

Project Zipline Keyblob

Micro Architecture Specification

**Authors:**

**Microsoft Corporation**

**Broadcom Corporation**

**Revision History**

|  |  |
| --- | --- |
| Date | Description |
| 05/01/2019 | Version 1.0 |
|  |  |

**License**

Contributions to this Specification are made under the terms and conditions set forth in the Open Web Foundation Contributor License Agreement (“OWF CLA 1.0”) (“Contribution License”) by:

Microsoft Corporation

Broadcom Corporation

Usage of this Specification is governed by the terms and conditions set forth in Open Web Foundation Final Specification Agreement (“OWFa 1.0”) (“Specification License”).

**Note**: The following clarifications, which distinguish technology licensed in the Contribution License and/or Specification License from those technologies merely referenced (but not licensed), were accepted by the Incubation Committee of the OCP:

**None.**

NOTWITHSTANDING THE FOREGOING LICENSES, THIS SPECIFICATION IS PROVIDED BY OCP "AS IS" AND OCP EXPRESSLY DISCLAIMS ANY WARRANTIES (EXPRESS, IMPLIED, OR OTHERWISE), INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR A PARTICULAR PURPOSE, OR TITLE, RELATED TO THE SPECIFICATION. NOTICE IS HEREBY GIVEN, THAT OTHER RIGHTS NOT GRANTED AS SET FORTH ABOVE, INCLUDING WITHOUT LIMITATION, RIGHTS OF THIRD PARTIES WHO DID NOT EXECUTE THE ABOVE LICENSES, MAY BE IMPLICATED BY THE IMPLEMENTATION OF OR COMPLIANCE WITH THIS SPECIFICATION. OCP IS NOT RESPONSIBLE FOR IDENTIFYING RIGHTS FOR WHICH A LICENSE MAY BE REQUIRED IN ORDER TO IMPLEMENT THIS SPECIFICATION. THE ENTIRE RISK AS TO IMPLEMENTING OR OTHERWISE USING THE SPECIFICATION IS ASSUMED BY YOU. IN NO EVENT WILL OCP BE LIABLE TO YOU FOR ANY MONETARY DAMAGES WITH RESPECT TO ANY CLAIMS RELATED TO, OR ARISING OUT OF YOUR USE OF THIS SPECIFICATION, INCLUDING BUT NOT LIMITED TO ANY LIABILITY FOR LOST PROFITS OR ANY CONSEQUENTIAL, INCIDENTAL, INDIRECT, SPECIAL OR PUNITIVE DAMAGES OF ANY CHARACTER FROM ANY CAUSES OF ACTION OF ANY KIND WITH RESPECT TO THIS SPECIFICATION, WHETHER BASED ON BREACH OF CONTRACT, TORT (INCLUDING NEGLIGENCE), OR OTHERWISE, AND EVEN IF OCP HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Contents

[1 Introduction 4](#_Toc7764652)

[1.1 Terms and Conventions 4](#_Toc7764653)

[1.2 Purpose 6](#_Toc7764654)

[1.3 References 6](#_Toc7764655)

[2 Overview 7](#_Toc7764656)

[3 Blob Wrap Key Format 8](#_Toc7764657)

[4 Keyblob Formats 8](#_Toc7764658)

[4.1 Summary of Formats 8](#_Toc7764659)

[4.2 Endianness 9](#_Toc7764660)

[4.3 Format Definitions 9](#_Toc7764661)

[4.3.1 KeyblobType 1 10](#_Toc7764662)

[4.3.2 KeyblobTypes 2 Through 6 10](#_Toc7764663)

[4.3.3 KeyblobType 7, KeyblobType 8, KeyblobType9 11](#_Toc7764664)

[4.3.4 KeyblobTypes 10 Through 13 12](#_Toc7764665)

[5 Keyblob Example 14](#_Toc7764666)

# Introduction

Encryption and authentication are commonly used to protect against data theft and corruption. Most encryption/authentication methods use one or more keys. A keyblob allows users to provide encryption/authentication keys directly or indirectly to hardware performing crypto operations. In addition, a keyblob can be encrypted (i.e. wrapped). Once encrypted, the keyblob can be stored anywhere in a system, including permanent storage. An encrypted keyblob must be decrypted (i.e. unwrapped) before being used. The format of the keyblob must be agreed on between the generator and the user of the keyblob. This specification describes the Zipline keyblob formats.

