scope 2
1.3 Acronyms and abbreviations 2
2 Overview 2
2.1 Blob conformance considerations 3
3 Blob types 3
3.1 Blob types differentiated by format 4
3.2 Blob types differentiated by content 5
3.3 Blob types differentiated by memory 5
4 Blob encapsulation 6
5 Blob decapsulation 7
6 Using Keyblob function 8
7 Software enablement 9
8 References 9
9 Revision history 9
10 Legal information 10

Date of release: 4 August 2022
Document identifier: AN13711