NETWORK WORKING GROUP
Internet-Draft
Intended status: Standards Track
Expires: October 1, 2017

N. Williams
Cryptonector LLC
A. Melnikov
Isode Ltd

March 30, 2017

# Namespace Considerations and Registries for GSS-API Extensions
# draft-ietf-kitten-gssapi-extensions-iana-11.txt

## Abstract

This document describes the ways in which the GSS-API may be extended and directs the creation of an IANA registry for various GSS-API namespaces.

## 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 October 1, 2017.

## Copyright Notice

Copyright (c) 2017 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. Conventions used in this document . . . . . 2
2. Introduction . . . . . 2
3. Extensions to the GSS-API . . . . . 2
4. Generic GSS-API Namespaces . . . . . 3
5. Language Binding-Specific GSS-API Namespaces . . . . . 3
6. Extension-Specific GSS-API Namespaces . . . . . 4
7. Registration Form . . . . . 4
8. IANA Considerations . . . . . 6
8.1. Initial Namespace Registrations . . . . . 7
8.1.1. Example registrations . . . . . 7
8.2. Registration Maintenance Guidelines . . . . . 9
8.2.1. Sub-Namespace Symbol Pattern Matching . . . . . 10
8.2.2. Expert Reviews of Individual Submissions . . . . . 10
8.2.3. Change Control . . . . . 11
9. Security Considerations . . . . . 12
10. References . . . . . 12
10.1. Normative References . . . . . 12
10.2. Informative References . . . . . 12
Authors' Addresses . . . . . 13

## 1. 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].

## 2. Introduction

There is a need for private-use and mechanism-specific extensions to the Generic Security Services Application Programming Interface (GSS-API). As such extensions are designed and standardized (or not), both at the IETF and elsewhere, there is a non-trivial risk of namespace pollution and conflicts. To avoid this we set out guidelines for extending the GSS-API and direct the creation of an IANA registry for GSS-API namespaces.

Registrations of individual items and sub-namespaces are allowed. Each sub-namespace may provide different rules for registration, e.g., for mechanism-specific and private-use extensions.

## 3. Extensions to the GSS-API

Extensions to the GSS-API can be categorized as follows:

- Abstract API extensions

- Implementation-specific
- Mechanism-specific
- Language binding-specific

Extensions to the GSS-API may be purely semantic, without effect on the GSS-API's namespaces. Or they may introduce new functions, constants, types, etc...; these clearly affect the GSS-API namespaces.

Extensions that affect the GSS-API namespaces should be registered with the IANA as described herein.

## 4. Generic GSS-API Namespaces

The abstract API namespaces for the GSS-API are:

- Type names
- Function names
- Constant names for various types
- Constant values for various types
- Name types (OID, type name and syntaxes)

Additionally we have namespaces associates with the OBJECT IDENTIFIER (OID) type. The IANA already maintains a registry of such OIDs:

- Mechanism OIDs
- Name Type OIDs

## 5. Language Binding-Specific GSS-API Namespaces

Language binding specific namespaces include, among others:

- Header/interface module names
- Object classes and/or types
- Methods and/or functions
- Constant names
- Constant values

## 6. Extension-Specific GSS-API Namespaces

Extensions to the GSS-API may create additional namespaces. See Section 8.2.

## 7. Registration Form

Registrations for GSS-API namespaces SHALL take the following form:

| Registration Field | Possible Values | Description |
|---|---|---|
| Bindings | 'Generic', 'C-bindings', 'Java', 'C#', <programming language name> | Indicates the name of the programming language that this registration involves, or, if 'Generic', that this is an entry for the generic abstract GSS-API (i.e., not specific to any programming language). |
| Registration type | 'Instance', 'Sub-Namespace' | Indicates whether this entry reserves a given symbol name (and possibly, constant value), or whether it reserves an entire sub-namespace (the name is a pattern) or constant value range. |
| Object Type | <Symbol> defined by the binding language (for example 'Data-Type', 'Function', 'Method', 'Integer', 'String', 'OID', 'Context-Flag', 'Name-Type', 'Macro', 'Header-File-Name', 'Module-Name', 'Class') | Indicates the type of the object whose symbolic name or constant value this entry registers. The possible values of this field depend on the programming language in question, therefore they are not all specified here. |
| Symbol Name/Prefix | <Symbol name or name pattern> | The name of a symbol or symbol sub-namespace being |

| | | |
|---|---|---|
| | | registered. See Section 8.2.1 |
| Binding of | <Name of abstract API element of which this object is a binding> | If the registration is for a specific language binding of the GSS-API, then this names the abstract API element of which it is a binding (OPTIONAL). |
| Constant Value/Range | <Constant value> or <constant value range> | The value of the constant named by the <Symbol Name/Prefix>. This field is present only for Instance and Sub-namespace registrations of Constant object types. |
| Description | <Text> | Description of the registration. Multiple instances of this field may result (see Section 8.2.3). |
| Registration Rules | <Reference> to an IANA registration Policy defined in [RFC5226] (or an RFC that updates it), for instance 'IESG Approval', 'Expert Review', 'First Come First Served', 'Private Use'. | Describes the rules for allocation of items that fall in this sub-namespace, for entries with Registration Type of Sub-namespace (OPTIONAL). For private use sub-namespaces the submitter MUST provide the e-mail address of a responsible contact. If this field is not specified for a sub-namespace, the default registration rules specified in Section 8.2 apply. |
| Reference | <Reference> | Reference to a document that describes the registration, if any (OPTIONAL). Multiple instances of this field are allowed, with one reference each. |
| Expert Reviewer | <Name of expert reviewers, possibly | OPTIONAL, see Section 8.2.2. Multiple instances of this |

| | | |
|---|---|---|
| | WG names> | field are allowed, with one expert reviewer per-instance. Leave this field blank when requesting a registration. It will be filled in by the Expert who reviews the registration. |
| Expert Review Notes | <Notes from the expert review> | Expert reviewers may request that some comments be included with the registration, e.g., regarding security considerations of the registered extension. |
| Status | 'Registered' or 'Obsoleted' | Status of the registration. |
| Obsoleting Reference | <Reference> | Reference to a document, if any, that obsoletes this registration. Multiple instances of this field are allowed, with one reference each. (OPTIONAL) |

The IANA should create a single GSS-API namespace registry, or multiple registries, one for symbolic names and one for constant values, and/or it may create a registry per-programming language, at its convenience.

Entries in these registries should consist of all the fields from their corresponding registration entries.

Entries should be sorted by: programming language, registration type, object type, and symbol name/pattern.

## 8. IANA Considerations

This document deals with IANA considerations throughout. Specifically it creates a single registry of various kinds of things, though the IANA may instead create multiple registries, each for one of those kinds of things. Of particular interest may be that IANA will now be the registration authority for the GSS-API name type OID space.

### 8.1. Initial Namespace Registrations

Initial registry content corresponding to the items defined in [RFC2743], [RFC2744], [RFC2853], [RFC1964] and [RFC4121] and others will be supplied during the IANA review portion of the RFC publishing process. [[Note to RFC Editor: Delete the following sentence before publication:]] The KITTEN WG chairs MUST indicate that such content has been reviewed by the WG and that there is WG consensus that the entries are in agreement with those RFCs.

#### 8.1.1. Example registrations

In order to sanity check recommended IANA registration templates, this section registers several entries.

| Registration Field | Possible Values |
|---|---|
| Bindings | C-bindings |
| Registration type | Instance |
| Object Type | Function |
| Symbol Name | gss_init_sec_context |
| Binding of | GSS_Init_sec_context |
| Constant Value/Range | N/A |
| Description | Create a security context by initiator |
| Registration Rules | N/A |
| Reference | RFC 2744 |
| Expert Reviewer | Kitten WG |
| Expert Review Notes | |
| Status | Registered |
| Obsoleting Reference | N/A |

| Registration Field | Possible Values |
|---|---|
| Bindings | C-bindings |
| Registration type | Instance |
| Object Type | Function |
| Symbol Name | gss_accept_sec_context |
| Binding of | GSS_Accept_sec_context |
| Constant Value/Range | N/A |
| Description | Accept a security context from initiator |
| Registration Rules | N/A |
| Reference | RFC 2744 |
| Expert Reviewer | Kitten WG |
| Expert Review Notes | |
| Status | Registered |
| Obsoleting Reference | N/A |

| Registration Field | Possible Values |
|---|---|
| Bindings | C-bindings |
| Registration type | Instance |
| Object Type | Context-Flag |
| Symbol Name | GSS_C_DELEG_FLAG |
| Binding of | deleg_state or deleg_req_flag |
| Constant Value/Range | 1 |
| Description | On output (if set): Delegated credentials are available via the delegated_cred_handle parameter of GSS_Accept_sec_context. On input (if set): With the call to GSS_Init_sec_context, delegate credentials to the acceptor. |
| Registration Rules | N/A |
| Reference | RFC 2744 |
| Expert Reviewer | Kitten WG |
| Expert Review Notes | |
| Status | Registered |
| Obsoleting Reference | N/A |

## 8.2. Registration Maintenance Guidelines

Standards-Track RFCs can create new items with any non-conflicting Symbol Name/Prefix value for this registry by virtue of IESG approval to publish as a Standards-Track RFC -- that is, without additional expert review.

Standards-Track RFCs can mark existing entries as obsolete, and can even create conflicting entries if explicitly stated (the IESG, of course, should review conflicts carefully, and may reject them).

IANA shall also consider submissions from individuals, and via Informational and Experimental RFCs, subject to Expert Review. IANA SHALL allow such registrations if a) they are not conflicting, b) provided that the registration is for object types other than Context-Flags, and c) subject to expert review. Guidelines for expert reviews are given below.

#### 8.2.1. Sub-Namespace Symbol Pattern Matching

Sub-namespace registrations must provide a pattern for matching symbols for which the sub-namespace's registration rules apply. The pattern consists of a string with the following special tokens:

- '*', meaning "match any string."
- "%m", meaning "match any mechanism family short-hand name."
- "%i", meaning "match any implementor vanity short-hand name."

For example, "GSS_%m_*" matches "GSS_krb5_foo" since "krb5" is a common short-hand for the Kerberos V GSS-API mechanism [RFC1964]. But "GSS_%m_*" does not match "GSS_foo_bar" unless "foo" is asserted to be a short-hand for some mechanism.

#### 8.2.2. Expert Reviews of Individual Submissions

[[The following paragraph should be deleted from the document before publication, as it will not age well. It should be moved to the shepherding write-up.]]

Expert review selection SHALL be done as follows. If, at the time that the IANA receives an individual submission for registration in this registry, there are any IETF Working Groups chartered to produce GSS-API-related documents, then the IANA SHALL ask the chairs of such WGs to be expert reviewers or to name one. If there are no such WGs at that time, then the IANA SHALL ask past chairs of the KITTEN WG and the author/editor of this RFC to act as expert reviewers or name an alternate.

Expert reviewers of individual registration submissions with Registration Type == Sub-namespace should check that the registration request has a suitable description (which doesn't need to be sufficiently detailed for others to implement) and that the Symbol Name/Prefix is sufficiently descriptive of the purpose of the sub-namespace or reflective of the name of the submitter or associated company.

Expert reviewers of individual registration submissions with

Registration Type == Instance should check that the Symbol Name falls under a sub-namespace controlled by the submitter. Registration of such entries which do not fall under such a sub-namespace may be allowed provided that they correspond to long existing non-standard extensions to the GSS-API and this can be easily checked or demonstrated, otherwise IESG Protocol Action is REQUIRED (see previous section). Also, reviewers should check that any registration of constant values have a detailed description that is suitable for other implementors to reproduce, and that they don't conflict with other usages or are otherwise dangerous in the reviewers estimation.

Expert reviewers should review impact on mechanisms, security and interoperability, and may reject or annotate registrations which can have mechanism impact that requires IESG protocol action. Consider, for example, new versions of GSS_Init_sec_context() and/or GSS_Accept_sec_context which have new input and/or output parameters which imply changes on the wire or in behaviour that may result in interoperability issues. A reviewer could choose to add notes to the registration describing such issues, or the reviewer might conclude that the danger to Internet interoperability is sufficient to warrant rejecting the registration.

### 8.2.3. Change Control

Registered entries may be marked obsoleted using the same expert review process as for registering new entries. Obsoleted entries are not, however, to be deleted, but merely marked having Obsoleted Status. Note that entries may be created as obsoleted to record the fact that the given symbol(s) have been used before, even though continued use of them is discouraged.

Registered entries may also be updated in two other ways: additional references, obsoleting references, and descriptions may be added.

All changes are subject to expert review, except for changes to registrations in a sub-namespace which are subject to the rules of the relevant sub-namespace. The submitter of a change request need not be the same as the original submitter.

Registrations may be modified by addition, but under no circumstance may any fields be modified except for the Status field or Contact Address, or to correct for transcription errors in filing or processing registration requests.

The IANA SHALL add a field describing the date that a an addition or modification was made, and a description of the change.

## 9. Security Considerations

General security considerations relating to IANA registration services apply; see [RFC5226].

Also, expert reviewers should look for and may document security related issues with submitters' GSS-API extensions, to the best of the reviewers' ability given the information furnished by the submitter. Reviewers may add comments regarding their limited ability to review a submission for security problems if the submitter is unwilling to provide sufficient documentation.

## 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, <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>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

### 10.2. Informative References

[RFC1964] Linn, J., "The Kerberos Version 5 GSS-API Mechanism", RFC 1964, DOI 10.17487/RFC1964, June 1996, <http://www.rfc-editor.org/info/rfc1964>.

[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>.

[RFC2853] Kabat, J. and M. Upadhyay, "Generic Security Service API Version 2 : Java Bindings", RFC 2853, DOI 10.17487/RFC2853, June 2000, <http://www.rfc-editor.org/info/rfc2853>.

[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>.