## Terms and Conventions

**Keyblob:** User provided key related information used to generate encryption and/or authentication keys.

**KeyblobType:**  A value which indicates the specific format used for a keyblob.

**Blob Authentication Key (BAK):** Cryptographic key inside a keyblob which can be used to generate a derived data authentication key.

**Blob Encryption Key (BEK):** Symmetric cryptographic key inside a keyblob which can be used to generate a derived data encryption key.

**Encrypted Blob Authentication Key (eBAK):** An encrypted form of the Blob AuthenticationKey (BAK) which resides in an encrypted keyblob. Decrypting the encrypted keyblob restores the original BAK.

**Encrypted Blob Encryption Key (eBEK):** An encrypted Blob EncryptionKey (BEK) which resides in an encrypted keyblob. Decrypting the encrypted keyblob restores the original BEK.

**Blob Wrap Key (BWK): A 256b key use to encrypt/decrypt a keyblob.**

**eKeyblob:** A encrypted keyblob in which the BAK and/or BEK have wrapped to produce an eBAK and eBAK.

**eBWK:** Key used to encrypt/decrypt the BEK portion of an eKeyblob.

**aBWK:** Key used to encrypt/decrypt the BAK portion of an eKeyblob.

**aeBWK**: Key used to encrypt/decrypt both the BEK and BAK portion of an eKeyblob.

**Initialization Vector (IV):** A fixed-size input to a cryptographic primitive that is required to be random or pseudorandom. Each IV should be a unique value within a system. This requirement can be satisfied with the approach described in NIST SP800-90A Revision 1.

**eIV:** IV data used when encrypting the BEK.

**aIV:** IV data used when encrypting the BAK.

**aeIV:** IV data used when encrypting both BEK and BAK

**Authentication Tag (AuthTag):** a fixed-size value associated with encrypted text that is used to verify the integrity of encrypted data.

**Authentication Tag for eBEK (eAuthTag):** Authentication tag for the eBEK and possibly the eBAK portion of an encrypted keyblob.

**Authentication Tag for eBAK (aAuthTag):** Authentication tag for the eBAK portion of an encrypted keyblob.

**Key Derivation Function (KDF):** An algorithm that uses a master key and non-secret fixed data to generate a new key(s). The KDF is application dependent and is not described by this specification.

**Derived Data Encryption Key (DDEK):** Final encryption key output used for downstream cryptographic operations. The format and use of this key is application dependent and is not described by this specification.

**Derived Data Authentication Key (DDAK):** Final authentication key output used for downstream cryptographic operations. The format and use of this key is application dependent and is not described by this specification.

## Purpose

This specification allows generators and users of Zipline keyblobs to interoperate.

The specification includes the following:

* The keyblob format for each KeyblobType value
* How to wrap/unwrap the material each keyblob

To provide some context a potential usage of a keyblob is described. However, this is not part of the specification as the actual usage of the wrapped material is application dependent and is NOT defined by this specification.

## References

* [AES] IPS PUB 197: Advanced Encryption Standard (AES), <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>
* [AESGCM] NIST Special Publication SP800-38D defining GCM and GMAC - <http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf>
* [DRNG] NIST SP800-90A Revision 1  
  http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf

# Overview

The following diagram illustrates the generation and usage of keyblobs.



