

NETWORK WORKING GROUP
Internet-Draft
Obsoletes: 4402 (if approved)
Intended status: Standards Track
Expires: June 13, 2016

S. Emery
Oracle
N. Williams
Cryptonector
December 11, 2015

A Pseudo-Random Function (PRF) for the Kerberos V Generic Security
Service Application Program Interface (GSS-API) Mechanism
draft-ietf-kitten-rfc4402bis-02

Abstract

This document defines the Pseudo-Random Function (PRF) for the Kerberos V mechanism for the Generic Security Service Application Program Interface (GSS-API), based on the PRF defined for the Kerberos V cryptographic framework, for keying application protocols given an established Kerberos V GSS-API security context.

This document obsoletes RFC 4402 and reclassifies that document as historic. RFC 4402 starts the PRF+ counter at 1, however a number of implementations starts the counter at 0. As a result, the original specification would not be interoperable with existing implementations.

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 13, 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. Conventions Used in This Document	2
3. Kerberos V GSS Mechanism PRF	2
4. IANA Considerations	3
5. Security Considerations	3
6. Acknowledgements	4
7. Normative References	4
Appendix A. Test Vectors	6
Authors' Addresses	8

1. Introduction

This document specifies the Kerberos V GSS-API mechanism's [RFC4121] pseudo-random function corresponding to [RFC4401]. The function is a "PRF+" style construction. For more information see [RFC4401], [RFC2743], [RFC2744] and [RFC4121].

This document obsoletes RFC 4402 and reclassifies that document as historic. RFC 4402 starts the PRF+ counter at 1, however a number of implementations starts the counter at 0. As a result, the original specification would not be interoperable with existing implementations.

2. Conventions Used in This Document

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. Kerberos V GSS Mechanism PRF

The GSS-API PRF [RFC4401] function for the Kerberos V mechanism [RFC4121] shall be the output of a PRF+ function based on the encryption type's PRF function keyed with the negotiated session key of the security context corresponding to the 'prf_key' input parameter of GSS_Pseudo_random().

This PRF+ MUST be keyed with the key indicated by the 'prf_key' input parameter as follows:

- o GSS_C_PRF_KEY_FULL -- use the sub-session key asserted by the acceptor, if any exists, or the sub-session asserted by the initiator, if any exists, or the Ticket's session key
- o GSS_C_PRF_KEY_PARTIAL -- use the sub-session key asserted by the initiator, if any exists, or the Ticket's session key

The PRF+ function is a simple counter-based extension of the Kerberos V pseudo-random function [RFC3961] for the encryption type of the security context's keys:

$$\text{PRF+}(K, L, S) = \text{truncate}(L, T_0 \parallel T_1 \parallel \dots \parallel T_n)$$
$$T_n = \text{pseudo-random}(K, n \parallel S)$$

where K is the key indicated by the 'prf_key' parameter, where '||' is the concatenation operator, 'n' is encoded as a network byte order 32-bit unsigned binary number, truncate(L, S) truncates the input octet string S to length L, and pseudo-random() is the Kerberos V pseudo-random function [RFC3961].

The maximum output size of the Kerberos V mechanism's GSS-API PRF then is, necessarily, 2³² times the output size of the pseudo-random() function for the encryption type of the given key.

When the input size is longer than 2¹⁴ octets as per [RFC4401] and exceeds an implementation's resources, then the mechanism MUST return GSS_S_FAILURE and GSS_KRB5_S_KG_INPUT_TOO_LONG as the minor status code.

4. IANA Considerations

This document has no IANA considerations currently. If and when a relevant IANA registry of GSS-API symbols and constants is created, then the GSS_KRB5_S_KG_INPUT_TOO_LONG minor status code should be added to such a registry.

5. Security Considerations

Kerberos V encryption types' PRF functions use a key derived from contexts' session keys and should preserve the forward security properties of the mechanisms' key exchanges.

Legacy Kerberos V encryption types may be weak, particularly the single-DES encryption types.

See also [RFC4401] for generic security considerations of `GSS_Pseudo_random()`.

See also [RFC3961] for generic security considerations of the Kerberos V cryptographic framework.

Use of Ticket session keys, rather than sub-session keys, when initiators and acceptors fail to assert sub-session keys, is dangerous as ticket reuse can lead to key reuse; therefore, initiators should assert sub-session keys always, and acceptors should assert sub-session keys at least when initiators fail to do so.

The computational cost of computing this PRF+ may vary depending on the Kerberos V encryption types being used, but generally the computation of this PRF+ gets more expensive as the input and output octet string lengths grow (note that the use of a counter in the PRF+ construction allows for parallelization).

6. Acknowledgements

This document is an update to Nico Williams' RFC. Greg Hudson has provided the test vectors based on MIT's implementation.

7. 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, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC2743] Linn, J., "Generic Security Service Application Program Interface Version 2, Update 1", RFC 2743, DOI 10.17487/RFC2743, January 2000, <<http://www.rfc-editor.org/info/rfc2743>>.
- [RFC2744] Wray, J., "Generic Security Service API Version 2 : C-bindings", RFC 2744, DOI 10.17487/RFC2744, January 2000, <<http://www.rfc-editor.org/info/rfc2744>>.
- [RFC3961] Raeburn, K., "Encryption and Checksum Specifications for Kerberos 5", RFC 3961, DOI 10.17487/RFC3961, February 2005, <<http://www.rfc-editor.org/info/rfc3961>>.