## Authors' Addresses

Nicolas Williams
Cryptonector LLC

Email: nico@cryptonector.com

Alexey Melnikov
Isode Ltd
5 Castle Business Village
36 Station Road
Hampton, Middlesex TW12 2BX
UK

Email: Alexey.Melnikov@isode.com

NETWORK WORKING GROUP
Internet-Draft
Updates: 4120,4121 (if approved)
Intended status: Standards Track
Expires: October 1, 2017

B. Kaduk, Ed.
Akamai
J. Schaad, Ed.
Soaring Hawk Consulting
L. Zhu
Microsoft Corporation
J. Altman
Secure Endpoints
March 30, 2017

# Initial and Pass Through Authentication Using Kerberos V5 and the GSS-API (IAKERB)
draft-ietf-kitten-iakerb-03

## Abstract

This document defines extensions to the Kerberos protocol and the GSS-API Kerberos mechanism that enable a GSS-API Kerberos client to exchange messages with the KDC by using the GSS-API acceptor as a proxy, encapsulating the Kerberos messages inside GSS-API tokens. With these extensions a client can obtain Kerberos tickets for services where the KDC is not accessible to the client, but is accessible to the application server.

## 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 October 1, 2017.

## Copyright Notice

Copyright (c) 2017 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 . . . . . 3
3. GSS-API Encapsulation . . . . . 3
3.1. Enterprise principal names . . . . . 6
4. Finish Message . . . . . 7
5. Addresses in Tickets . . . . . 8
6. Security Considerations . . . . . 8
7. Acknowledgements . . . . . 9
8. Assigned Numbers . . . . . 10
9. IANA Considerations . . . . . 10
10. References . . . . . 10
10.1. Normative References . . . . . 10
10.2. Informative references . . . . . 11
Appendix A. Interoperate with Previous MIT version . . . . . 11
Authors' Addresses . . . . . 12

## 1. Introduction

When authenticating using Kerberos V5, clients obtain tickets from a KDC and present them to services. This model of operation cannot work if the client does not have access to the KDC. For example, in remote access scenarios, the client must initially authenticate to an access point in order to gain full access to the network. Here the client may be unable to directly contact the KDC either because it does not have an IP address, or the access point packet filter does not allow the client to send packets to the Internet before it authenticates to the access point. The Initial and Pass Through Authentication Using Kerberos (IAKERB) mechanism allows for the use of Kerberos in such scenarios where the client is unable to directly contact the KDC, by using the service to pass messages between the client and the KDC. This allows the client to obtain tickets from the KDC and present them to the service, as in normal Kerberos operation.

Recent advancements in extending Kerberos permit Kerberos authentication to complete with the assistance of a proxy. The

Kerberos [RFC4120] pre-authentication framework [RFC6113] prevents the exposure of weak client keys over the open network. The Kerberos support of anonymity [RFC6112] provides for privacy and further complicates traffic analysis. The kdc-referrals option defined in [RFC6113] may reduce the number of messages exchanged while obtaining a ticket to exactly two even in cross-realm authentications.

Building upon these Kerberos extensions, this document extends [RFC4120] and [RFC4121] such that the client can communicate with the KDC using a Generic Security Service Application Program Interface (GSS-API) [RFC2743] acceptor as a message-passing proxy. (This is completely unrelated to the type of proxy specified in [RFC4120].) The client acts as a GSS-API initiator, and the GSS-API acceptor relays the KDC request and reply messages between the client and the KDC, transitioning to normal [RFC4121] GSS-krb5 messages once the client has obtained the necessary credentials. Consequently, IAKERB as defined in this document requires the use of the GSS-API.

The GSS-API acceptor, when relaying these Kerberos messages, is called an IAKERB proxy.

## 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. GSS-API Encapsulation

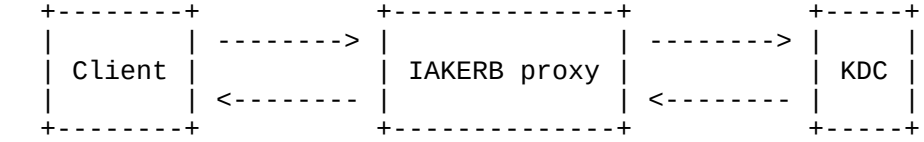
The GSS-API mechanism Objection Identifier (OID) for IAKERB is id-kerberos-iakerb:

```
id-kerberos-iakerb ::=
{ iso(1) org(3) dod(6) internet(1) security(5) kerberosV5(2)
iakerb(5) }
```

All context establishment tokens of IAKERB MUST have the token framing described in section 4.1 of [RFC4121] with the mechanism OID being id-kerberos-iakerb. MIT implemented an earlier draft of this specification; details on how to interoperate with that implementation can be found in Appendix A.

The client starts by constructing a ticket request, as if it is being made directly to the KDC. Instead of contacting the KDC directly, the client encapsulates the request message into the output token of the GSS_Init_security_context() call and returns GSS_S_CONTINUE_NEEDED [RFC2743], indicating that at least one more token is required in order to establish the context. The output

token is then passed over the application protocol for use as the input token to the GSS_Accept_sec_context() call in accordance with GSS-API. The GSS-API acceptor extracts the Kerberos request from the input token, locates the target KDC, and sends the request on behalf of the client. After receiving the KDC reply, the GSS-API acceptor then encapsulates the reply message into the output token of GSS_Accept_sec_context(). The GSS-API acceptor returns GSS_S_CONTINUE_NEEDED [RFC2743] indicating that at least one more token is required in order to establish the context. The output token is passed to the initiator over the application protocol in accordance with GSS-API.




For all context tokens generated by the IAKERB mechanism, the innerToken described in section 4.1 of [RFC4121] has the following format: it starts with a two-octet token-identifier (TOK_ID), which is followed by an IAKERB message or a Kerberos message.

Only one IAKERB specific message, namely the IAKERB_PROXY message, is defined in this document. The TOK_ID values for Kerberos messages are the same as defined in [RFC4121].

| Token | TOK_ID Value in Hex |
|---|---|
| IAKERB_PROXY | 05 01 |

The content of the IAKERB_PROXY message is defined as an IAKERB-HEADER structure immediately followed by a Kerberos message, which is optional. The Kerberos message can be an AS-REQ, an AS-REP, a TGS-REQ, a TGS-REP, or a KRB-ERROR as defined in [RFC4120].

```
IAKERB-HEADER ::= SEQUENCE {
    -- Note that the tag numbers start at 1, not 0, which would
    -- be more conventional for Kerberos.
    target-realm      [1] UTF8String,
       -- The name of the target realm.
    cookie            [2] OCTET STRING OPTIONAL,
       -- Opaque data, if sent by the server,
       -- MUST be copied by the client verbatim into
       -- the next IAKRB_PROXY message.
    ...
}
```

The IAKERB-HEADER structure and all the Kerberos messages MUST be encoded using Abstract Syntax Notation One (ASN.1) Distinguished Encoding Rules (DER) [CCITT.X680.2002] [CCITT.X690.2002].

The client fills out the IAKERB-HEADER structure as follows: the target-realm contains the realm name the ticket request is addressed to. In the initial message from the client, the cookie field is absent. The client MAY send a completely empty IAKERB_PROXY message (consisting solely of the octets 05 01 and an IAKERB_HEADER with zero-length target-realm) in order to query the Kerberos realm of the acceptor, see Section 3.1. In all other cases, the client MUST specify a target-realm. This can be the realm of the client's host, if no other realm information is available. client's host.

Upon receipt of the IAKERB_PROXY message, the GSS-API acceptor inspects the target-realm field in the IAKERB_HEADER, locates a KDC for that realm, and sends the ticket request to that KDC. The IAKERB proxy MAY engage in fallback behavior, retransmitting packets to a given KDC and/or sending the request to other KDCs in that realm if the initial transmission does not receive a reply, as would be done if the proxy was making requests on its own behalf.

The GSS-API acceptor encapsulates the KDC reply message in the returned IAKERB message. It fills out the target realm using the realm sent by the client and the KDC reply message is included immediately following the IAKERB-HEADER header.

When the GSS-API acceptor is unable to obtain an IP address for a KDC in the client's realm, it sends a KRB_ERROR message with the code KRB_AP_ERR_IAKERB_KDC_NOT_FOUND to the client in place of an actual reply from the KDC, and the context fails to establish. There is no accompanying error data defined in this document for this error code.

```
KRB_AP_ERR_IAKERB_KDC_NOT_FOUND      85
    -- The IAKERB proxy could not find a KDC.
```

When the GSS-API acceptor has an IP address for at least one KDC in the target realm, but does not receive a response from any KDC in the realm (including in response to retries), it sends a KRB_ERROR message with the code KRB_AP_ERR_IAKERB_KDC_NO_RESPONSE to the client and the context fails to establish. There is no accompanying error data defined in this document for this error code.

```
KRB_AP_ERR_IAKERB_KDC_NO_RESPONSE    86
    -- The KDC did not respond to the IAKERB proxy.
```

The IAKERB proxy can send opaque data in the cookie field of the IAKERB-HEADER structure in the server reply to the client, in order

to, for example, minimize the amount of state information kept by the GSS-API acceptor. The content and the encoding of the cookie field is a local matter of the IAKERB proxy. Whenever the cookie is present in a token received by the initiator, the initiator MUST copy the cookie verbatim into its subsequent response tokens which contain IAKERB_PROXY messages.

The client and the server can repeat the sequence of sending and receiving the IAKERB messages as described above for an arbitrary number of message exchanges, in order to allow the client to interact with the KDC through the IAKERB proxy, and to obtain Kerberos tickets as needed to authenticate to the acceptor.

Once the client has obtained the service ticket needed to authenticate to the acceptor, subsequent GSS-API context tokens are of type KRB_AP_REQ, not IAKERB_PROXY, and the client performs the client-server application exchange as defined in [RFC4120] and [RFC4121].

For implementations conforming to this specification, both the authenticator subkey and the GSS_EXTS_FINISHED extension as defined in Section 4 MUST be present in the AP-REQ authenticator. This checksum provides integrity protection for the IAKERB messages previously exchanged, including the unauthenticated clear texts in the IAKERB-HEADER structure.

If the pre-authentication data is encrypted in the long-term password-based key of the principal, the risk of security exposures is significant. Implementations SHOULD utilize the AS_REQ armoring as defined in [RFC6113] unless an alternative protection is deployed. In addition, the anonymous Kerberos FAST option is RECOMMENDED for the client to complicate traffic analysis.

### 3.1. Enterprise principal names

The introduction of principal name canonicalization by [RFC6806] created the possibility for a client to have a principal name (of type NT-ENTERPRISE) for which it is trying to obtain credentials, but no information about what realm’s KDC to contact to obtain those credentials. A Kerberos client not using IAKERB would typically resolve the NT-ENTERPRISE name to a principal name by starting from the realm of the client’s host and finding out the true realm of the enterprise principal based on referrals [RFC6806].

A client using IAKERB may not have any realm information, even for the realm of the client’s host, or may know that the client host’s realm is not appropriate for a given enterprise principal name. In such cases, the client can retrieve the realm of the GSS-API acceptor

as follows: the client returns GSS_S_CONTINUE_NEEDED with the output token containing an IAKERB message with an empty target-realm in the IAKERB-HEADER and no Kerberos message following the IAKERB-HEADER structure. Upon receipt of the realm request, the GSS-API acceptor fills out an IAKERB_PROXY response message, filling the target-realm field with the realm of the acceptor, and returns GSS_S_CONTINUE_NEEDED with the output token containing the IAKERB message with the server's realm and no Kerberos message following the IAKERB-HEADER header. The GSS-API initiator can then use the returned realm in subsequent IAKERB messages to resolve the NT-ENTERPRISE name type. Since the GSS-API acceptor can act as a Kerberos acceptor, it always has an associated Kerberos realm.

## 4. Finish Message

For implementations conforming to this specification, the authenticator subkey in the AP-REQ MUST alway be present, and the Exts field in the GSS-API authenticator [RFC6542] MUST contain an extension of type GSS_EXTS_FINISHED with extension data containing the ASN.1 DER encoding of the structure KRB-FINISHED.

```
GSS_EXTS_FINISHED             2
     --- Data type for the IAKERB checksum.

KRB-FINISHED ::= {
     -- Note that the tag numbers start at 1, not 0, which would be
     -- more conventional for Kerberos.
     gss-mic [1] Checksum,
         -- Contains the checksum [RFC3961] of the GSS-API tokens
         -- exchanged between the initiator and the acceptor,
         -- and prior to the containing AP_REQ GSS-API token.
         -- The checksum is performed over the GSS-API tokens
         -- exactly as they were transmitted and received,
         -- in the order that the tokens were sent.
      ...
}
```

The gss-mic field in the KRB-FINISHED structure contains a Kerberos checksum [RFC3961] of all the preceding context tokens of this GSS-API context (including the generic token framing of the GSSAPI-Token type from [RFC4121]), concatenated in chronological order (note that GSS-API context token exchanges are synchronous). The checksum type is the required checksum type of the enctype of the subkey in the authenticator, the protocol key for the checksum operation is the authenticator subkey, and the key usage number is KEY_USAGE_FINISHED.

```
KEY_USAGE_FINISHED            41
```

The GSS-API acceptor MUST then verify the checksum contained in the GSS_EXTS_FINISHED extension. This checksum provides integrity protection for the messages exchanged including the unauthenticated clear texts in the IAKERB-HEADER structure.

## 5. Addresses in Tickets

In IAKERB, the machine sending requests to the KDC is the GSS-API acceptor and not the client. As a result, the client should not include its addresses in any KDC requests for two reasons. First, the KDC may reject the forwarded request as being from the wrong client. Second, in the case of initial authentication for a dial-up client, the client machine may not yet possess a network address. Hence, as allowed by [RFC4120], the addresses field of the AS-REQ and TGS-REQ requests SHOULD be blank.

## 6. Security Considerations

The IAKERB proxy is a man-in-the-middle for the client's Kerberos exchanges. The Kerberos protocol is designed to be used over an untrusted network, so this is not a critical flaw, but it does expose to the IAKERB proxy all information sent in cleartext over those exchanges, such as the principal names in requests. Since the typical usage involves the client obtaining a service ticket for the service operating the proxy, which will receive the client principal as part of normal authentication, this is also not a serious concern. However, an IAKERB client not using an armored FAST channel [RFC6113] sends an AS_REQ with pre-authentication data encrypted in the long-term keys of the user, even before the acceptor is authenticated. This subjects the user's long-term key to an offline attack by the proxy. To mitigate this threat, the client SHOULD use FAST [RFC6113] and its KDC authentication facility to protect the user's credentials.