To generate a keyblob the user supplies a BEK and/or BAK along with IV, Blob Wrap Key(s) and the Key Type. The format of these inputs is defined in this specification. The delivery of the inputs is application dependent. If the inputs are not handled in a secure manner the protection provided by the keyblob will be lost. The algorithm used to wrap encrypted keyblobs is AES-GCM 256 SP-800-38D. There is no restriction on what combination of hardware and software performs the wrapping. However, it is recommended that this be done in a secure environment. All information required to reverse the operation is contained within the keyblob *except* the KeyblobType and the Blob Wrap Key (BWK). In addition, authentication tags are generated and are part of the keyblob. To maintain security the Blob Wrap Key should be stored in a secure location which can be accessed securely when the keyblob is unwrapped. The KeyblobType is not included in the keyblob to conserve storage. It is the responsibility of the application to track the format used for each keyblob.

Once an encrypted keyblob is generated it can be stored anywhere the application wishes. Without access to the correct Blob Wrap Key the original BEK and BAK cannot be recovered.

When the application wishes to use the BEK/BAK keys it supplies the keyblob containing the BEK/BAK along with the KeyblobType. These may be supplied over secure or unsecure channels. In addition, the Blob Wrap Key must be securely delivered to the combination of hardware and software which performs the unwrapping. There is no restriction on what combination of hardware and software performs the unwrapping. It is recommended that this be done in a secure place in the system but that lies outside the scope of this keyblob format specification. In addition to recovering the BEK/BAK the authentication tags should be used to authenticate the keys to ensure the keyblob was not corrupted. It is possible for an application to skip this step but that is highly undesirable.

This specification does not determine how the recovered keys are used. For context the gray boxes in the left of the diagram above illustrate one possibility.

# Blob Wrap Key Format

The Blob Wrap Key (BWK) is used to perform AES-GCM 256b encryption on all or a portion of the keyblob. The BWK is always 256b in length.

# Keyblob Formats

## Summary of Formats

There are total of 13 KeyblobTypes specified. There are three categories of keyblobs:

* KeyblobType 1: There are no keys present. The keyblob size is 0 bytes.
* KeyblobType 2-6: These keyblobs are not wrapped.
* KeyblobType 7-13: These keyblobs contain an encrypted BEK and/or encrypted BAK of various sizes, depending on the KeyblobType. They are encrypted using AES-GCM 256 SP-800-38D with a 96-bit IV and a 96-bit authentication tag. Users must provide the Blob Wrap Key(s) when creating or using a keyblob.

The following table summarizes the contents of the keyblob for each KeyblobType and which keys and IVs are used for wrapping and unwrapping the keyblob. Table 1 KeyBlob Content with Different KeyblobTypes.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Key-  blob  Type | Key-  blob  Name | Len (bits) | eBEK + e[a]IV +  e[a]AuthTag  Size (b) | eBAK +  aIV +  aAuthTag  Size (b) | Wrap  Key  Used | IV  Used | Notes |
| 0 | NULL | 0 | 0 | 0 | na | na | No KeyBlob |
| 1 | other | 0 | 0 | 0 | na | na | Reserved to indicate key comes from another source |
| 2 | UKB\_DEK256 | 256 | 256 | 0 | na | na | BEK(256b) only |
| 3 | UKB\_DEK512 | 512 | 512 | 0 | na | na | BEK(512b) only |
| 4 | UKB\_DAK256 | 256 | 0 | 256 | na | na | BAK(256b) only |
| 5 | UNK\_COMB256 | 512 | 256 | 256 | na | na | Combined BEK(256b) and BAK(256) |
| 6 | UNK\_COMB512 | 768 | 512 | 256 | na | na | Combined BEK(512b) and BAK(256) |
| 7 | EKB\_DEK256 | 448 | 448 | 0 | eBWK | eIV | eBEK (256b) only |
| 8 | EKB\_DEK512 | 704 | 704 | 0 | eBWK | eIV | eBEK (512b) only |
| 9 | EKB\_DAK256 | 448 | 0 | 448 | aBWK | aIV | eBAK (256b) only |
| 10 | EKB\_IND256 | 896 | 448 | 448 | eBWK  aBWK | eIV  aIV | Individual eBEK (256b) and eBAK(256b) |
| 11 | EKB\_IND512 | 1152 | 704 | 448 | eBWK  aBWK | eIV  aIV | Individual eBEK (512b) and eBAK(256b) |
| 12 | EKB\_COMB256 | 704 | 448 | 256 | aeBWK | aeIV | Combined eBEK (256b) and eBAK(256b) |
| 13 | EKB\_COMB512 | 960 | 704 | 256 | aeBWK | aeIV | Combined eBEK (512b) and eBAK(256b) |

