Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 19, 2021

Y. Sheffer
G. Keselman
Intuit
Y. Nir
Dell Technologies

January 15, 2021

A Generic Ciphertext Format
draft-sheffer-ietf-ciphertext-format-01

Abstract

This document defines a set of structured headers for encrypted data. The main goal of this format is to enable detection of encrypted data in large data stores, and associating it back to the system where it was created and the key with which it was encrypted. This allows organizations to extend the concept of data governance to encrypted data, and to manage such data even when encrypted by multiple different systems and cloud providers.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 19, 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction and Design Principles 2
  1.1. Terminology 3
2. Motivation 3
  2.1. Design Goals 3
  2.2. Previous Work 4
3. The Ciphertext Format 4
  3.1. Format Overview 4
    3.1.1. Fixed Header 5
    3.1.2. Variable Header 5
    3.1.3. Deriving a Specific Key 6
  3.2. Receiving Ciphertext 7
  3.3. Fixed Header Rationale 7
4. Example 8
  4.1. Fixed Header 8
  4.2. Variable Header: CBOR Diagnostic Notation 8
  4.3. Variable Header: Binary 8
  4.4. Complete Header 8
  4.5. CDDL 8
5. IANA Considerations 9
6. Security Considerations 9
  6.1. Integrity Protection 9
7. References 9
  7.1. Normative References 9
  7.2. Informative References 10
  7.3. URIs 10
Appendix A. Document History 11
  A.1. draft-sheffer-ietf-ciphertext-format-01 11
  A.2. draft-sheffer-ietf-ciphertext-format-00 11
Authors' Addresses 11

1. Introduction and Design Principles

Organizations that manage sensitive data often employ application-level encryption to protect data at rest. When this solution is used, it is common that very large numbers of encrypted data items are stored, potentially for a long time. Security best practices, complicated organizational structures, as well as the existence of modern key management systems, lead to the proliferation of large numbers of encryption keys. After a while it becomes difficult to identify the encryption key that was used for a particular piece of

data, with the situation becoming even more complicated when multiple key management systems are used by the same organization.

Application-level encryption can be deployed at different scales: in some cases a multi-megabyte file may be encrypted with a single key. In other cases, we may want to deploy encryption for specific database fields, which can easily manifest itself as millions of keys for a single database table.

Tagging encrypted data with metadata supports a number of important use cases: it allows the organization to better catalog the data (a.k.a. "data governance"), to discover the owner of each piece of encrypted data, to detect data encrypted with outdated keys.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Motivation

Our main goal in defining a common ciphertext format is to allow organizations to manage large scale data, encrypted at rest using multiple key management and encryption services. Additional motivations for an enterprise to use a common format are:

- Cross-KMS-provider interoperability, to simplify automated management of data sourced from multiple origins.

- Proprietary data encryption formats mean that the data remains tied to a single vendor.

- Standardization around key management best practices.

2.1. Design Goals

Some of the goals behind this design include:

- The format should allow simple and efficient detection of encrypted data, in support of automated data governance and key lifecycle management.

- The format should be space-efficient, since it may be used for very large numbers of small encrypted items. As a result,

important information is associated with the (stored) key, rather than the ciphertext.

- Specifically, following security best practices, a given key material should be used with only a single cryptographic algorithm. Therefore, the algorithm identifier should be stored with the key (or the key version), rather than with the ciphertext.

- The format defined here only covers the ciphertext header, and not the ciphertext itself (referred to as "body" in this document). The body is defined elsewhere, such as [NISTSP800-38D] for AES-GCM.

- The header is not encrypted. Integrity-protection is optional. See Section 6.1 for details.

- The format should support key versioning, i.e. automated, periodic rotation of keys.

- The format should support granular key management by allowing for key derivation and key wrapping.

- The format should allow for generic tools to perform partial attribution of ciphertext, i.e. to associate it with a specific key provider. More specific, possibly provider-specific tools are required for full attribution.

2.2. Previous Work

A few notable formats are:

- The Amazon Web Services SDK message format, documented here [1]. This format is specific to the AWS library, and aimed at users of the AWS Key Management System (KMS).

- The wire format [2] defined by Google's Tink library.

- The format defined by the KMIP 2.1 [3] specification, which is targeted at data transmittal, rather than storage.

3. The Ciphertext Format

3.1. Format Overview

The ciphertext is prefixed by a header, which in turn, consists of a short fixed header and variable header. The variable header is a CBOR [RFC8949] map.

Following the header is the body of the ciphertext. The format (including length) of the body is out of scope for this document.

3.1.1. Fixed Header

The fixed header consists of:

- A single constant octet 0x08 (see Section 3.3).
- A single octet denoting the format version. The version is 0x01 for the format defined in this document.

3.1.2. Variable Header

The variable header is a CBOR map consisting of elements from the following table.

