Network Working Group
Internet-Draft
Intended status: Informational
Expires: November 1, 2020

L. Howard
PADL

April 30, 2020

# A Simple Anonymous GSS-API Mechanism
# draft-howard-gss-sanon-14

Abstract

This document defines protocols, procedures and conventions for a Generic Security Service Application Program Interface (GSS-API) security mechanism that provides key agreement without authentication of either party.

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 November 1, 2020.

Copyright Notice

Copyright (c) 2020 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 . . . . . 2
2. Requirements notation . . . . . 2
3. Discovery and Negotiation . . . . . 3
4. Naming . . . . . 3
4.1. Mechanism Names . . . . . 3
4.2. Display Name Format . . . . . 3
4.3. Exported Name Format . . . . . 3
5. Definitions and Token Formats . . . . . 4
5.1. Context Establishment Tokens . . . . . 4
5.1.1. Initial context token . . . . . 4
5.1.2. Acceptor context token . . . . . 5
5.1.3. Initiator context completion . . . . . 5
5.2. Per-Message Tokens . . . . . 6
5.3. Context Deletion Tokens . . . . . 6
6. Key derivation . . . . . 6
7. Pseudo-Random Function . . . . . 7
8. Security Considerations . . . . . 7
9. Acknowledgements . . . . . 7
10. References . . . . . 7
10.1. Normative References . . . . . 8
10.2. Informative References . . . . . 8
Appendix A. Test Vectors . . . . . 9
Appendix B. Mechanism Attributes . . . . . 10
Appendix C. NegoEx . . . . . 10
Author's Address . . . . . 11

## 1. Introduction

The Generic Security Service Application Program Interface (GSS-API) [RFC2743] provides a framework for authentication and message protection services through a common programming interface.

The Simple Anonymous mechanism (hereafter SAnon) described in this document is a simple protocol based on the X25519 elliptic curve Diffie-Hellman (ECDH) key agreement scheme defined in [RFC7748]. No authentication of initiator or acceptor is provided. A potential use of SAnon is to provide a degree of privacy when bootstrapping unkeyed entities.

## 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. Discovery and Negotiation

The SAnon mechanism is identified by the following OID:

```
sanon-x25519 OBJECT IDENTIFIER ::=
    {iso(1)identified-organization(3)dod(6)internet(1)
     private(4)enterprise(1)padl(5322)gss-sanon(26)
     mechanisms(1)sanon-x25519(110)}
```

The means of discovering GSS-API peers and their supported mechanisms is out of this specification's scope. To avoid multiple layers of negotiation, SAnon is not crypto-agile; a future variant using a different algorithm would be assigned a different OID.

If anonymity is not desired then SAnon MUST NOT be used. Either party can test for anon_state (GSS_C_ANON_FLAG) to check if anonymous authentication was performed.

## 4. Naming

### 4.1. Mechanism Names

A SAnon mechanism name is abstractly a boolean indicating whether it represents an anonymous identity. Anonymous identities are names imported with the GSS_C_NT_ANONYMOUS name type. Implementations MAY map other names to anonymous identities according to local policy. Names representing non-anonymous identities MUST be importable so that initiators with non-default credentials can engage SAnon by setting anon_req_flag (GSS_C_ANON_FLAG).

### 4.2. Display Name Format

When GSS_Display_name() is called on a mechanism name representing an anonymous identity, the display string is WELLKNOWN/ANONYMOUS@WELLKNOWN:ANONYMOUS [RFC8062] and the name type is GSS_C_NT_ANONYMOUS. This is always the name observed by a SAnon peer. All context APIs that return peer names MUST return this name for both parties if the context is established.