Similarly, the client principal name is in cleartext in the AS and TGS exchanges, whereas in the AP exchanges embedded in GSS context tokens for the regular krb5 mechanism, the client principal name is present only in encrypted form. Thus, more information is exposed over the network path between initiator and acceptor when IAKERB is used than when the krb5 mechanism is used, unless FAST armor is employed. (This information would be exposed in other traffic from the initiator when the krb5 mech is used.) As such, to complicate traffic analysis and provide privacy for the client, the client SHOULD request the anonymous Kerberos FAST option [RFC6113].

Similar to other network access protocols, IAKERB allows an unauthenticated client (possibly outside the security perimeter of an organization) to send messages that are proxied to servers inside the

perimeter. To reduce the attack surface, firewall filters can be applied to restrict from which hosts client requests can be proxied, and the proxy can further restrict the set of realms to which requests can be proxied.

In the intended use scenario, the client uses the proxy to obtain a TGT and then a service ticket for the service it is authenticating to (possibly preceeded by exchanges to produce FAST armor). However, the protocol allows arbitrary KDC-REQs to be passed through, and there is no limit to the number of exchanges that may be proxied. The client can send KDC-REQs unrelated to the current authentication, and obtain service tickets for other service principals in the database of the KDC being contacted.

In a scenario where DNS SRV RR's are being used to locate the KDC, IAKERB is being used, and an external attacker can modify DNS responses to the IAKERB proxy, there are several countermeasures to prevent arbitrary messages from being sent to internal servers:

1. KDC port numbers can be statically configured on the IAKERB proxy. In this case, the messages will always be sent to KDC's. For an organization that runs KDC's on a static port (usually port 88) and does not run any other servers on the same port, this countermeasure would be easy to administer and should be effective.

2. The proxy can do application level sanity checking and filtering. This countermeasure should eliminate many of the above attacks.

3. DNS security can be deployed. This countermeasure is probably overkill for this particular problem, but if an organization has already deployed DNS security for other reasons, then it might make sense to leverage it here. Note that Kerberos could be used to protect the DNS exchanges. The initial DNS SRV KDC lookup by the proxy will be unprotected, but an attack here is at most a denial of service (the initial lookup will be for the proxy's KDC to facilitate Kerberos protection of subsequent DNS exchanges between itself and the DNS server).

## 7. Acknowledgements

Jonathan Trostle, Michael Swift, Bernard Aboba and Glen Zorn wrote earlier revision of this document.

The hallway conversations between Larry Zhu and Nicolas Williams formed the basis of this document.

## 8. Assigned Numbers

The value for the error code KRB_AP_ERR_IAKERB_KDC_NOT_FOUND is 85.

The value for the error code KRB_AP_ERR_IAKERB_KDC_NO_RESPONSE is 86.

The key usage number KEY_USAGE_FINISHED is 41.

The key usage number KEY_USAGE_IAKERB_FINISHED is 42.

## 9. IANA Considerations

IANA is requested to make a modification in the "Kerberos GSS-API Token Type Identifiers" registry.

The following data to the table:

| ID | Description | Reference |
|---|---|---|
| 05 01 | IAKERB_PROXY | [THIS RFC] |

## 10. References

### 10.1. Normative References

[CCITT.X680.2002]
International Telephone and Telegraph Consultative Committee, "Abstract Syntax Notation One (ASN.1): Specification of basic notation", CCITT Recommendation X.680, July 2002.

[CCITT.X690.2002]
International Telephone and Telegraph Consultative Committee, "ASN.1 encoding rules: Specification of basic encoding Rules (BER), Canonical encoding rules (CER) and Distinguished encoding rules (DER)", CCITT Recommendation X.690, July 2002.

[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>.

[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>.

[RFC4120] Neuman, C., Yu, T., Hartman, S., and K. Raeburn, "The Kerberos Network Authentication Service (V5)", RFC 4120, DOI 10.17487/RFC4120, July 2005, <http://www.rfc-editor.org/info/rfc4120>.

[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>.

[RFC6542] Emery, S., "Kerberos Version 5 Generic Security Service Application Program Interface (GSS-API) Channel Binding Hash Agility", RFC 6542, DOI 10.17487/RFC6542, March 2012, <http://www.rfc-editor.org/info/rfc6542>.

## 10.2. Informative references

[RFC6112] Zhu, L., Leach, P., and S. Hartman, "Anonymity Support for Kerberos", RFC 6112, DOI 10.17487/RFC6112, April 2011, <http://www.rfc-editor.org/info/rfc6112>.

[RFC6113] Hartman, S. and L. Zhu, "A Generalized Framework for Kerberos Pre-Authentication", RFC 6113, DOI 10.17487/RFC6113, April 2011, <http://www.rfc-editor.org/info/rfc6113>.

[RFC6806] Hartman, S., Ed., Raeburn, K., and L. Zhu, "Kerberos Principal Name Canonicalization and Cross-Realm Referrals", RFC 6806, DOI 10.17487/RFC6806, November 2012, <http://www.rfc-editor.org/info/rfc6806>.

# Appendix A. Interoperate with Previous MIT version

MIT implemented an early draft version of this document. This section gives a method for detecting and interoperating with that version.

Initiators behave as follows:

- If the first acceptor token begins with generic token framing as described in section 3.1 of [RFC2743], then use the protocol as defined in this document.

- If the first acceptor token is missing the generic token framing (i.e., the token begins with the two-byte token ID 05 01), then

  * When creating the finish message, the value of one (1) should be used in place of GSS_EXTS_FINISHED.

  * When computing the checksum, the value of KEY_USAGE_IAKERB_FINISHED should be used in place of KEY_USAGE_FINISHED.

    KEY_USAGE_IAKERB_FINISHED 42

Acceptors behave as follows:

- After the first initiator token, allow initiator tokens to omit generic token framing. This allowance is required only for IAKERB_PROXY messages (those using token ID 05 01), not for tokens defined in [RFC4121].

- If the AP-REQ authenticator contains an extension of type 1 containing a KRB-FINISHED message, then process the extension as if it were of type GSS_EXTS_FINISHED, except with a key usage of KEY_USAGE_IAKERB_FINISHED (42) instead of KEY_USAGE_FINISHED (41).

## Authors' Addresses

Benjamin Kaduk (editor)
Akamai

Email: kaduk@mit.edu

Jim Schaad (editor)
Soaring Hawk Consulting

Email: ietf@augustcellars.com

Larry Zhu
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
US

Email: lzhu@microsoft.com

Jeffery Altman
Secure Endpoints
255 W 94th St
New York, NY 10025
US

Email: jaltman@secure-endpoints.com

Network Working Group
Internet-Draft
Updates: rfc4120 (if approved)
Intended status: Standards Track
Expires: October 1, 2017

T. Yu
MIT Kerberos Consortium

March 30, 2017

# Move Kerberos protocol parameter registries to IANA
# draft-ietf-kitten-kerberos-iana-registries-04

## Abstract

The Keberos 5 network authentication protocol has several numeric protocol parameters. Most of these parameters are not currently under IANA maintenance. This document requests that IANA take over the maintenance of the remainder of these Kerberos parameters.

## 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 October 1, 2017.

## Copyright Notice

Copyright (c) 2017 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.

## 1. 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].

## 2. Introduction

The Keberos 5 network authentication protocol[RFC4120][RFC1510] has several numeric protocol parameters. This document requests that IANA take over the maintenance of the Kerberos protocol parameters that are not currently under IANA maintenance. Several instances of number conflicts in Kerberos implementations could have been prevented by having IANA registries for those numbers. This document updates [RFC4120].

## 3. General registry format

Unless otherwise specified, each Kerberos protocol number registry will have the following fields: "number", "name", "reference", and "comments".

The name must begin with a lowercase letter, and must consist of ASCII letters, digits, and hyphens. Two or more hyphens must not appear directly adjacent to each other. A hyphen must not appear at the end of a name. It is preferred that words in a name be separated by hyphens, and that all of the letters be lowercase.

(These rules are consistent with the lexical rules for an ASN.1 valuereference or identifier. Where the constraints are stricter than the ASN.1 lexical rules, they make it easier to systematically transform the names for use in implementation languages.)

Names for numeric parameter values have no inherent meaning in the Kerberos protocol, but they can guide choices for internal implementation symbol names and for user-visible non-numeric representations. When written in English prose in specifications, or when used as symbolic constants in implementation languages (e.g., C preprocessor macros), it is common to transform the name into all uppercase letters, and possibly to replace hyphens with underscores.

## 4. General registration procedure

This document requests that the IESG establish a pool of Kerberos experts who will manage the Kerberos registries using these guidelines. The IESG may wish to consider including the set of designated IANA experts for existing Kerberos IANA registries as candidates for this pool.

IANA will select an expert from this pool for each registration request. The expert will review the registration request and may approve the registration, decline the registration with comments, or recommend that the registration request should follow a specific alternative process. The alternative processes that the expert may recommend are the IETF review process and the standards action process.

Initially, the expert reviewers will use a permissive process, generally approving registrations that are architecturally consistent with Kerberos and the protocol parameter in question. Over time, with input from the community, the experts may refine the requirements that registrations are expected to meet. The experts will maintain a current version of these guidelines in a manner that is generally accessible to the entire community. As the guidelines evolve, experts may consider the technical quality of specifications, security impacts of the registrations, architectural consistency, and interoperability impact. Experts may require a publicly available specification in order to make certain registrations.

[ For the individual registries, include "Registrations in this registry are managed by the expert review process [RFC5226] or in exceptional cases by IESG approval. See section x for guidelines for the experts to be used with this registry." ]

## 5. Integer assignments

Names for integer assignments must be unique across all Kerberos integer parameter registries. This is normally accomplished by including a name prefix that identifies the registry.

Assignments for integers parameters will follow the general registration procedure outlined above, except as otherwise noted in the section that contains the description of the parameter. Kerberos integer parameters take on signed 32-bit values (-2147483648 to 2147483647). Negative values are for private or local use.

### 5.1. Address types

Registry name: Address types

Assignment policy: General registration procedure

Valid values: Signed 32-bit integers

Address types historically align with numeric constants used in the Berkeley sockets API. Future address type assignments should conform

to this historical practice when possible. The name prefix for address types is "addrtype-".

### 5.2. Authorization data types

Registry name: Authorization data types

Assignment policy: General registration procedure

Valid values: Signed 32-bit integers

The name prefix for authorization data types is "ad-".

### 5.3. Error codes

Registry name: Error codes

Assignment policy: Standards action

Valid values: Signed 32-bit integers

Assignments for error codes require standards action due to their scarcity: assigning error codes greater than 127 could require significant changes to certain implementations. The name prefixes for error codes are "kdc-err-", "krb-err-", and "krb-ap-err-".

### 5.4. Key usages

Registry name: Key usages

Assignment policy: General registration procedure

Valid values: Unsigned 32-bit integers

Key usages are unsigned 32-bit integers (0 to 4294967295). Zero is reserved and may not be assigned.

The name prefix for key usages is "ku-".

### 5.5. Name types

Registry name: Name types

Assignment policy: General registration procedure

Valid values: Signed 32-bit integers

The name prefix for name types is "nt-".

| number | name | reference | comment |
|---|---|---|---|
| 0 | nt-unknown | RFC4120 | Name type not known |
| 1 | nt-principal | RFC4120 | Just the name of the principal as in DCE, or for users |
| 2 | nt-srv-inst | RFC4120 | Service and other unique instance (krbtgt) |
| 3 | nt-srv-hst | RFC4120 | Service with host name as instance (telnet, rcommands) |
| 4 | nt-srv-xhst | RFC4120 | Service with host as remaining components |
| 5 | nt-uid | RFC4120 | Unique ID |
| 6 | nt-x500-principal | RFC4120 | Encoded X.509 Distinguished name [RFC2253] |
| 7 | nt-smtp-name | RFC4120 | Name in form of SMTP email name (e.g., user@example.com) |
| 10 | nt-enterprise | RFC4120 | Enterprise name - may be mapped to principal name |
| 11 | nt-wellknown | RFC6111 | Well-known principal name |
| 12 | nt-srv-hst-domain | RFC5179 | Domain-based names |

## 5.6. Pre-authentication and typed data

Registry name: Pre-authentication and typed data

Assignment policy: General registration procedure

Valid values: Signed 32-bit integers

This document requests that IANA modify the existing Kerberos Pre-authentication and typed data registry to be consistent with the procedures in this document.

The name prefix for pre-authentication type numbers is "pa-". The name prefix for typed data numbers is "td-". Pre-authentication and typed data numbers are in the same registry, but a pre-authentication number may be also be assigned to a related typed data number.

## 6. Named bit assignments

Assignments for named bits require standards action, due to their scarcity: assigning bit numbers greater than 31 could require significant changes to implementations. Names for named bit assignments must be unique within a given named bit registry, and typically do not have name prefixes that identify which registry they belong to.

### 6.1. AP-REQ options

Registry name: AP-REQ options

Assignment policy: Standards action

Valid values: ASN.1 bit numbers 0 through 31

### 6.2. KDC-REQ options

Registry name: KDC-REQ options

Assignment policy: Standards action

Valid values: ASN.1 bit numbers 0 through 31

### 6.3. Ticket flags

Registry name: Ticket flags

Assignment policy: Standards action

Valid values: ASN.1 bit numbers 0 through 31

## 7. Numbers that will not be registered

ASN.1 application tag numbers (which are always equal to the "msg-type" field in Kerberos messages where they appear) will not be registered. Any Kerberos protocol change that requires a new application tag number will be a sufficiently major change that the specification of the change MUST define a new ASN.1 module and MUST be Standards Track.

Transited encoding values will not be registered. There is only one transited encoding type for the Kerberos protocol. The interoperability concerns inherent to the cross-realm operation of Kerberos mean that specifications of new transited encoding types are very unlikely. Any specification of new transited encoding types MUST be Standards Action.

Protocol version number (pvno) values will not be registered. The location of the "pvno" value in Kerberos messages is not in a place that implementations can meaningfully use to distinguish among different variants of the Kerberos protocol.

## 8. Contributors

Sam Hartman proposed the text of the expert review guidelines. Love Hornquist Astrand wrote a previous document (draft-lha-krb-wg-some-numbers-to-iana-00) with the same goals as this document.

## 9. Acknowledgments

Thanks to Tom Petch for providing useful feedback on previous versions of this document.

