AEAD Modes for Kerberos GSS-API
draft-howard-gssapi-aead-00

Abstract

This document updates RFC4121 with support for encryption mechanisms that can authenticate associated data such as Counter with CBC-MAC (CCM) and Galois/Counter Mode (GCM). These mechanisms are often more performant and need not expand the message as much as conventional modes.

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 http://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 June 15, 2016.

Copyright Notice

Copyright (c) 2015 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 (http://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 ........................................... 2
2. Requirements notation .................................. 2
3. Authenticated Encryption with Associated Data (AEAD) Overview 2
4. Updates to RFC 2743 .................................... 3
4.1. GSS_Wrap_AEAD ...................................... 3
4.2. GSS_Unwrap_AEAD .................................... 4
5. Updates to RFC 4121 .................................... 4
5.1. Support for Associated Data .......................... 4
5.2. Existing Encryption Types ............................ 5
5.3. Native AEAD Encryption Types ........................ 5
5.3.1. Restriction on Native AEAD Usage ................. 5
5.3.2. Application-provided Cipherstate .................. 5
5.3.3. Encryption and Checksum Operations .............. 6
5.3.4. DCE RPC Interoperability ......................... 7
6. Security Considerations ............................... 7
7. Acknowledgements ....................................... 8
8. References ............................................. 8
8.1. Normative References ............................... 8
8.2. Informative References ............................. 8
Author’s Address .......................................... 9

1. Introduction

This document updates [RFC4121] with support for encryption mechanisms that support Authenticated Encryption with Associated Data (AEAD). These mechanisms often have performance advantage over conventional encryption modes as they can be efficiently parallelized and do not expand the plaintext when encrypting.

In addition, this document defines new GSS-API functions for protecting associated data in addition to a plaintext.

2. Requirements notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Authenticated Encryption with Associated Data (AEAD) Overview

The Kerberos 5 GSS-API mechanism specified in [RFC4121] provides for the authenticated encryption of plaintext, that is, it provides both for confidentiality and a way to check the plaintext for integrity and authenticity.
It can be useful in many applications to provide for the integrity and authenticity of some additional unencrypted data; this is termed Authenticated Encryption with Associated Data (AEAD). This can be done by the generic composition of existing encryption and checksum mechanisms, or using algorithms which specifically provide for AEAD (see [RFC5116]). The latter class of algorithms, referred to as native AEAD, may have additional constraints (further described in [KRB-AEAD]).

4. Updates to RFC 2743

[RFC2743] is updated with variations of GSS_Wrap() and GSS_Unwrap() that permit the inclusion of associated data to be authenticated along with a plaintext.

[[CREF1: TBD: do we allow interleaved plaintext and associated data (which SSPI does and indeed requires for DCE), or do we limit it to a single octet string each? If the former, we need to define GSS_Wrap_IOV instead of GSS_Wrap_AEAD (and the Unwrap equivalents).]]

4.1. GSS_Wrap_AEAD

Inputs:

- context_handle CONTEXT HANDLE,
- conf_req_flag BOOLEAN,
- qop_req INTEGER, -- 0 specifies default QOP
- input_assoc_data OCTET STRING, -- associated data
- input_message OCTET STRING -- plaintext

Outputs:

- major_status INTEGER,
- minor_status INTEGER,
- conf_state BOOLEAN,
- output_message OCTET STRING -- caller must release with GSS_Release_buffer()

Performs the data origin authentication, data integrity and (optionally) data confidentiality functions of GSS_Wrap(), additionally integrity protecting the data in input_assoc_data.
Return values are as for GSS_Wrap(). Note that output_message does not include the data in input_assoc_data.

4.2. GSS_Unwrap_AEAD

Inputs:

-o context_handle CONTEXT HANDLE,
-o input_message OCTET STRING, -- plaintext
-o input_assoc_data OCTET STRING -- associated data

Outputs:

-o conf_state BOOLEAN,
-o qop_state INTEGER,
-o major_status INTEGER,
-o minor_status INTEGER,
-o output_message OCTET STRING -- caller must release with GSS_Release_buffer()

Processes a data element generated (and optionally encrypted) by GSS_Wrap(), provided as input_message, additionally validating the data origin and integrity of input_assoc_data. Return values are as for GSS_Unwrap(). Note that output_message does not include the data in input_assoc_data.

5. Updates to RFC 4121

5.1. Support for Associated Data

The generation of per-message tokens using the GSS_Wrap_AEAD() and GSS_Unwrap_AEAD() functions is identical to GSS_Wrap() and GSS_Unwrap(), except that:

-o the encrypt-with-ad and decrypt-with-ad functions are used instead of the encrypt and decrypt functions (respectively)
-o the input_assoc_data parameter is passed as the associated data
-o the is-longterm parameter is always false
5.2. Existing Encryption Types

For existing encryption mechanisms that use a generic composition of encryption and checksum functions (such as the Simplified Profile in [RFC3961]), the only operative difference to [RFC4121] is that the associated data is prepended to the plaintext before invoking the checksum function. As such, for these encryption types GSS_Wrap_AEAD() with no associated data has an identical output to GSS_Wrap().