### 4.3. Exported Name Format

SAnon uses the mechanism-independent exported name object format defined in [RFC2743] Section 3.2. All lengths are encoded as big-endian integers. The export of non-anonymous mechanism names MUST fail with GSS_S_BAD_NAME.

| Length | Name | Description |
|---|---|---|
| 2 | TOK_ID | 04 01 |
| 2 | MECH_OID_LEN | Length of the mechanism OID |
| MECH_OID_LEN | MECH_OID | The SAnon mechanism OID, in DER |
| 4 | NAME_LEN | 00 00 00 01 |
| 1 | NAME | 01 |

# 5. Definitions and Token Formats

## 5.1. Context Establishment Tokens

### 5.1.1. Initial context token

The initial context token is framed per Section 1 of [RFC2743]:

```
GSS-API DEFINITIONS ::=
    BEGIN

    MechType ::= OBJECT IDENTIFIER -- 1.3.6.1.4.1.5322.26.1.110
    GSSAPI-Token ::=
    [APPLICATION 0] IMPLICIT SEQUENCE {
         thisMech MechType,
         innerToken ANY DEFINED BY thisMech
             -- 32 byte initiator public key
             -- 8 byte protocol flags (optional)
    }
    END
```

On the first call to GSS_Init_sec_context(), the mechanism checks if one or more of the following are true:

The caller set anon_req_flag (GSS_C_ANON_FLAG)

The claimant credential identity is anonymous (see Section 4.1)

The claimant credential is the default one and target identity is anonymous

If none of these are the case, the call MUST fail with GSS_S_UNAVAILABLE.

If proceeding, the initiator generates a fresh secret and public key pair per [RFC7748] Section 6.1 and returns GSS_S_CONTINUE_NEEDED, indicating that a subsequent context token from the acceptor is expected. The innerToken field of the output_token contains the initiator's 32 byte public key, optionally concatenated with a 64-bit big-endian integer containing flags that are not optional and the acceptor would be otherwise be unable to infer (such as those defined in [RFC4757] Section 7.1).

Portable initiators are RECOMMENDED to use default credentials whenever possible and request anonymity only through anon_req_flag (see [RFC8062] Section 6).

### 5.1.2. Acceptor context token

Upon receiving a context token from the initiator, the acceptor validates that the token is well formed. The acceptor generates a fresh secret and public key pair. The context session key is computed as specified in Section 6.

The acceptor constructs an output_token by concatenating its public key with the token emitted by calling GSS_GetMIC() with the default QOP and zero-length octet string. The output token is sent to the initiator without additional framing.

The acceptor then returns GSS_S_COMPLETE, setting src_name to the canonical anonymous name. The reply_det_state (GSS_C_REPLAY_FLAG), sequence_state (GSS_C_SEQUENCE_FLAG), conf_avail (GSS_C_CONF_FLAG), integ_avail (GSS_C_INTEG_FLAG) and anon_state (GSS_C_ANON_FLAG) security context flags are set, along with any additional flags received from the initiator. The context is ready to use.

### 5.1.3. Initiator context completion

Upon receiving the acceptor context token and verifying it is well formed, the initiator extracts the acceptor's public key (being the first 32 bytes of the input token) and computes the context session key per Section 6.

The initiator calls GSS_VerifyMIC() with the MIC extracted from the context token and the zero-length octet string. If successful, the initiator returns GSS_S_COMPLETE to the caller, to indicate the initiator is authenticated and the context is ready for use. No output token is emitted. The security context flags are set as for the acceptor, including any additional flags sent in the initial context token.

### 5.2. Per-Message Tokens

The per-message tokens definitions are imported from [RFC4121] Section 4.2. The base key used to derive specific keys for signing and sealing messages is defined in Section 6. The [RFC3961] encryption and checksum algorithms use the aes128-cts-hmac-sha256-128 encryption type defined in [RFC8009]. The AcceptorSubkey flag as defined in [RFC4121] Section 4.2.2 MUST be set.

### 5.3. Context Deletion Tokens

Context deletion tokens are empty in this mechanism. The behavior of GSS_Delete_sec_context() [RFC2743] is as specified in [RFC4121] Section 4.3.