- [RFC4121] Zhu, L., Jaganathan, K., and S. Hartman, "The Kerberos Version 5 Generic Security Service Application Program Interface (GSS-API) Mechanism: Version 2", RFC 4121, DOI 10.17487/RFC4121, July 2005, <<http://www.rfc-editor.org/info/rfc4121>>.
- [RFC4401] Williams, N., "A Pseudo-Random Function (PRF) API Extension for the Generic Security Service Application Program Interface (GSS-API)", RFC 4401, DOI 10.17487/RFC4401, February 2006, <<http://www.rfc-editor.org/info/rfc4401>>.

Appendix A. Test Vectors

Here are some test vectors from the MIT implementation provided by Greg Hudson. Test cases used include input string lengths of 0 and 61 bytes, and an output length of 44 bytes. 61 bytes of input is just enough to produce a partial second MD5 or SHA1 hash block with the four-byte counter prefix. 44 bytes of output requires two full and one partial RFC 3961 PRF output for all existing encetypes. All keys were randomly generated.

Enctype: des-cbc-crc

Key: E607FE9DABB57AE0

Input: (empty string)

Output: 803C4121379FC4B87CE413B67707C4632EBED2C6D6B7
2A55E878836E35E21600D915D590DED5B6D77BB30A1F

Enctype: des-cbc-crc

Key: 54758316B6257A75

Input: ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz123456789

Output: 279E4105F7ADC9BD6EF28ABE31D89B442FE0058388BA
33264ACB5729562DC637950F6BD144B654BE7700B2D6

Enctype: des3-cbc-sha1

Key: 70378A19CD64134580C27C0115D6B34A1CF2FEECEF9886A2

Input: (empty string)

Output: 9F8D127C520BB826BFF3E0FE5EF352389C17E0C073D9
AC4A333D644D21BA3EF24F4A886D143F85AC9F6377FB

Enctype: des3-cbc-sha1

Key: 3452A167DF1094BA1089E0A20E9E51ABEF1525922558B69E

Input: ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz123456789

Output: 6BF24FABC858F8DD9752E4FCD331BB831F238B5BE190
4EEA42E38F7A60C588F075C5C96A67E7F8B7BD0AECF4

Enctype: rc4-hmac

Key: 3BB3AE288C12B3B9D06B208A4151B3B6

Input: (empty string)

Output: 9AEa11A3BCF3C53F1F91F5A0BA2132E2501ADF5F3C28
3C8A983AB88757CE865A22132D6100EAD63E9E291AFA

Enctype: rc4-hmac

Key: 6DB7B33A01BD2B72F7655CB7B3D5FA0B

Input: ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz123456789

Output: CDA9A544869FC84873B692663A82AFDA101C8611498B
A46138B01E927C9B95EEC953B562807434037837DDDF

Enctype: aes128-cts-hmac-sha1-96

Key: 6C742096EB896230312B73972FA28B5D

Input: (empty string)

Output: 94208D982FC1BB7778128BDD77904420B45C9DA699F3
117BCE66E39602128EF0296611A6D191A5828530F20F

Enttype: aes128-cts-hmac-sha1-96

Key: FA61138C109D834A477D24C7311BE6DA

Input: ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz123456789

Output: 0FAEDF0F842CC834FEE750487E1B622739286B975FE5
B7F45AB053143C75CA0DF5D3D4BBB80F6A616C7C9027

Enttype: aes256-cts-hmac-sha1-96

Key: 08FCDAFD5832611B73BA7B497FEBFF8C954B4B58031CAD9B977C3B8C25192FD6

Input: (empty string)

Output: E627EFC14EF5B6D629F830C7109DEA0D3D7D36E8CD57
A1F301C5452494A1928F05AFFBEE3360232209D3BE0D

Enttype: aes256-cts-hmac-sha1-96

Key: F5B68B7823D8944F33F41541B4E4D38C9B2934F8D16334A796645B066152B4BE

Input: ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz123456789

Output: 112F2B2D878590653CCC7DE278E9F0AA46FA5A380B62
59F774CB7C134FCD37F61A50FD0D9F89BF8FE1A6B593

Enttype: camellia128-cts-cmac

Key: 866E0466A178279A32AC0BDA92B72AEB

Input: (empty string)

Output: 97FBB354BF341C3A160DCC86A7A910FDA824601DF677
68797BACEEBF5D250AE929DEC9760772084267F50A54

Enttype: camellia128-cts-cmac

Key: D4893FD37DA1A211E12DD1E03E0F03B7

Input: ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz123456789

Output: 1DEE2FF126CA563A2A2326B9DD3F0095013257414C83
FAD4398901013D55F367C82681186B7B2FE62F746BA4

Enttype: camellia256-cts-cmac

Key: 203071B1AE77BD3D6FCE70174AF95C225B1CED46B35CF52B6479EFEB47E6B063

Input: (empty string)

Output: 9B30020634C10FDA28420CEE7B96B70A90A771CED43A
D8346554163E5949CBAE2FB8EF36AFB6B32CE75116A0

Enttype: camellia256-cts-cmac

Key: A171AD582C1AFBBAD52ABD622EE6B6A14D19BF95C6914B2BA40FFD99A88EC660

Input: ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz123456789

Output: A47CBB6E104DCC77E4DB48A7A474B977F2FB6A7A1AB6
52317D50508AE72B7BE2E4E4BA24164E029CBACF786B

Authors' Addresses

Shawn Emery
Oracle Corporation
500 Eldorado Blvd Bldg 1
Broomfield, CO 78727
US

EMail: shawn.emery@oracle.com

Nicolas Williams
Cryptonector, LLC

EMail: nico@cryptonector.com