## 10. Security Considerations

Assignments of new Keberos protocol parameter values can have security implications. In cases where the assignment policy calls for expert review, the reviewer is responsible for evaluating whether adequate documentation exists concerning the security considerations for the requested assignment. For assignments that require IETF review or standards action, the normal IETF processes ensure adequate treatment of security considerations.

## 11. IANA Considerations

This document requests that IANA create several registries for Kebreros protocol parameters:

- Address types
- Authorization data types
- Error codes
- Key usages
- Name types
- AP-REQ options
- KDC-REQ options
- Ticket flags

This document requests that IANA modify the existing "Pre-authentication data and typed data" registry to contain an additional reference to this document, and to transform existing names in that registry to the lowercase-and-hyphens style.

## 12. Open issues

Do we make a registry for application tag numbers (equal to message type numbers)? We've said that we would replace the entire ASN.1 module in that case, but Nico's recent proposal doesn't do that, and if we want to accommodate that sort of proposal, it would probably be best to establish a registry. (It should require standards action for registrations.)

Do transited encodings need a registry? They would probably require standards action, even if there were a registry.

## 13. References

### 13.1. 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, <http://www.rfc-editor.org/info/rfc2119>.

[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>.

[RFC4120] Neuman, C., Yu, T., Hartman, S., and K. Raeburn, "The Kerberos Network Authentication Service (V5)", RFC 4120, DOI 10.17487/RFC4120, July 2005, <http://www.rfc-editor.org/info/rfc4120>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

### 13.2. Informative References

[RFC1510] Kohl, J. and C. Neuman, "The Kerberos Network Authentication Service (V5)", RFC 1510, DOI 10.17487/RFC1510, September 1993, <http://www.rfc-editor.org/info/rfc1510>.

Author's Address

Tom Yu
MIT Kerberos Consortium
77 Massachusetts Ave
Cambridge, Massachusetts
USA

Email: tlyu@mit.edu

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 11, 2021

S. Cantor
Shibboleth Consortium
M. Cullen
Painless Security
S. Josefsson
SJD AB

May 10, 2021

# SAML Enhanced Client SASL and GSS-API Mechanisms
# draft-ietf-kitten-sasl-saml-ec-20

## Abstract

Security Assertion Markup Language (SAML) 2.0 is a generalized framework for the exchange of security-related information between asserting and relying parties. Simple Authentication and Security Layer (SASL) and the Generic Security Service Application Program Interface (GSS-API) are application frameworks that facilitate an extensible authentication model, among other things. This document specifies a SASL and GSS-API mechanism for SAML 2.0 that leverages the capabilities of a SAML-aware "enhanced client" to address significant barriers to federated authentication in a manner that encourages reuse of existing SAML bindings and profiles designed for non-browser scenarios.

## 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 11, 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 . . . . . 3
2. Terminology . . . . . 5
3. Applicability for Non-HTTP Use Cases . . . . . 6
4. SAML Enhanced Client SASL Mechanism Specification . . . . . 8
4.1. Advertisement . . . . . 8
4.2. Initiation . . . . . 9
4.3. Server Response . . . . . 9
4.4. User Authentication with Identity Provider . . . . . 10
4.5. Client Response . . . . . 10
4.6. Outcome . . . . . 10
4.7. Additional Notes . . . . . 10
5. SAML EC GSS-API Mechanism Specification . . . . . 11
5.1. GSS-API Credential Delegation . . . . . 12
5.2. GSS-API Channel Binding . . . . . 13
5.3. Session Key Derivation . . . . . 13
5.3.1. Generated by Identity Provider . . . . . 14
5.3.2. Alternate Key Derivation Mechanisms . . . . . 15
5.4. Per-Message Tokens . . . . . 15
5.5. Pseudo-Random Function (PRF) . . . . . 15
5.6. GSS-API Principal Name Types for SAML EC . . . . . 16
5.6.1. User Naming Considerations . . . . . 16
5.6.2. Service Naming Considerations . . . . . 17
6. Example . . . . . 17
7. Security Considerations . . . . . 25
7.1. Risks Left Unaddressed . . . . . 26
7.2. User Privacy . . . . . 26
7.3. Collusion between RPs . . . . . 27
8. IANA Considerations . . . . . 27
8.1. GSS-API and SASL Mechanism Registration . . . . . 27
8.2. XML Namespace Name for SAML-EC . . . . . 27
9. References . . . . . 28
9.1. Normative References . . . . . 28
9.2. Informative References . . . . . 30
Appendix A. XML Schema . . . . . 31
Appendix B. Acknowledgments . . . . . 33
Appendix C. Changes . . . . . 33

Authors' Addresses . . . . . 34

## 1. Introduction

Security Assertion Markup Language (SAML) 2.0 [OASIS.saml-core-2.0-os] is a modular specification that provides various means for a user to be identified to a relying party (RP) through the exchange of (typically signed) assertions issued by an identity provider (IdP).

Simple Authentication and Security Layer (SASL) [RFC4422] is a generalized mechanism for identifying and authenticating a user and for optionally negotiating a security layer for subsequent protocol interactions. SASL is used by application protocols like IMAP [RFC3501], the Post Office Protocol(POP [RFC1939]) and XMPP [RFC6120]. The effect of SASL is to make authentication modular, so that newer authentication mechanisms can be added as needed.

There are related protocols, protocol bindings [OASIS.saml-bindings-2.0-os], and interoperability profiles [OASIS.saml-profiles-2.0-os] designed for different use cases. Additional profiles and extensions are also routinely developed and published.

The Generic Security Service Application Program Interface (GSS-API) [RFC2743] provides a framework for applications to support multiple authentication mechanisms through a unified programming interface, as well as additional optional cryptographic functionality. This document defines a pure SASL mechanism for SAML, but it conforms to the bridge between SASL and GSS-API called GS2 [RFC5801]. This means that this document defines both a SASL mechanism and a GSS-API mechanism. The GSS-API interface is optional for SASL implementers, and the GSS-API considerations can be avoided in environments that use SASL directly without GSS-API.

The mechanisms specified in this document allow a SASL- or GSS-API-enabled server to act as a SAML relying party, or service provider (SP), by advertising this mechanism as an option for SASL or GSS-API clients that support the use of SAML to communicate identity and attribute information. Clients supporting this mechanism are termed "enhanced clients" in SAML terminology because they understand the federated authentication model and have specific knowledge of the IdP(s) associated with the user. This knowledge, and the ability to act on it, addresses a significant problem with browser-based SAML profiles known as the "discovery", or "where are you from?" (WAYF) problem. In a "dumb" client such as a web browser, various intrusive user interface techniques are used to determine the appropriate IdP to use because the request to the IdP is generated as an HTTP

redirect by the RP, which does not generally have prior knowledge of the IdP to use. Obviating the need for the RP to interact with the client to determine the right IdP (and its network location) is both a user interface and security improvement.

The SAML mechanism described in this document is an adaptation of an existing SAML profile, the Enhanced Client or Proxy (ECP) Profile (V2.0) [SAMLECP20].

Figure 1 describes the interworking between SAML and SASL: this document requires enhancements to the RP and to the client (as the two SASL communication endpoints) but no changes to the SAML IdP are assumed apart from its support for the applicable SAML profile. To accomplish this, a SAML protocol exchange between the RP and the IdP, brokered by the client, is tunneled within SASL. There is no assumed communication between the RP and the IdP, but such communication may occur in conjunction with additional SAML-related profiles not in scope for this document.

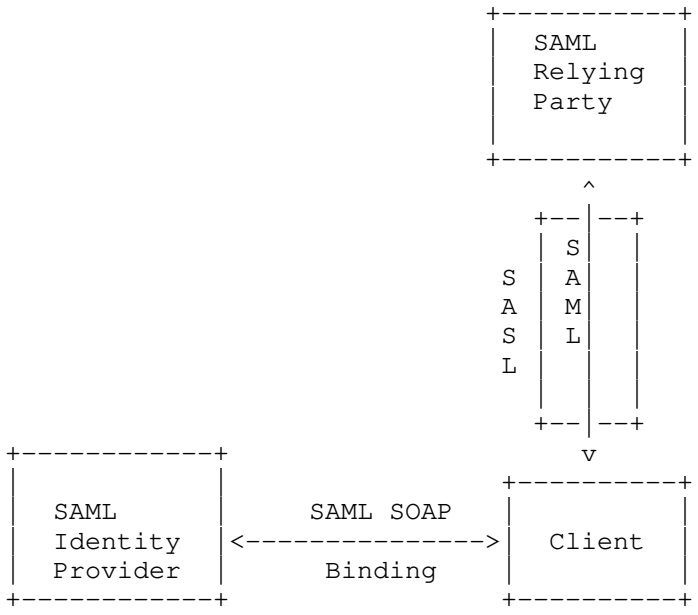



Figure 1: Interworking Architecture

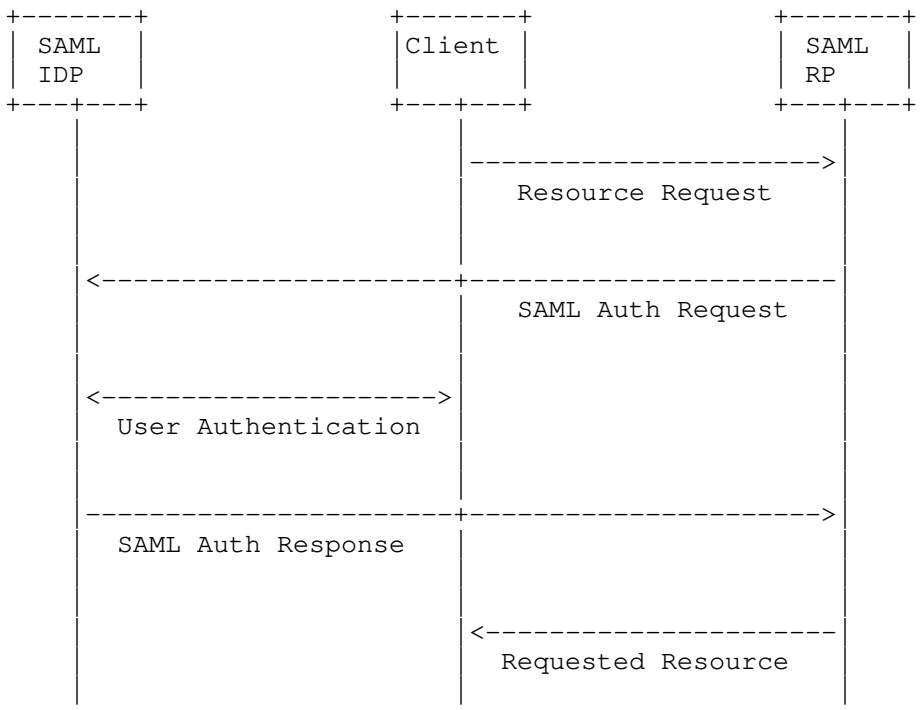



Figure 2: Communication Flow

## 2. 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.

The reader is also assumed to be familiar with the terms used in the SAML 2.0 specification, and an understanding of the Enhanced Client or Proxy (ECP) Profile (V2.0) [SAMLECP20] is necessary, as part of this mechanism explicitly reuses and references it.

This document can be implemented without knowledge of GSS-API since the normative aspects of the GS2 protocol syntax have been duplicated in this document. The document may also be implemented to provide a GSS-API mechanism, and then knowledge of GSS-API is essential.

## 3. Applicability for Non-HTTP Use Cases

While SAML is designed to support a variety of application scenarios, the profiles for authentication defined in the original standard are designed around HTTP [RFC7230] applications. They are not, however, limited to browsers, because browsers do not always meet the needs of more security-sensitive applications. Specifically, the notion of an "Enhanced Client" (or a proxy acting as one on behalf of a browser, thus the term "ECP") was specified for a software component that acts somewhat like a browser from an application perspective, but includes limited, but sufficient, awareness of SAML to play a more conscious role in the authentication exchange between the RP and the IdP. What follows is an outline of the Enhanced Client or Proxy (ECP) Profile (V2.0) [SAMLECP20], as applied to the web/HTTP service use case:

1. The Enhanced Client requests a resource of a Relying Party (RP) (via an HTTP request). In doing so, it advertises its "enhanced" capability using HTTP headers.

2. The RP, desiring SAML authentication and noting the client's capabilities, responds not with an HTTP redirect or form, but with a SOAP [W3C.soap11] envelope containing a SAML <AuthnRequest> along with some supporting headers. This request identifies the RP (and may be signed), and may provide hints to the client as to what IdPs the RP finds acceptable, but the choice of IdP is generally left to the client.

3. The client is then responsible for delivering the body of the SOAP message to the IdP it is instructed to use (often via out-of-band configuration). The user authenticates to the IdP ahead of, during, or after the delivery of this message, and perhaps explicitly authorizes the response to the RP.

4. Whether authentication succeeds or fails, the IdP responds with its own SOAP envelope, generally containing a SAML <Response> message for delivery to the RP. In a successful case, the message will include one or more SAML <Assertion> elements containing authentication, and possibly attribute, statements about the subject. Either the response or each assertion is signed, and the assertion(s) may be encrypted to a key negotiated with or known to belong to the RP.

5. The client then delivers the SOAP envelope containing the <Response> to the RP at a location the IdP directs (which acts as an additional, though limited, defense against MITM attacks). This completes the SAML exchange.

6. The RP now has sufficient identity information to approve the original HTTP request or not, and acts accordingly. Everything between the original request and this response can be thought of as an "interruption" of the original HTTP exchange.

When considering this flow in the context of an arbitrary application protocol and SASL, the RP and the client both must change their code to implement this SASL mechanism, but the IdP can remain unmodified. The existing RP/client exchange that is tunneled through HTTP maps well to the tunneling of that same exchange in SASL. In the parlance of SASL [RFC4422], this mechanism is "client-first" for consistency with GS2. The steps are shown below:

1. The server MAY advertise the SAML20EC and/or SAML20EC-PLUS mechanisms.

2. The client initiates a SASL authentication with SAML20EC or SAML20EC-PLUS.

3. The server sends the client a challenge consisting of a SOAP envelope containing its SAML <AuthnRequest>.

4. The SASL client unpacks the SOAP message and communicates with its chosen IdP to relay the SAML <AuthnRequest> to it. This communication, and the authentication with the IdP, proceeds separately from the SASL process.