## 6. Key derivation

The context session key is known as the base key, and is computed using a key derivation function from [SP800-108] Section 5.1 (using HMAC as the PRF):

```
base key = HMAC-SHA-256(K1, i | label | 0x00 | context | L)
```

where:

| | |
|---|---|
| K1 | the output of X25519(local secret key, peer public key) as specified in [RFC7748] Section 6.1 |
| i | the constant 0x00000001, representing the iteration count expressed in big-endian binary representation of 4 bytes |
| label | the string "sanon-x25519" (without quotation marks) |
| context | initiator public key \| acceptor public key \| flags \| channel binding application data (if present) |
| L | the constant 0x00000080, being length in bits of the key to be outputted expressed in big-endian binary representation of 4 bytes |

The flags input to the context contains any flags sent by the initiator, defaulting to zero if none were sent, expressed in big-endian binary representation of 8 bytes.

The inclusion of channel bindings in the key derivation function means that the acceptor cannot ignore initiator channel bindings; this differs from some other mechanisms. Being the only variable

length input to the key derivation function, the length is not included.

The base key provides the acceptor-asserted subkey defined in [RFC4121] Section 2 and is used to generate keys for per-message tokens and the GSS-API PRF. Its encryption type is aes128-cts-hmac-sha256-128 per [RFC8009]. The [RFC3961] algorithm protocol parameters are as given in [RFC8009] Section 5.

## 7. Pseudo-Random Function

The [RFC4401] GSS-API pseudo-random function for this mechanism imports the definitions from [RFC8009], using the base key for both GSS_C_PRF_KEY_FULL and GSS_C_PRF_KEY_PARTIAL usages.

## 8. Security Considerations

This document defines a GSS-API security mechanism, and therefore deals in security and has security considerations text embedded throughout. This section only addresses security considerations associated with the SAnon mechanism described in this document. It does not address security considerations associated with the GSS-API itself.

This mechanism provides only for key agreement. It does not authenticate the identity of either party. It MUST NOT be selected if either party requires identification of its peer.

SAnon mechanism names are not unary: there may be many real identities that map to either the anonymous or non-anonymous mechanism name. As such, implementations MUST ensure that GSS_Compare_name() always sets name_equal to FALSE when comparing mechanism names.

## 9. Acknowledgements

AuriStor, Inc funded the design of this protocol, along with an implementation for the Heimdal GSS-API library.

Jeffrey Altman, Greg Hudson, Simon Josefsson, and Nicolas Williams provided valuable feedback on this document.

## 10. References

## 10.1. 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>.

[RFC2743] Linn, J., "Generic Security Service Application Program Interface Version 2, Update 1", RFC 2743, DOI 10.17487/RFC2743, January 2000, <https://www.rfc-editor.org/info/rfc2743>.

[RFC3961] Raeburn, K., "Encryption and Checksum Specifications for Kerberos 5", RFC 3961, DOI 10.17487/RFC3961, February 2005, <https://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, <https://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, <https://www.rfc-editor.org/info/rfc4401>.

[RFC7748] Langley, A., Hamburg, M., and S. Turner, "Elliptic Curves for Security", RFC 7748, DOI 10.17487/RFC7748, January 2016, <https://www.rfc-editor.org/info/rfc7748>.

[RFC8009] Jenkins, M., Peck, M., and K. Burgin, "AES Encryption with HMAC-SHA2 for Kerberos 5", RFC 8009, DOI 10.17487/RFC8009, October 2016, <https://www.rfc-editor.org/info/rfc8009>.

## 10.2. Informative References

[I-D.zhu-negoex] Short, M., Zhu, L., Damour, K., and D. McPherson, "SPNEGO Extended Negotiation (NEGOEX) Security Mechanism", draft-zhu-negoex-04 (work in progress), January 2011.

[RFC4178] Zhu, L., Leach, P., Jaganathan, K., and W. Ingersoll, "The Simple and Protected Generic Security Service Application Program Interface (GSS-API) Negotiation Mechanism", RFC 4178, DOI 10.17487/RFC4178, October 2005, <https://www.rfc-editor.org/info/rfc4178>.

[RFC4757] Jaganathan, K., Zhu, L., and J. Brezak, "The RC4-HMAC Kerberos Encryption Types Used by Microsoft Windows", RFC 4757, DOI 10.17487/RFC4757, December 2006, <https://www.rfc-editor.org/info/rfc4757>.