## 

## Endianness

All fields in the keyblob are treated as a byte array. There is no notion of “most significant byte” for an IV, key or authentication tag. The keyblob diagrams below show only the sequence in the byte array.

## Format Definitions

Following the conventions in the NIST specifications the bytes for each field are shown in big endian format in the diagrams below. The location of each *byte* is shown in the diagrams, not each bit.

### KeyblobType 1

KeyblobType 1 doesn’t have keyblob associated with it.

### KeyblobTypes 2 Through 6

The next 5 formats define keyblobs which are *not* wrapped.



### KeyblobType 7, KeyblobType 8, KeyblobType9

The next 3 formats have a single key and a corresponding wrapping IV. The source of the key is implementation independent.



### KeyblobTypes 10 Through 13

The are 4 encrypted formats which contain both data and authentication keys. Either a single wrapping key or multiple wrapping keys may be used to wrap the key material.

# Keyblob Example

This section provides an example of building a keyblob where Keyblob Type is 11.

The fields used to create the keyblob and the resulting field values in the keyblob are:

eBWK = b442f6bf03271e62b37b232edd5f219f0f92b0511e2a77e26e8ac95e593e3222  
aBWK = 637fe9110c62eb7bc26c912baf3bb5dc1ac0ef6089468b8cf8a2890265cd82f1

eIV = 7c1c71cbc5227a2b0b381904  
aIV = edf16ace06d476cbb2d02a9c

BEK = 0802f7b719a089a5e8f7fa6f4544fffa4c2f82378eda60f5c8426d81f39ec3cd  
 4f5e98dc39bdecc36e6e68d44fa4ae1723b11db3eee6a850a6ded1c24a825f9f  
eBEK = 3e0173c6f6937c1957a669dd232fd606f58d52ab4e6f6656239e509d4a7c2d65  
 c8db7a1f3bc55af93f8830db6f442314f90d2f9fd30f4ddf48d89987f9651bac

BAK = 4d0e38ffdf2e60b0cd048ea1af6f596b5ba3ed6db53960350b01755d7f9f1356  
eBAK = 2fc4d77e3ad186bdd204a93d32631255569dfb1a7be0f7d8b694b495fae117cd

eAuthTag = 31bd054d9f551e660ebd90c4  
aAuthTag = b216d38cf312942c57398a92

Reading and printing the keyblob one byte at a time results in the following output:

**B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF**

ffffab03`8854f538  7c 1c 71 cb c5 22 7a 2b-0b 38 19 04 3e 01 73 c6

ffffab03`8854f548  f6 93 7c 19 57 a6 69 dd-23 2f d6 06 f5 8d 52 ab

ffffab03`8854f558  4e 6f 66 56 23 9e 50 9d-4a 7c 2d 65 c8 db 7a 1f

ffffab03`8854f568  3b c5 5a f9 3f 88 30 db-6f 44 23 14 f9 0d 2f 9f

ffffab03`8854f578  d3 0f 4d df 48 d8 99 87-f9 65 1b ac 31 bd 05 4d

ffffab03`8854f588  9f 55 1e 66 0e bd 90 c4-ed f1 6a ce 06 d4 76 cb

ffffab03`8854f598  b2 d0 2a 9c 2f c4 d7 7e-3A d1 86 bd d2 04 a9 3d

ffffab03`8854f5a8  32 63 12 55 56 9d fb 1a 7b e0 f7 d8 b6 94 b4 95

ffffab03`8854f5b8  fa e1 17 cd b2 16 d3 8c f3 12 94 2c 57 39 8a 92