5.3. Native AEAD Encryption Types

When used with native AEAD encryption types as defined in [KRB-AEAD], the generation of [RFC4121] per-message tokens is modified as described below.

5.3.1. Restriction on Native AEAD Usage

Implementations SHALL NOT use native AEAD encryption types where the deterministic cipherstate length is less than 12 octets (96 bytes).

[[CREF2: TBD: if we want to support CCM with a 32-bit counter, we could remove the Filler byte and reduce the required cipherstate length to 11 octets. However, this may make it more difficult to use TLS-oriented GCM implementations that expose the Fixed-Common and Fixed-Distinct nonce components independently.]]

Native AEAD encryption types that do not support long-term keys SHOULD only be negotiated for use in GSS-API using the cryptosystem negotiation extension defined in [RFC4537].

5.3.2. Application-provided Cipherstate

The cipherstate for each invocation of encrypt-with-ad or decrypt-with-ad is given as follows. (For consistency with [RFC4121] the following definition uses 0-based indexing.)
The output cipherstate from the encrypt-with-ad and decrypt-with-ad functions is discarded as it is always specified explicitly as described above.

The use of application-managed cipherstate allows the per-message token size to be reduced by omitting the confounder and encrypted copy of the token header. There is no limit on the number or size of messages that can be protected beyond those imposed by the sequence number size and the underlying cryptosystem.

5.3.3. Encryption and Checksum Operations

This text amends [RFC4121] Section 4.2.4.

In Wrap tokens that provide for confidentiality, the first 16 octets of the token (the "header", as defined in [RFC4121] Section 4.2.6) SHALL NOT be appended to the plaintext data before encryption. Instead, the TOK_ID, Flags and SND_SEQ fields of the token header are protected by the initialization vector (cipherstate). The EC field is unprotected, a change from [RFC4121]. The receiver MUST explicitly validate the EC field. For the native AEAD encryption types profiled in [KRB-AEAD] Section 5, EC SHALL be zero (except when GSS_C_DCE_STYLE is in use, see below). This specification does not support native AEAD encryption types that require the plaintext to be padded.

In Wrap tokens that do not provide for confidentiality, the first 16 octets of the token SHALL NOT be appended to the to-be-signed plaintext data. As with Wrap tokens that do provide for confidentiality, all fields except EC and RRC are protected by the initialization vector. The receiver MUST validate that EC is the correct constant value. For the AEAD encryption types defined in
Section 5, EC SHALL be sixteen, reflecting the tag length of 16 octets (128 bits).

Because native AEAD encryption types lack an explicit checksum operation, MIC tokens are generated similarly to Wrap tokens, using the encrypt-with-ad function passing the to-be-signed data as the associated data and using a plaintext length of zero. The key usage and initialization vector serve to disambiguate MIC from Wrap tokens. The octet string output by the encrypt-with-ad function contains the authentication tag, which is placed in the SGN_CKSUM field of the token.

5.3.4. DCE RPC Interoperability

Existing implementations that support the GSS_C_DCE_STYLE context flag will, when this flag is in set, set EC for Wrap tokens with confidentiality to the underlying cipher’s block size and use an effective Right Rotation Count (RRC) of EC + RRC bytes. This document does not specify otherwise.

When GSS_C_DCE_STYLE is set, receivers MUST verify that the otherwise unprotected EC field is the underlying cipher’s block size for Wrap tokens with confidentiality. (For Wrap tokens without confidentiality, the EC field remains the length of the authentication tag.)

DCE interleaves plaintext and associated data; because native AEAD algorithms may require associated data to be processed before any plaintext, any plaintext and associated data must each be coalesced before encrypting or decrypting. This document does not specify an API for processing interleaved plaintext and associated data.

6. Security Considerations

The combination of a context-specific session key and the presence of the TOK_ID and SND_SEQ fields in the cipherstate guarantees that the key/IV combination is safe from reuse. The allows native AEAD modes such as [GCM] and [CCM] to be used securely.

Because the initialization vector has a deterministic (but non-repeating) construction, it is safe for use with GCM without any limitation on the number of invocations of the authenticated encryption function other than that imposed by the requirement that the cipherstate not repeat. (Section 8.3 of [GCM] imposes an invocation limit of $2^{32}$ where the cipherstate is randomly generated or is a length other than 96 bits.)
The reordering of plaintext and associated data for GSS_C_DCE_STYLE interoperability may be problematic where the plaintext and associated data lengths are variable.

7. Acknowledgements

The author would like to thank the following individuals for their comments and suggestions: Nicolas Williams and Greg Hudson.

8. References

8.1. Normative References


8.2. Informative References


Author’s Address

Luke Howard
PADL Software
PO Box 59
Central Park, VIC 3145
Australia

Email: lukeh@padl.com