5. Upon completion of the exchange with the IdP, the client responds to the SASL server with a SOAP envelope containing the SAML <Response> it obtained, or a SOAP fault, as warranted.

6. The SASL Server indicates success or failure.

Note: The details of the SAML processing, which are consistent with the Enhanced Client or Proxy (ECP) Profile (V2.0) [SAMLECP20], are such that the client MUST interact with the IdP in order to complete any SASL exchange with the RP. The assertions issued by the IdP for the purposes of the profile, and by extension this SASL mechanism, are short lived, and therefore cannot be cached by the client for later use.

Encompassed in step four is the client-driven selection of the IdP, authentication to it, and the acquisition of a response to provide to the SASL server. These processes are all external to SASL.

Note also that unlike an HTTP-based profile, the IdP cannot participate in the selection of, or evaluation of, the location to which the SASL Client Response will be delivered by the client. The

use of GSS-API Channel Binding is an important mitigation of the risk of a "Man in the Middle" attack between the client and RP, as is the use of a negotiated or derived session key in whatever protocol is secured by this mechanism.

With all of this in mind, the typical flow appears as follows:

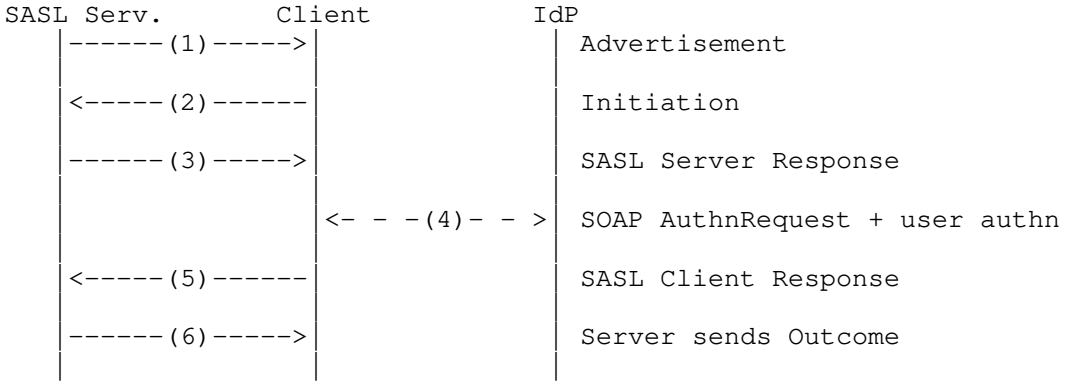



Figure 3: Authentication flow

## 4. SAML Enhanced Client SASL Mechanism Specification

Based on the previous figures, the following operations are defined by the SAML SASL mechanism:

### 4.1. Advertisement

To advertise that a server supports this mechanism, during application session initiation, it displays the name "SAML20EC" and/or "SAML20EC-PLUS" in the list of supported SASL mechanisms.

In accordance with [RFC5801] the "-PLUS" variant indicates that the server supports channel binding and would be selected by a client with that capability.

### 4.2. Initiation

A client initiates "SAML20EC" or "SAML20EC-PLUS" authentication. If supported by the application protocol, the client MAY include an initial response, otherwise it waits until the server has issued an empty challenge (because the mechanism is client-first).

The format of the initial client response ("initresp") is as follows:

```
hok = "urn:oasis:names:tc:SAML:2.0:cm:holder-of-key"

mut = "urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp:2.0:" \
    "WantAuthnRequestsSigned"

del = "urn:oasis:names:tc:SAML:2.0:conditions:delegation"

initresp = gs2-cb-flag "," [gs2-authzid] "," [hok] "," [mut] "," [del]
```

The gs2-cb-flag flag MUST be set as defined in [RFC5801] to indicate whether the client supports channel binding. This takes the place of the PAOS HTTP header extension used in [SAMLECP20] to indicate channel binding support.

The optional "gs2-authzid" field holds the authorization identity, as requested by the client.

The optional "hok" field is a constant that signals the client's support for stronger security by means of a locally held key. This takes the place of the PAOS HTTP header extension used in [SAMLECP20] to indicate "holder of key" support.

The optional "mut" field is a constant that signals the client's desire for mutual authentication. If set, the SASL server MUST digitally sign its SAML <AuthnRequest> message. The URN constant above is a single string; the linefeed is shown for RFC formatting reasons.

The optional "del" field is a constant that signals the client's desire for the acceptor to request an assertion usable for delegation of the client's identity to the acceptor.

### 4.3. Server Response

The SASL server responds with a SOAP envelope constructed in accordance with section 2.3.2 of [SAMLECP20]. This includes adhering to the SOAP header requirements of the SAML PAOS Binding [OASIS.saml-bindings-2.0-os], for compatibility with the existing

profile. Various SOAP headers are also consumed by the client in exactly the same manner prescribed by that section.

### 4.4. User Authentication with Identity Provider

Upon receipt of the Server Response (Section 4.3), the steps described in sections 2.3.3 through 2.3.6 of [SAMLECP20] are performed between the client and the chosen IdP. The means by which the client determines the IdP to use, and where it is located, are out of scope of this mechanism.

The exact means of authentication to the IdP are also out of scope, but clients supporting this mechanism MUST support HTTP Basic Authentication as defined in [RFC7617] and TLS 1.3 client authentication as defined in [RFC8446].

### 4.5. Client Response

Assuming a response is obtained from the IdP, the client responds to the SASL server with a SOAP envelope constructed in accordance with section 2.3.7 of [SAMLECP20]. This includes adhering to the SOAP header requirements of the SAML PAOS Binding [OASIS.saml-bindings-2.0-os], for compatibility with the existing profile. If the client is unable to obtain a response from the IdP, or must otherwise signal failure, it responds to the SASL server with a SOAP envelope containing a SOAP fault.

### 4.6. Outcome

The SAML protocol exchange having completed, the SASL server will transmit the outcome to the client depending on local validation of the client responses (including the assertion conveyed from the chosen IDP). This outcome is transmitted in accordance with the application protocol in use.

### 4.7. Additional Notes

Because this mechanism is an adaptation of an HTTP-based profile, there are a few requirements outlined in [SAMLECP20] that make reference to a response URL that is normally used to regulate where the client returns information to the RP. There are also security-related checks built into the profile that involve this location.

For compatibility with existing IdP and profile behavior, and to provide for mutual authentication, the SASL server MUST populate the responseConsumerURL and AssertionConsumerServiceURL attributes with its service name. As discussed in Section 5.6.2, most SASL profiles rely on a service name format of "service@host", but regardless of

the form, the service name is used directly rather than transformed into an absolute URI if it is not already one, and MUST be percent-encoded per [RFC3986].

The IdP MUST securely associate the service name with the SAML entityID claimed by the SASL server, such as through the use of SAML metadata [OASIS.saml-metadata-2.0-os]. If metadata is used, a SASL service's <SPSSODescriptor> role MUST contain a corresponding <AssertionConsumerService> whose Location attribute contains the appropriate service name, as described above. The Binding attribute MUST be one of "urn:ietf:params:xml:ns:samlec" (RECOMMENDED) or "urn:oasis:names:tc:SAML:2.0:bindings:PAOS" (for compatibility with older implementations of the ECP profile in existing IdP software).

Finally, note that the use of HTTP status signaling between the RP and client mandated by [SAMLECP20] may not be applicable.

## 5. SAML EC GSS-API Mechanism Specification

This section and its sub-sections and all normative references of it not referenced elsewhere in this document are INFORMATIONAL for SASL implementors, but they are NORMATIVE for GSS-API implementors.

The SAML Enhanced Client SASL mechanism is also a GSS-API mechanism. The messages are the same, but a) the GS2 [RFC5801] header on the client's first authentication message is excluded when SAML EC is used as a GSS-API mechanism, and b) the [RFC2743] section 3.1 initial context token header is used for the client's first authentication message (context token) instead, with the body of the message being the same as for the SASL mechanism case.

The GSS-API mechanism OID for SAML EC is OID-TBD (IANA to assign: see IANA considerations). The DER encoding of the OID is TBD.

The mutual_state request flag (GSS_C_MUTUAL_FLAG) MAY be set to TRUE, resulting in the "mut" option set in the initial client response. The security context mutual_state flag is set to TRUE only if the server digitally signs its SAML <AuthnRequest> message and the signature and signing credential are appropriately verified by the IdP. The IdP signals this to the client in an <ecp:RequestAuthenticated> SOAP header block.

The lifetime of a security context established with this mechanism SHOULD be limited by the value of a SessionNotOnOrAfter attribute, if any, in the <AuthnStatement> element(s) of the SAML assertion(s) received by the RP. By convention, in the rare case that multiple valid/confirmed assertions containing <AuthnStatement> elements are

received, the most restrictive SessionNotOnOrAfter is generally applied.

### 5.1. GSS-API Credential Delegation

This mechanism can support credential delegation through the issuance of SAML assertions that an IdP will accept as proof of authentication by a service on behalf of a subject. An initiator may request delegation of its credentials by setting the "del" option field in the initial client response to "urn:oasis:names:tc:SAML:2.0:conditions:delegation".

An acceptor, upon receipt of this constant, requests a delegated assertion by including in its <AuthnRequest> message a <Conditions> element containing an <AudienceRestriction> identifying the IdP as a desired audience for the assertion(s) to be issued. In the event that the specific IdP to be used is unknown, the constant "urn:oasis:names:tc:SAML:2.0:conditions:delegation" may be used as a stand-in, per Section 2.3.2 of [SAMLECP20].

Upon receipt of an assertion satisfying this property, and containing a <SubjectConfirmation> element that the acceptor can satisfy, the security context will have its deleg_state flag (GSS_C_DELEG_FLAG) set to TRUE.

The IdP, if it issues a delegated assertion to the acceptor, MUST include in the SOAP response to the initiator a <samlec:Delegated> SOAP header block, indicating that delegation was enabled. It has no content, other than mandatory SOAP attributes (an example follows):

```
<samlec:Delegated xmlns:samlec="urn:ietf:params:xml:ns:samlec"
    xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
    S:mustUnderstand="1"
    S:actor="http://schemas.xmlsoap.org/soap/actor/next" />
```

Upon receipt of such a header block, the initiator MUST fail the establishment of the security context if it did not request delegation in its initial client response to the acceptor. It SHOULD signal this failure to the acceptor with a SOAP fault message in its final client response.

As noted previously, the exact means of client authentication to the IdP is formally out of scope of this mechanism. This extends to the use of a delegation assertion as a means of authentication by an acceptor acting as an initiator. In practice, some profile of

[WSS-SAML] is used to attach the assertion and a confirmation proof to the SOAP message from the client to the IdP.

### 5.2. GSS-API Channel Binding

GSS-API channel binding [RFC5554] is a protected facility for exchanging a cryptographic identifier for an enclosing channel between the initiator and acceptor. The initiator sends channel binding data and the acceptor confirms that channel binding data has been checked.

The acceptor SHOULD accept any channel binding provided by the initiator if null channel bindings are passed into gss_accept_sec_context. Protocols such as HTTP Negotiate [RFC4559] depend on this behavior of some Kerberos implementations.

The exchange and verification of channel binding information is described by [SAMLECP20].

### 5.3. Session Key Derivation

Some GSS-API features (discussed in the following sections) require a session key be established as a result of security context establishment. In the common case of a "bearer" assertion in SAML, a mechanism is defined to communicate a key to both parties via the IdP. In other cases such as assertions based on "holder of key" confirmation bound to a client-controlled key, there may be additional methods defined in the future, and extension points are provided for this purpose.

Information defining or describing the session key, or a process for deriving one, is communicated between the initiator and acceptor using a <samlec:SessionKey> element, defined by the XML schema in Appendix A. This element is a SOAP header block. The content of the element further depends on the specific use in the mechanism. The Algorithm XML attribute identifies a mechanism for key derivation. It is omitted to identify the use of an IdP-generated key (see following section) or will contain a URI value identifying a derivation mechanism defined outside this specification. Each header block's mustUnderstand and actor attributes MUST be set to "1" and "http://schemas.xmlsoap.org/soap/actor/next" respectively.

In the acceptor's first response message containing its SAML request, one or more <samlec:SessionKey> SOAP header blocks MUST be included. The element MUST contain one or more <EncType> elements containing the number of a supported encryption type defined in accordance with [RFC3961]. Encryption types should be provided in order of preference by the acceptor.

In the final client response message, a single <samlec:SessionKey> SOAP header block MUST be included. A single <EncType> element MUST be included to identify the chosen encryption type used by the initiator.

All parties MUST support the "aes128-cts-hmac-sha1-96" encryption type, number 17, defined by [RFC3962].

Further details depend on the mechanism used, one of which is described in the following section.

#### 5.3.1. Generated by Identity Provider

The IdP, if issuing a bearer assertion for use with this mechanism, SHOULD provide a generated key for use by the initiator and acceptor. This key is used as pseudorandom input to the "random-to-key" function for a specific encryption type defined in accordance with [RFC3961]. The key is base64-encoded and placed inside a <samlec:GeneratedKey> element. The IdP does not participate in the selection of the encryption type and simply generates enough pseudorandom bits to supply key material to the other parties.

The resulting <samlec:GeneratedKey> element is placed within the <saml:Advice> element of the assertion issued. The identity provider MUST encrypt the assertion (implying that it MUST have the means to do so, typically knowledge of a key associated with the RP). If multiple assertions are issued (allowed, but not typical), the element need only be included in one of the assertions issued for use by the relying party.

A copy of the element is also added as a SOAP header block in the response from the IdP to the client (and then removed when constructing the response to the acceptor).

If this mechanism is used by the initiator, then the <samlec:SessionKey> SOAP header block attached to the final client response message will identify this via the omission of the Algorithm attribute and will identify the chosen encryption type using the <samlec:EncType> element:

```
<samlec:SessionKey xmlns:samlec="urn:ietf:params:xml:ns:samlec"
    xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
    S:mustUnderstand="1"
    S:actor="http://schemas.xmlsoap.org/soap/actor/next">
  <samlec:EncType>17</samlec:EncType>
<samlec:SessionKey>
```

Both the initiator and acceptor MUST execute the chosen encryption type's random-to-key function over the pseudorandom value provided by the <samlec:GeneratedKey> element. The result of that function is used as the protocol and session key. Support for subkeys from the initiator or acceptor is not specified.