[RFC5587] Williams, N., "Extended Generic Security Service Mechanism Inquiry APIs", RFC 5587, DOI 10.17487/RFC5587, July 2009, <https://www.rfc-editor.org/info/rfc5587>.

[RFC8062] Zhu, L., Leach, P., Hartman, S., and S. Emery, Ed., "Anonymity Support for Kerberos", RFC 8062, DOI 10.17487/RFC8062, February 2017, <https://www.rfc-editor.org/info/rfc8062>.

[SP800-108] Chen, L., "Recommendation for Key Derivation Using Pseudorandom Functions (Revised)", October 2009.

## Appendix A. Test Vectors

The example exchange below contains no additional flags or channel binding information.

| | |
|---|---|
| initiator secret key | 83 33 f2 ea 2a 22 eb aa 05 39 c6 06 1d 6a 99 05<br>84 24 49 9e 2c 16 c1 b1 34 d9 22 27 f3 f4 5e bd |
| initiator public key | 5f 40 66 22 5a 3c fd 72 57 23 c1 8f ae 71 3e 8c<br>ab 32 a7 2c 93 b9 76 66 04 4b 8f e4 a0 c9 69 19 |
| initiator token | 60 2c 06 0a 2b 06 01 04 01 a9 4a 1a 01 6e 5f 40<br>66 22 5a 3c fd 72 57 23 c1 8f ae 71 3e 8c ab 32<br>a7 2c 93 b9 76 66 04 4b 8f e4 a0 c9 69 19 |
| acceptor secret key | b0 db 16 32 39 0a dd 93 1e f7 62 bc d3 c9 1d 03<br>e8 d9 59 52 48 eb e2 f2 b5 f7 d8 06 ec dd 50 60 |
| acceptor public key | 2f 81 51 9f a8 9c 07 f8 eb b2 95 6c 0c c3 22 77<br>ae a1 0e 62 0c 79 33 81 ef 9a c5 b2 f0 d9 1e 06 |
| base key | 80 76 2c 43 32 6a 95 f5 be 30 6d ea 10 ba f3 d0 |
| acceptor token | 2f 81 51 9f a8 9c 07 f8 eb b2 95 6c 0c c3 22 77<br>ae a1 0e 62 0c 79 33 81 ef 9a c5 b2 f0 d9 1e 06<br>04 04 05 ff ff ff ff ff 00 00 00 00 00 00 00 00<br>4d 5e a9 e0 e1 9c 7a 61 c2 6a 9a c5 e8 17 5f 04 |
| initiator negoex key | 2a c8 f9 d0 31 87 40 42 cb d4 50 07 ce db c2 c2 |

acceptor negoex key 73 9f 4d a2 f1 2d f7 f7 d7 ea e4 9d a4 08 62 5b

## Appendix B. Mechanism Attributes

The [RFC5587] mechanism attributes for this mechanism are:

GSS_C_MA_MECH_CONCRETE

GSS_C_MA_ITOK_FRAMED

GSS_C_MA_AUTH_INIT_ANON

GSS_C_MA_AUTH_TARG_ANON

GSS_C_MA_INTEG_PROT

GSS_C_MA_CONF_PROT

GSS_C_MA_MIC

GSS_C_MA_WRAP

GSS_C_MA_REPLAY_DET

GSS_C_MA_OOS_DET

GSS_C_MA_CBINDINGS

GSS_C_MA_PFS

GSS_C_MA_CTX_TRANS

## Appendix C. NegoEx

When SAnon is negotiated by [I-D.zhu-negoex], the authentication scheme identifier is DEE384FF-1086-4E86-BE78-B94170BFD376.

The initiator and acceptor keys for NegoEx checksum generation and verification are derived using the GSS-API PRF (see Section 7), with the input data "sanon-x25519-initiator-negoex-key" and "sanon-x25519-acceptor-negoex-key" respectively (without quotation marks). No metadata is defined and any, if present, SHOULD be ignored.

It is RECOMMENDED that GSS-API implementations supporting both SPNEGO [RFC4178] and NegoEx advertise SAnon under both to maximise interoperability.

Author's Address

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

Email: lukeh@padl.com