Field Name	Map Key	Value Type	Meaning	Mandatory
Key Provider	1	Unsigned integer	The organization responsible for the key management system.	Y
Key ID	2	Byte string	An encryption key identifier, where the key is stored in a key management system. This must denote a unique key, even if the Provider supports multiple tenants. Encoding of this field is Provider-specific. The field must appear once.	Y
Key Version	3	Unsigned integer	A version of a key, where the key is rotated on a periodic basis. Encoding of this field is Provider-specific. The field must appear at most	N

			once.		
Auxiliary Data	4	Byte string	Additional data required to derive a specific key from the referenced key (and key version, if any), see also Section 3.1.3. The field must appear at most once.	N	
Nonce	5	Byte string	A nonce or initialization vector (IV), if required by the cipher algorithm. We note that an implementation may prefer to store the nonce and authentication tag in-line with the ciphertext.	N	
Authentication Tag	6	Byte string	An authentication tag or integrity check value (ICV), if required by the cipher algorithm.	N	
Additional Authenticated Data	7	Byte string	Additional authenticated data (AAD), which is integrity-protected but not encrypted by the cipher.	N	

3.1.3. Deriving a Specific Key

The Auxiliary Data field is used to support derivation of a key, specific to the ciphertext being managed. There are two common ways to obtain this specific key:

- Using a key derivation function: SK = KDF(key, aux-data)

- Decryption of a wrapped key: SK = Decrypt(key, aux-data)

The exact algorithm is implementation dependent, and should be uniquely defined by the combination of Key Provider, Key ID and (if given) Key Version.

3.2. Receiving Ciphertext

Correct interpretation of the format may have security implications, making it important to define the exact semantics even when the entity that receives a ciphertext may not understand parts of the header.

- A recipient MUST reject a malformed header, e.g. if the total length is larger than the physical length allocated to it based on higher-level network protocols or storage formats.

- A recipient MUST reject a ciphertext if it does not recognize the format version.

- A recipient MUST reject a ciphertext if the variable header is not valid CBOR, as per [RFC8949] Sec. 5.3.1. In particular, it MUST reject duplicate map keys.

- A recipient MUST accept a ciphertext even if it does not recognize some of the map keys. It MUST ignore the unknown map keys and MUST interpret all known ones. In other words, the only way to introduce new mandatory map keys is by incrementing the format version.

- If ciphertext integrity protection coverage includes the header, a recipient MUST reject the header as well as the ciphertext if the integrity protection fails to validate.

3.3. Fixed Header Rationale

We chose the initial byte 0x08, since strings are very unlikely to start with it, as we explain below. Automated tools can detect encrypted data in structured contexts (e.g., a SQL database column) by sampling a number of data items and if all start with this byte, determining that they are encrypted with a high probability.

The byte 0x08 encodes the ASCII control character "backspace". It has the same meaning in UTF-8, and the 08 block of UTF-16 characters is only populated by two very small languages and rarely-used extended Arabic characters [4].

4. Example

4.1. Fixed Header

"08 01"

4.2. Variable Header: CBOR Diagnostic Notation

" {1: 65535, 2: h'1122334455', 3: 6, } "

4.3. Variable Header: Binary

" a3 01 19 ff ff 02 45 11 22 33 44 55 03 06 "

4.4. Complete Header

" 08 01 a3 01 19 ff ff 02 45 11 22 33 44 55 03 06 "

4.5. CDDL

The following non-normative snippet defines the format of the variable header using CDDL [RFC8610].

```
var_header = {
        K_KEY_PROVIDER: uint,
        K_KEY_ID: bstr,
        ? K_KEY_VERSION: uint,
        ? K_AUX_DATA: bstr,
        ? K_NONCE : bstr,
        ? K_AUTH_TAG : bstr,
        ? K_AAD : bstr,
        *uint => any ; extensions
}

K_RESERVED = 0
K_KEY_PROVIDER = 1
K_KEY_ID = 2
K_KEY_VERSION = 3
K_AUX_DATA = 4
K_NONCE = 5
K_AUTH_TAG = 6
K_AAD = 7
        ; extend here
```

5. IANA Considerations

TBD: establish a registry for Types, with 128-255 as private use.

TBD: establish a registry of Key Providers.

6. Security Considerations

6.1. Integrity Protection

The format defined here does not include integrity protection for the header, and neither does it mandate that the encrypted item's integrity protection should include the header.

Data encrypted at rest is typically vulnerable to denial of service attacks, since (assuming the data is integrity protected) an attacker that can change the ciphertext can trivially cause it to fail validation.

There are cases where it is convenient to manipulate the ciphertext header, even if the data itself remains encrypted and unmodified. For example, when migrating between formats or when bulk-changing metadata associated with the ciphertext. On the other hand, it is a best practice to protect cryptographic metadata against malicious modification. We are currently not aware of a specific threat vector associated with malicious changes to the proposed format, at least assuming the use of AEAD ciphers.

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/info/rfc8949>.

7.2. Informative References

[NISTSP800-38D]
Dworkin, M., "Recommendation for block cipher modes of operation :: GaloisCounter Mode (GCM) and GMAC", National Institute of Standards and Technology report, DOI 10.6028/nist.sp.800-38d, 2007.

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://www.rfc-editor.org/info/rfc8610>.

7.3. URIs

[1] https://docs.aws.amazon.com/encryption-sdk/latest/developer-guide/message-format.html

[2] https://github.com/google/tink/blob/master/docs/WIRE-FORMAT.md

[3] https://docs.oasis-open.org/kmip/kmip-profiles/v2.1/csprd01/kmip-profiles-v2.1-csprd01.html

[4] https://en.wikipedia.org/wiki/Arabic_Extended-A

Appendix A. Document History

A.1. draft-sheffer-ietf-ciphertext-format-01

- SAAG feedback: the variable header is now CBOR.
- Binary example.
- Non-normative CDDL.
- Additional types for non-inline AEAD.

A.2. draft-sheffer-ietf-ciphertext-format-00

- Initial version.

Authors' Addresses

Yaron Sheffer
Intuit

EMail: yaronf.ietf@gmail.com

Gleb Keselman
Intuit

EMail: gleb.keselman@gmail.com

Yoav Nir
Dell Technologies

EMail: ynir.ietf@gmail.com