#### 5.3.2. Alternate Key Derivation Mechanisms

In the event that a client is proving possession of a secret or private key, a formal key agreement algorithm might be supported. This specification does not define such a mechanism, but the <samlec:SessionKey> element is extensible to allow for future work in this space by means of the Algorithm attribute and an optional <ds:KeyInfo> child element to carry extensible content related to key establishment.

However a key is derived, the <samlec:EncType> element will identify the chosen encrytion type, and both the initiator and acceptor MUST execute the encryption type's random-to-key function over the result of the key agreement or derivation process. The result of that function is used as the protocol key.

### 5.4. Per-Message Tokens

The per-message tokens SHALL be the same as those for the Kerberos V5 GSS-API mechanism [RFC4121] (see Section 4.2 and sub-sections).

The replay_det_state (GSS_C_REPLAY_FLAG), sequence_state (GSS_C_SEQUENCE_FLAG), conf_avail (GSS_C_CONF_FLAG) and integ_avail (GSS_C_INTEG_FLAG) security context flags are always set to TRUE.

The "protocol key" SHALL be a key established in a manner described in the previous section. "Specific keys" are then derived as usual as described in Section 2 of [RFC4121], [RFC3961], and [RFC3962].

The terms "protocol key" and "specific key" are Kerberos V5 terms [RFC3961].

SAML20EC is PROT_READY as soon as the SAML response message has been seen.

### 5.5. Pseudo-Random Function (PRF)

The GSS-API has been extended with a Pseudo-Random Function (PRF) interface in [RFC4401]. The purpose is to enable applications to derive a cryptographic key from an established GSS-API security context. This section defines a GSS_Pseudo_random that is applicable for the SAML20EC GSS-API mechanism.

The GSS_Pseudo_random() [RFC4401] SHALL be the same as for the Kerberos V5 GSS-API mechanism [RFC7802]. There is no acceptor-asserted sub-session key, thus GSS_C_PRF_KEY_FULL and GSS_C_PRF_KEY_PARTIAL are equivalent. The protocol key to be used for the GSS_Pseudo_random() SHALL be the same as the key defined in the previous section.

### 5.6. GSS-API Principal Name Types for SAML EC

Services that act as SAML relying parties are typically identified by means of a URI called an "entityID". Clients that are named in the <Subject> element of a SAML assertion are typically identified by means of a <NameID> element, which is an extensible XML structure containing, at minimum, an element value that names the subject and a Format attribute.

In practice, a GSS-API client and server are unlikely to know in advance the name of the initiator as it will be expressed by the SAML IdP upon completion of authentication. It is also generally incorrect to assume that a particular acceptor name will directly map into a particular RP entityID, because there is often a layer of naming indirection between particular services on hosts and the identity of a relying party in SAML terms.

To avoid complexity, and avoid unnecessary use of XML within the naming layer, the SAML EC mechanism relies on the common/expected name types used for acceptors and initiators, GSS_C_NT_HOSTBASED_SERVICE and GSS_C_NT_USER_NAME. The mechanism provides for validation of the host-based service name in conjunction with the SAML exchange. It does not attempt to solve the problem of mapping between an initiator "username", the user's identity while authenticating to the IdP, and the information supplied by the IdP to the acceptor. These relationships must be managed through local policy at the initiator and acceptor.

SAML-based information associated with the initiator SHOULD be expressed to the acceptor using GSS-API naming extensions [RFC6680], in a similar manner to [RFC7056].

#### 5.6.1. User Naming Considerations

The GSS_C_NT_USER_NAME form represents the name of an individual user. Clients often rely on this value to determine the appropriate credentials to use in authenticating to the IdP, and supply it to the server for use by the acceptor.

Upon successful completion of this mechanism, the server MUST construct the authenticated initiator name based on the <saml:NameID>

element in the assertion it successfully validated. The name is constructed as a UTF-8 string in the following form:

```
name = element-value "!" Format  "!" NameQualifier
    "!" SPNameQualifier "!" SPProvidedID
```

The "element-value" token refers to the content of the <saml:NameID> element. The other tokens refer to the identically named XML attributes defined for use with the element. If an attribute is not present, which is common, it is omitted (i.e., replaced with the empty string). The Format value is never omitted; if not present, the SAML-equivalent value of "urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified" is used.

Not all SAML assertions contain a <saml:NameID> element. In the event that no such element is present, including the exceptional cases of a <saml:BaseID> element or a <saml:EncryptedID> element that cannot be decrypted, the GSS_C_NT_ANONYMOUS name type MUST be used for the initiator name.

As noted in the previous section, it is expected that most applications able to rely on SAML authentication would make use of naming extensions to obtain additional information about the user based on the assertion. This is particularly true in the anonymous case, or in cases in which the SAML name is pseudonymous or transient in nature. The ability to express the SAML name in GSS_C_NT_USER_NAME form is intended for compatibility with applications that cannot make use of additional information.

### 5.6.2. Service Naming Considerations

The GSS_C_NT_HOSTBASED_SERVICE name form represents a service running on a host; it is textually represented as "service@host". This name form is required by most SASL profiles and is used by many existing applications that use the Kerberos GSS-API mechanism. As described in in the SASL mechanism's Section 4.7, such a name is used directly by this mechanism as the effective AssertionConsumerService "location" associated with the service and applied in IdP verification of the request against the claimed SAML entityID.

## 6. Example

Suppose the user has an identity at the SAML IdP saml.example.org and a Jabber Identifier (jid) "somenode@example.com", and wishes to authenticate his XMPP connection to xmpp.example.com (and example.com and example.org have established a SAML-capable trust relationship). The authentication on the wire would then look something like the following:

Step 1: Client initiates stream to server:

```
<stream:stream xmlns='jabber:client'
xmlns:stream='http://etherx.jabber.org/streams'
to='example.com' version='1.0'>
```

Step 2: Server responds with a stream tag sent to client:

```
<stream:stream
xmlns='jabber:client' xmlns:stream='http://etherx.jabber.org/streams'
id='some_id' from='example.com' version='1.0'>
```

Step 3: Server informs client of available authentication mechanisms:

```
<stream:features>
 <mechanisms xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
  <mechanism>DIGEST-MD5</mechanism>
  <mechanism>PLAIN</mechanism>
  <mechanism>SAML20EC</mechanism>
 </mechanisms>
</stream:features>
```

Step 4: Client selects an authentication mechanism and sends the initial client response (it is base64 encoded as specified by the XMPP SASL protocol profile):

```
<auth xmlns='urn:ietf:params:xml:ns:xmpp-sasl' mechanism='SAML20EC'>
biwsLCw=
</auth>
```

The initial response is "n,,,," which signals that channel binding is not used, there is no authorization identity, and the client does not support key-based confirmation, or want mutual authentication or delegation.

Step 5: Server sends a challenge to client in the form of a SOAP envelope containing its SAML <AuthnRequest>:

```
<challenge xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
PFM6RW52ZWxvcGUKICAgIHhtbG5zOnNhbWw9InVybjpvYXNpczpuYW1lczp0YzpT
QU1MOjIuMDphc3NlcnRpb24iCiAgICB4bWxuczpzYW1scD0idXJuOm9hc2lzOm5h
bWVzOnRjOlNBTUw6Mi4wOnByb3RvY29sIgogICAgeG1sbnM6Uz0iaHR0cDovL3Nj
aGVtYXMueG1sc29hcC5vcmcvc29hcC9lbnZlbG9wZS8iPgogIDxTOkhlYWRlcj4K
ICAgIDxwYW9zOlJlcXVlc3QgeG1sbnM6cGFvcz0idXJuOmxpYmVydHk6cGFvczoy
MDAzLTA4IgogICAgICBtZXNzYWdlSUQ9ImMzYTRmOGI5YzJkIiBTOm11c3RVbmRl
cnN0YW5kPSIxIgogICAgICBTOmFjdG9yPSJodHRwOi8vc2NoZW1hcy54bWxzb2Fw
Lm9yZy9zb2FwL2FjdG9yL25leHQiCiAgICAgIHJlc3BvbnNlQ29uc3VtZXJVUkw9
InhtcHBAeG1wcC5leGFtcGxlLmNvbSIKICAgICAgc2VydmljZT0idXJuOm9hc2lz
Om5hbWVzOnRjOlNBTUw6Mi4wOnByb2ZpbGVzOlNTTzplY3AiLz4KICAgIDxlY3A6
UmVxdWVzdAogICAgICB4bWxuczplY3A9InVybjpvYXNpczpuYW1lczp0YzpTQU1M
OjIuMDpwcm9maWxlczpTU086ZWNwIgogICAgICBTOmFjdG9yPSJodHRwOi8vc2No
ZW1hcy54bWxzb2FwLm9yZy9zb2FwL2FjdG9yL25leHQiCiAgICAgIFM6MbXVzdFVu
ZGVyc3RhbmQ9IjEiIFByb3ZpZGVyTmFtZT0iSmFiYmVyIGF0IGV4YW1wbGUuY29t
Ij4KICAgICAgPHNhbWw6SXNzdWVyPmh0dHBzOi8veG1wcC5leGFtcGxlLmNvbTwv
c2FtbDpJc3N1ZXI+CiAgICA8L2VjcDpSZXF1ZXN0PgogICAgPFM6bVzdFVu
aW9uS2V5IHhtbG5zOnNhbWxlYz0idXJuOmlldGY6cGFyYW1zOnhtbDpuczpzYW1s
ZWMiCiAgICAgIHhtbG5zOlM9Imh0dHA6Ly9zY2hlbWFzLnhtbHNvYXAub3JnL3Nv
YXAvZW52ZWxvcGUvIgogICAgICBTOm11c3RVbmRlcnN0YW5kPSIxIgogICAgICBT
OmFjdG9yPSJodHRwOi8vc2NoZW1hcy54bWxzb2FwLm9yZy9zb2FwL2FjdG9yL25l
eHQiPgogICAgICA8c2FtbGVjOkVuY1R5cGU+MTc8L3NhbWxlYzpFbmNUeXBlPgog
ICAgICA8c2FtbGVjOkVuY1R5cGU+MTg8L3NhbWxlYzpFbmNUeXBlPgogICAgPHNh
bWxlYzpTZXNzaW9uS2V5PgogIDwvUzpFNTDoyLjA6bmFtZWlkLWZvcm1hdDpwZXJzaXN0
ZW50Ii8+CiAgICAgIDxzYW1scDpSZXF1ZXN0ZWRBdXRobkNvbnRleHQgQ29tcGFy
aXNvbj0iZXhhY3QiPgogICAgICAgPHNhbWw6QXV0aG5Db250ZXh0Q2xhc3NNSZWY+
CiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAg
ICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAg
YXNzd29yZFByb3RlY3RlZFRyYW5zcG9ydAogICAgICAgPC9zYW1sOkF1dGhuQ29u
dGV4dENsYXNzUmVmPgogICAgICA8L3NhbWxwOlJlcXVlc3RlZEF1dGhuQ29udGV4
dD4gCiAgICA8L3NhbWxwOkF1dGhuUmVxdWVzdD4KPC9TOkVu
dmVsb3BlPgo=
</challenge>
```

The Base64 [RFC4648] decoded envelope:

```
<S:Envelope
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
    xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Header>
    <paos:Request xmlns:paos="urn:liberty:paos:2003-08"
      messageID="c3a4f8b9c2d" S:mustUnderstand="1"
      S:actor="http://schemas.xmlsoap.org/soap/actor/next"
      responseConsumerURL="xmpp@xmpp.example.com"
      service="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"/>
    <ecp:Request
      xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
      S:actor="http://schemas.xmlsoap.org/soap/actor/next"
      S:mustUnderstand="1" ProviderName="Jabber at example.com">
      <saml:Issuer>https://xmpp.example.com</saml:Issuer>
    </ecp:Request>
    <samlec:SessionKey xmlns:samlec="urn:ietf:params:xml:ns:samlec"
      xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
      S:mustUnderstand="1"
      S:actor="http://schemas.xmlsoap.org/soap/actor/next">
      <samlec:EncType>17</samlec:EncType>
      <samlec:EncType>18</samlec:EncType>
    <samlec:SessionKey>
  </S:Header>
  <S:Body>
    <samlp:AuthnRequest
      ID="c3a4f8b9c2d" Version="2.0" IssueInstant="2020-12-10T11:39:34Z"
      AssertionConsumerServiceURL="xmpp@xmpp.example.com">
      <saml:Issuer xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
       https://xmpp.example.com
      </saml:Issuer>
      <samlp:NameIDPolicy AllowCreate="true"
        Format="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent"/>
      <samlp:RequestedAuthnContext Comparison="exact">
       <saml:AuthnContextClassRef>
       urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
       </saml:AuthnContextClassRef>
      </samlp:RequestedAuthnContext>
    </samlp:AuthnRequest>
  </S:Body>
</S:Envelope>
```

Step 5 (alt): Server returns error to client:

```
<failure xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
 <incorrect-encoding/>
</failure>
</stream:stream>
```

Step 6: Client relays the request to IdP in a SOAP message transmitted over HTTP (over TLS). The HTTP portion is not shown, so the use of Basic Authentication is assumed. The body of the SOAP envelope is exactly the same as received in the previous step.

```
<S:Envelope
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
    xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Body>
        <samlp:AuthnRequest>
        <!-- same as above -->
        </samlp:AuthnRequest>
    </S:Body>
</S:Envelope>
```

Step 7: IdP responds to client with a SOAP response containing a SAML <Response> containing a short-lived SSO assertion (shown as an encrypted variant in the example). A generated key is included in the assertion and in a header for the client.

```
<S:Envelope
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
    xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Header>
    <ecp:Response S:mustUnderstand="1"
      S:actor="http://schemas.xmlsoap.org/soap/actor/next"
      AssertionConsumerServiceURL="xmpp@xmpp.example.com"/>
    <samlec:GeneratedKey xmlns:samlec="urn:ietf:params:xml:ns:samlec">
      3w1wSBKUosRLsU69xGK7dg==
    </samlec:GeneratedKey>
  </S:Header>
  <S:Body>
    <samlp:Response ID="d43h94r389309r" Version="2.0"
        IssueInstant="2020-12-10T11:42:34Z" InResponseTo="c3a4f8b9c2d"
        Destination="xmpp@xmpp.example.com">
      <saml:Issuer>https://saml.example.org</saml:Issuer>
      <samlp:Status>
        <samlp:StatusCode
            Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
      </samlp:Status>
      <saml:EncryptedAssertion>
        <!-- contents elided, copy of samlec:GeneratedKey in Advice -->
      </saml:EncryptedAssertion>
    </samlp:Response>
  </S:Body>
</S:Envelope>
```

Step 8: Client sends SOAP envelope containing the SAML <Response> as a response to the SASL server’s challenge:

```
<response xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
PFM6RW52ZWxvcGUKICAgIHhtbG5zOnNhbWw9InVybjpvYXNpczpuYW1lczp0YzpT
QU1MOjIuMDphc3NlcnRpb24iCiAgICB4bWxuczpzYW1scD0idXJuOm9hc2lzOm5h
bWVzOnRjOlNBTUw6Mi4wOnByb3RvY29sIgogICAgeG1sbnM6Uz0iaHR0cDovL3Nj
aGVtYXMueG1sc29hcC5vcmcvc29hcC9lbnZlbG9wZS8iPgogIDxTOkhlYWRlcj4K
ICAgIDxwYW9zOlJlc3BvbnNlIHhtbG5zOnBhb3M9InVybjpsaWJlcnR5OnBhb3M6
MjAwMy0wOCIKICAgICAgUzphY3Rvcj0iaHR0cDovL3NjaGVtYXMueG1sc29hcC5v
cmcvc29hcC9hY3Rvci9uZXh0IgogICAgICBTOm11c3RVbmRlcnN0YW5kPSIxIiBy
ZWZUb01lc3NhZ2VJRD0iNmMzYTRmOGI5YzJkIi8+CiAgICA8c2FtbGVjOlNlc3Np
b25LZXkgeG1sbnM6c2FtbGVjPSJ1cm46aWV0ZjpwYXJhbXM6eG1sOm5zOnNhbWxl
YyIKICAgICAgeG1sbnM6Uz0iaHR0cDovL3NjaGVtYXMueG1sc29hcC5vcmcvc29h
cC9lbnZlbG9wZS8iCiAgICAgIFM6bXVzdFVuZGVyc3RhbmQ9IjEiCiAgICAgIFM6
YWN0b3I9Imh0dHA6Ly9zY2hlbWFzLnhtbHNvYXAub3JnL3NvYXAvYWN0b3IvbmV4
dCI+CiAgICAgIDxzYW1sZWM6RW5jVHlwZT5hZXMxMjgtY3RzLWhtYWMtc2hhMS05
Njwvc2FtbGVjOkVuY1R5cGU+CiAgICA8c2FtbGVjOlNlc3Npb25LZXk+CiAgPC9T
OkhlYWRlcj4KICA8UzpCb2R5PgogICAgPHNhbWxwOlJlc3BvbnNlIElEPSJkNDNo
OTRyMzg5MzA5ciIgVmVyc2lvbj0iMi4wIgogICAgICAgIElzc3VlSW5zdGFudD0i
MjAwNy0xMi0xMTo0MjozNFoiIEluUmVzcG9uc2VUbz0iYzNhNGY4YjljMmQi
CiAgICAgICAgRGVzdGluYXRpb249InhtcHBAeG1wcC5leGFtcGxlLmNvbSI+CiAg
ICAgIDxzYW1sOklzc3Vlcj5odHRwczovL3NhbWwuZXhhbXBsZS5vcmc8L3NhbWw6
SXNzdWVyPgogICAgICA8c2FtbHA6U3RhdHVzPgogICAgICAgIDxzYW1scDpTdGF0
dXNDb2RlCiAgICAgICAgICAgIFZhbHVlPSJ1cm46b2FzaXM6bmFtZXM6dGM6U0FN
TDoyLjA6c3RhdHVzOlN1Y2Nlc3MiLz4KICAgICAgPC9zYW1scDpTdGF0dXM+CiAg
ICAgIDxzYW1sOkVuY3J5cHRlZEFzc2VydGlvbj4KICAgICAgICA8IS0tIGNvbnRl
bnRzIGVsaWRlZCwgY29weSBvZiBzYW1sZWM6R2VuZXJhdGVkS2V5IGluIEFkdmlj
ZSAtLT4KICAgICAgPC9zYW1sOkVuY3J5cHRlZEFzc2VydGlvbj4KICAgIDwvc2Ft
bHA6UmVzcG9uc2U+CiAgPC9TOkJvZHk+CjwvUzpFbnZlbG9wZT4K
</response>
```

The Base64 [RFC4648] decoded envelope:

```
<S:Envelope
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
    xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Header>
    <paos:Response xmlns:paos="urn:liberty:paos:2003-08"
      S:actor="http://schemas.xmlsoap.org/soap/actor/next"
      S:mustUnderstand="1" refToMessageID="6c3a4f8b9c2d"/>
    <samlec:SessionKey xmlns:samlec="urn:ietf:params:xml:ns:samlec"
      xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
      S:mustUnderstand="1"
      S:actor="http://schemas.xmlsoap.org/soap/actor/next">
      <samlec:EncType>17</samlec:EncType>
    <samlec:SessionKey>
  </S:Header>
  <S:Body>
    <samlp:Response ID="d43h94r389309r" Version="2.0"
        IssueInstant="2020-12-10T11:42:34Z" InResponseTo="c3a4f8b9c2d"
        Destination="xmpp@xmpp.example.com">
      <saml:Issuer>https://saml.example.org</saml:Issuer>
      <samlp:Status>
        <samlp:StatusCode
            Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
      </samlp:Status>
      <saml:EncryptedAssertion>
        <!-- contents elided, copy of samlec:GeneratedKey in Advice -->
      </saml:EncryptedAssertion>
    </samlp:Response>
  </S:Body>
</S:Envelope>
```

Step 9: Server informs client of successful authentication:

```
<success xmlns='urn:ietf:params:xml:ns:xmpp-sasl'/>
```

Step 9 (alt): Server informs client of failed authentication:

```
<failure xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
 <temporary-auth-failure/>
</failure>
</stream:stream>
```

Step 10: Client initiates a new stream to server:

```
<stream:stream xmlns='jabber:client'
xmlns:stream='http://etherx.jabber.org/streams'
to='example.com' version='1.0'>
```

Step 11: Server responds by sending a stream header to client along with any additional features (or an empty features element):

```
<stream:stream xmlns='jabber:client'
xmlns:stream='http://etherx.jabber.org/streams'
id='c2s_345' from='example.com' version='1.0'>
<stream:features>
 <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'/>
 <session xmlns='urn:ietf:params:xml:ns:xmpp-session'/>
</stream:features>
```

Step 12: Client binds a resource:

```
   <iq type='set' id='bind_1'>
     <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'>
       <resource>someresource</resource>
     </bind>
   </iq>
```

Step 13: Server informs client of successful resource binding:

```
   <iq type='result' id='bind_1'>
     <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'>
       <jid>somenode@example.com/someresource</jid>
     </bind>
   </iq>
```

Please note: line breaks were added to the base64 for clarity.

## 7. Security Considerations

This section will address only security considerations associated with the use of SAML with SASL applications. For considerations relating to SAML in general, the reader is referred to the SAML specification and to other literature. Similarly, for general SASL Security Considerations, the reader is referred to that specification.

Version 2.0 of the Enhanced Client or Proxy Profile [SAMLECP20] adds optional support for channel binding and use of "Holder of Key" subject confirmation. The former is strongly recommended for use with this mechanism to detect "Man in the Middle" attacks between the client and the RP without relying on the commercial TLS infrastructure that does not provide the level of assurance desired by sensitive SAML applications. The latter may be impractical in many cases, but is a valuable way of strengthening client authentication, protecting against phishing, and improving the overall mechanism.

### 7.1. Risks Left Unaddressed

The adaptation of a web-based profile that is largely designed around security-oblivious clients and a bearer model for security token validation results in a number of basic security exposures that should be weighed against the compatibility and client simplification benefits of this mechanism.

When channel binding is not used, protection against "Man in the Middle" attacks is left solely to lower layer protocols such as TLS, and the development of user interfaces able to implement that has not been effectively demonstrated. Failure to detect a MITM can result in phishing of the user's credentials if the attacker is between the client and IdP, or the theft and misuse of a short-lived credential (the SAML assertion) if the attacker is able to impersonate a RP. SAML allows for source address checking as a minor mitigation to the latter threat, but this is often impractical. IdPs can mitigate to some extent the exposure of personal information to RP attackers by encrypting assertions with authenticated keys.

### 7.2. User Privacy

The IdP is aware of each RP that a user logs into. There is nothing in the protocol to hide this information from the IdP. It is not a requirement to track the activity, but there is nothing technically that prohibits the collection of this information. Servers should be aware that SAML IdPs will track - to some extent - user access to their services. This exposure extends to the use of session keys generated by the IdP to secure messages between the parties, but note that when bearer assertions are involved, the IdP can freely impersonate the user to any relying party in any case.

It is also out of scope of the mechanism to determine under what conditions an IdP will release particular information to a relying party, and it is generally unclear in what fashion user consent could be established in real time for the release of particular

information. The SOAP exchange with the IdP does not preclude such interaction, but neither does it define that interoperably.

### 7.3. Collusion between RPs

Depending on the information supplied by the IdP, it may be possible for RPs to correlate data that they have collected. By using the same identifier to log into every RP, collusion between RPs is possible. SAML supports the notion of pairwise, or targeted/directed, identity. This allows the IdP to manage opaque, pairwise identifiers for each user that are specific to each RP. However, correlation is often possible based on other attributes supplied, and is generally a topic that is beyond the scope of this mechanism. It is sufficient to say that this mechanism does not introduce new correlation opportunities over and above the use of SAML in web-based use cases.

## 8. IANA Considerations

### 8.1. GSS-API and SASL Mechanism Registration

The IANA is requested to assign a new entry for this GSS mechanism in the sub-registry for SMI Security for Mechanism Codes, whose prefix is iso.org.dod.internet.security.mechanisms (1.3.6.1.5.5) and to reference this specification in the registry.

The IANA is requested to register the following SASL profile:

SASL mechanism profiles: SAML20EC and SAML20EC-PLUS

Security Considerations: See this document

Published Specification: See this document

For further information: Contact the authors of this document.

Owner/Change controller: the IETF

Note: None

### 8.2. XML Namespace Name for SAML-EC

A URN sub-namespace for XML constructs introduced by this mechanism is defined as follows:

URI: urn:ietf:params:xml:ns:samlec

Specification: See Appendix A of this document.

Description: This is the XML namespace name for XML constructs introduced by the SAML Enhanced Client SASL and GSS-API Mechanisms.

Registrant Contact: the IESG

# 9. References

## 9.1. Normative References

[OASIS.saml-bindings-2.0-os]
Cantor, S., Hirsch, F., Kemp, J., Philpott, R., and E. Maler, "Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0", OASIS Standard saml-bindings-2.0-os, March 2005.

[OASIS.saml-core-2.0-os]
Cantor, S., Kemp, J., Philpott, R., and E. Maler, "Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V2.0", OASIS Standard saml-core-2.0-os, March 2005.

[OASIS.saml-profiles-2.0-os]
Hughes, J., Cantor, S., Hodges, J., Hirsch, F., Mishra, P., Philpott, R., and E. Maler, "Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0", OASIS Standard OASIS.saml-profiles-2.0-os, March 2005.

[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>.

[RFC3962] Raeburn, K., "Advanced Encryption Standard (AES) Encryption for Kerberos 5", RFC 3962, DOI 10.17487/RFC3962, February 2005, <https://www.rfc-editor.org/info/rfc3962>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[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>.

[RFC4422] Melnikov, A., Ed. and K. Zeilenga, Ed., "Simple Authentication and Security Layer (SASL)", RFC 4422, DOI 10.17487/RFC4422, June 2006, <https://www.rfc-editor.org/info/rfc4422>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/info/rfc4648>.

[RFC5554] Williams, N., "Clarifications and Extensions to the Generic Security Service Application Program Interface (GSS-API) for the Use of Channel Bindings", RFC 5554, DOI 10.17487/RFC5554, May 2009, <https://www.rfc-editor.org/info/rfc5554>.

[RFC5801] Josefsson, S. and N. Williams, "Using Generic Security Service Application Program Interface (GSS-API) Mechanisms in Simple Authentication and Security Layer (SASL): The GS2 Mechanism Family", RFC 5801, DOI 10.17487/RFC5801, July 2010, <https://www.rfc-editor.org/info/rfc5801>.

[RFC6680] Williams, N., Johansson, L., Hartman, S., and S. Josefsson, "Generic Security Service Application Programming Interface (GSS-API) Naming Extensions", RFC 6680, DOI 10.17487/RFC6680, August 2012, <https://www.rfc-editor.org/info/rfc6680>.

[RFC7056] Hartman, S. and J. Howlett, "Name Attributes for the GSS-API Extensible Authentication Protocol (EAP) Mechanism", RFC 7056, DOI 10.17487/RFC7056, December 2013, <https://www.rfc-editor.org/info/rfc7056>.

[RFC7617] Reschke, J., "The 'Basic' HTTP Authentication Scheme", RFC 7617, DOI 10.17487/RFC7617, September 2015, <https://www.rfc-editor.org/info/rfc7617>.

[RFC7802] Emery, S. and N. Williams, "A Pseudo-Random Function (PRF) for the Kerberos V Generic Security Service Application Program Interface (GSS-API) Mechanism", RFC 7802, DOI 10.17487/RFC7802, March 2016, <https://www.rfc-editor.org/info/rfc7802>.

[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>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[SAMLECP20] Cantor, S., "SAML V2.0 Enhanced Client or Proxy Profile Version 2.0", OASIS Committee Specification OASIS.sstc-saml-ecp-v2.0-cs01, August 2013.

[W3C.soap11] Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H., Thatte, S., and D. Winer, "Simple Object Access Protocol (SOAP) 1.1", W3C Note soap11, May 2000, <http://www.w3.org/TR/SOAP/>.

## 9.2. Informative References

[OASIS.saml-metadata-2.0-os] Cantor, S., Moreh, J., Philpott, R., and E. Maler, "Metadata for the Security Assertion Markup Language (SAML) V2.0", OASIS Standard saml-metadata-2.0-os, March 2005.

[RFC1939] Myers, J. and M. Rose, "Post Office Protocol - Version 3", STD 53, RFC 1939, DOI 10.17487/RFC1939, May 1996, <https://www.rfc-editor.org/info/rfc1939>.

[RFC3501] Crispin, M., "INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1", RFC 3501, DOI 10.17487/RFC3501, March 2003, <https://www.rfc-editor.org/info/rfc3501>.

[RFC4559] Jaganathan, K., Zhu, L., and J. Brezak, "SPNEGO-based Kerberos and NTLM HTTP Authentication in Microsoft Windows", RFC 4559, DOI 10.17487/RFC4559, June 2006, <https://www.rfc-editor.org/info/rfc4559>.

[RFC6120] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, DOI 10.17487/RFC6120, March 2011, <https://www.rfc-editor.org/info/rfc6120>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/info/rfc7230>.

[W3C.REC-xmlschema-1] Thompson, H., Beech, D., Maloney, M., and N. Mendelsohn, "XML Schema Part 1: Structures", W3C REC-xmlschema-1, May 2001, <http://www.w3.org/TR/xmlschema-1/>.

[WSS-SAML] Monzillo, R., "Web Services Security SAML Token Profile Version 1.1.1", OASIS Standard OASIS.wss-SAMLTokenProfile, May 2012.

## Appendix A. XML Schema

The following schema formally defines the "urn:ietf:params:xml:ns:samlec" namespace used in this document, in conformance with [W3C.REC-xmlschema-1] While XML validation is optional, the schema that follows is the normative definition of the constructs it defines. Where the schema differs from any prose in this specification, the schema takes precedence.

```
   <schema
     targetNamespace="urn:ietf:params:xml:ns:samlec"
     xmlns="http://www.w3.org/2001/XMLSchema"
     xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
     xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
     xmlns:samlec="urn:ietf:params:xml:ns:samlec"
     elementFormDefault="unqualified"
     attributeFormDefault="unqualified"
     blockDefault="substitution"
     version="1.0">

     <import namespace="http://www.w3.org/2000/09/xmldsig#"/>
     <import namespace="http://schemas.xmlsoap.org/soap/envelope/"/>

     <element name="SessionKey" type="samlec:SessionKeyType"/>
     <complexType name="SessionKeyType">
       <sequence>
         <element ref="samlec:EncType" maxOccurs="unbounded"/>
         <element ref="ds:KeyInfo" minOccurs="0"/>
       </sequence>
       <attribute ref="S:mustUnderstand" use="required"/>
       <attribute ref="S:actor" use="required"/>
       <attribute name="Algorithm"/>
     </complexType>

     <element name="EncType" type="integer"/>

     <element name="GeneratedKey" type="samlec:GeneratedKeyType"/>
     <complexType name="GeneratedKeyType">
       <simpleContent>
         <extension base="base64Binary">
           <attribute ref="S:mustUnderstand"/>
           <attribute ref="S:actor"/>
         </extension>
       </simpleContent>
     </complexType>

     <element name="Delegated" type="samlec:DelegatedType"/>
     <complexType name="DelegatedType">
       <sequence/>
       <attribute ref="S:mustUnderstand" use="required"/>
       <attribute ref="S:actor" use="required"/>
     </complexType>

   </schema>
```

## Appendix B. Acknowledgments

The authors would also like to thank Klaas Wierenga, Sam Hartman, Nico Williams, Jim Basney, Venkat Yekkirala, and Ben Kaduk for their contributions.

## Appendix C. Changes

This section to be removed prior to publication.

- 20, address nits and easy fixes from Ben Kaduk's AD review
- 19, update obsoleted references
- 15,16,17,18 avoid expiration
- 14, address some minor comments
- 13, clarify SAML metadata usage, adding a recommended Binding value alongside the backward-compatibility usage of PAOS
- 12, clarifying comments based on WG feedback, with a normative change to use enctype numbers instead of names
- 11, update EAP Naming reference to RFC
- 10, update SAML ECP reference to final CS
- 09, align delegation signaling to updated ECP draft
- 08, more corrections, added a delegation signaling header
- 07, corrections, revised section on delegation
- 06, simplified session key schema, moved responsibility for random-to-key to the endpoints, and defined advertisement of session key algorithm and enctypes by acceptor
- 05, revised session key material, added requirement for random-to-key, revised XML schema to capture enctype name, updated GSS naming reference
- 04, stripped down the session key material to simplify it, and define an IdP-brokered keying approach, moved session key XML constructs from OASIS draft into this one
- 03, added TLS key export as a session key option, revised GSS naming material based on list discussion

- 02, major revision of GSS-API material and updated references
- 01, SSH language added, noted non-assumption of HTTP error handling, added guidance on life of security context.
- 00, Initial Revision, first WG-adopted draft. Removed support for unsolicited SAML responses.

## Authors' Addresses

Scott Cantor
Shibboleth Consortium
1050 Carmack Rd
Columbus, Ohio 43210
United States

Phone: +1 614 247 6147
Email: cantor.2@osu.edu

Margaret Cullen
Painless Security
4 High St, Suite 134
North Andover, Massachusets 01845
United States

Phone: +1 781 405 7464
Email: mrcullen42@painless-security.com

Simon Josefsson
SJD AB
Hagagatan 24
Stockholm 113 47
SE

Email: simon@josefsson.org
URI: http://josefsson.org/

Internet Engineering Task Force
Internet-Draft
Updates: 4120 (if approved)
Intended status: Standards Track
Expires: April 6, 2015

S. Sorce, Ed.
Red Hat
T. Yu, Ed.
T. Hardjono, Ed.
MIT Kerberos Consortium

October 3, 2014

# Kerberos Authorization Data Container Authenticated by Multiple MACs
draft-ietf-krb-wg-cammac-11

## Abstract

Abstract: This document specifies a Kerberos Authorization Data container that supersedes AD-KDC-ISSUED. It allows for multiple Message Authentication Codes (MACs) or signatures to authenticate the contained Authorization Data elements. This document updates RFC 4120.

## 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 April 6, 2015.

## Copyright Notice

Copyright (c) 2014 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 Language . . . . . 2
3. Motivations . . . . . 2
4. Encoding . . . . . 4
5. Usage . . . . . 6
6. Assigned numbers . . . . . 6
7. IANA Considerations . . . . . 6
8. Security Considerations . . . . . 6
9. Acknowledgements . . . . . 7
10. References . . . . . 8
10.1. Normative References . . . . . 8
10.2. Informative References . . . . . 8
Authors' Addresses . . . . . 8

## 1. Introduction

This document specifies a new Authorization Data container for Kerberos, called AD-CAMMAC (Container Authenticated by Multiple MACs), that supersedes AD-KDC-ISSUED. This new container allows both the receiving application service and the Key Distribution Center (KDC) itself to verify the authenticity of the contained authorization data. The AD-CAMMAC container can also include additional verifiers that "trusted services" can use to verify the contained authorization data.

## 2. Requirements Language

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 RFC 2119 [RFC2119].

## 3. Motivations

The Kerberos protocol allows clients to submit arbitrary authorization data for a KDC to insert into a Kerberos ticket. These client-requested authorization data allow the client to express authorization restrictions that the application service will interpret. With few exceptions, the KDC can safely copy these client-requested authorization data to the issued ticket without necessarily inspecting, interpreting, or filtering their contents.

The AD-KDC-ISSUED authorization data container specified in RFC 4120 [RFC4120] is a means for KDCs to include positive or permissive

(rather than restrictive) authorization data in service tickets in a way that the service named in a ticket can verify that the KDC has issued the contained authorization data. This capability takes advantage of a shared symmetric key between the KDC and the service to assure the service that the KDC did not merely copy client-requested authorization data to the ticket without inspecting them.

The AD-KDC-ISSUED container works well for situations where the flow of authorization data is from the KDC to the service. However, protocol extensions such as Constrained Delegation (S4U2Proxy [MS-SFU]) require that a service present to the KDC a service ticket that the KDC previously issued, as evidence that the service is authorized to impersonate the client principal named in that ticket. In the S4U2Proxy extension, the KDC uses the evidence ticket as the basis for issuing a derivative ticket that the service can then use to impersonate the client. The authorization data contained within the evidence ticket constitute a flow of authorization data from the application service to the KDC. The properties of the AD-KDC-ISSUED container are insufficient for this use case because the service knows the symmetric key for the checksum in the AD-KDC-ISSUED container. Therefore, the KDC has no way to detect whether the service has tampered with the contents of the AD-KDC-ISSUED container within the evidence ticket.

The new AD-CAMMAC authorization data container specified in this document improves upon AD-KDC-ISSUED by including additional verifier elements. The svc-verifier element of the CAMMAC has the same functional and security properties as the ad-checksum element of AD-KDC-ISSUED; the svc-verifier allows the service to verify the integrity of the AD-CAMMAC contents as it already could with the AD-KDC-ISSUED container. The kdc-verifier and other-verifiers elements are new to AD-CAMMAC and provide its enhanced capabilities.

The kdc-verifier element of the AD-CAMMAC container allows a KDC to verify the integrity of authorization data that it previously inserted into a ticket, by using a key that only the KDC knows. The KDC thus avoids recomputing all of the authorization data for the issued ticket; this operation might not always be possible when that data includes ephemeral information such as the strength or type of authentication method used to obtain the original ticket.

The verifiers in the other-verifiers element of the AD-CAMMAC container are not required, but can be useful when a lesser-privileged service receives a ticket from a client and needs to extract the CAMMAC to demonstrate to a higher-privileged "trusted service" on the same host that it is legitimately acting on behalf of that client. The trusted service can use a verifier in the other-

verifiers element to validate the contents of the CAMMAC without further communication with the KDC.

## 4. Encoding

The Kerberos protocol is defined in [RFC4120] using Abstract Syntax Notation One (ASN.1) [X.680] and using the ASN.1 Distinguished Encoding Rules (DER) [X.690]. For consistency, this specification also uses ASN.1 for specifying the layout of AD-CAMMAC. The ad-data of the AD-CAMMAC authorization data element is the ASN.1 DER encoding of the AD-CAMMAC ASN.1 type specified below.

```
KerberosV5CAMMAC DEFINITIONS EXPLICIT TAGS ::= BEGIN

AD-CAMMAC                   ::= SEQUENCE {
      elements              [0] AuthorizationData,
      kdc-verifier          [1] Verifier-MAC OPTIONAL,
      svc-verifier          [2] Verifier-MAC OPTIONAL,
      other-verifiers       [3] SEQUENCE (SIZE (1..MAX))
                                OF Verifier OPTIONAL
}

Verifier             ::= CHOICE {
      mac            Verifier-MAC,
      ...
}

Verifier-MAC         ::= SEQUENCE {
      identifier     [0] PrincipalName OPTIONAL,
      kvno           [1] UInt32 OPTIONAL,
      enctype        [2] Int32 OPTIONAL,
      mac            [3] Checksum
}

END
```

elements:
: A sequence of authorization data elements issued by the KDC. These elements are the authorization data that the verifier fields authenticate.

Verifier:
: A CHOICE type that currently contains only one alternative: Verifier-MAC. Future extensions might add support for public-key signatures.

Verifier-MAC:

Contains an RFC 3961 [RFC3961] Checksum (MAC) computed over the ASN.1 DER encoding of the AuthorizationData value in the elements field of the AD-CAMMAC. The identifier, kvno, and enctype fields help the recipient locate the key required for verifying the MAC. For the kdc-verifier and the svc-verifier, the identifier, kvno and enctype fields are often obvious from context and MAY be omitted. For the kdc-verifier, the MAC is computed differently than for the svc-verifier and the other-verifiers, as described later. The key usage for computing the MAC (Checksum) is 64.

kdc-verifier:
A Verifier-MAC where the key is a long-term key of the local Ticket-Granting Service (TGS). The checksum type is the required checksum type for the enctype of the TGS key. In contrast to the other Verifier-MAC elements, the KDC computes the MAC in the kdc-verifier over the ASN.1 DER encoding of the EncTicketPart of the surrounding ticket, but where the AuthorizationData value in the EncTicketPart contains the AuthorizationData value contained in the CAMMAC instead of the AuthorizationData value that would otherwise be present in the ticket. This altered Verifier-MAC computation binds the kdc-verifier to the other contents of the ticket, assuring the KDC that a malicious service has not substituted a mismatched CAMMAC received from another ticket.

svc-verifier:
A Verifier-MAC where the key is the same long-term service key that the KDC uses to encrypt the surrounding ticket. The checksum type is the required checksum type for the enctype of the service key used to encrypt the ticket. This field MUST be present if the service principal of the ticket is not the local TGS, including when the ticket is a cross-realm TGT.

other-verifiers:
A sequence of additional verifiers. In each additional Verifier-MAC, the key is a long-term key of the principal name specified in the identifier field. The PrincipalName MUST be present and be a valid principal in the realm. KDCs MAY add one or more "trusted service" verifiers. Unless otherwise administratively configured, the KDC SHOULD determine the "trusted service" principal name by replacing the service identifier component of the sname of the surrounding ticket with "host". The checksum is computed using a long-term key of the identified principal, and the checksum type is the required checksum type for the enctype of that long-term key. The kvno and enctype SHOULD be specified to disambiguate which of the long-term keys of the trusted service is used.

## 5. Usage

Application servers and KDCs MAY ignore the AD-CAMMAC container and the authorization data elements it contains. For compatibility with older Kerberos implementations, a KDC issuing an AD-CAMMAC SHOULD enclose it in an AD-IF-RELEVANT container unless the KDC knows that the application server is likely to recognize it.

## 6. Assigned numbers

The ad-type number for AD-CAMMAC is 96.

The key usage number for the Verifier-MAC checksum is 64.

## 7. IANA Considerations

[ RFC Editor: please remove this section prior to publication. ]

There are no IANA considerations in this document. Any numbers assigned in this document are not in IANA-controlled number spaces.

## 8. Security Considerations

Although authorization data are generally conveyed within the encrypted part of a ticket and are thereby protected by the existing encryption scheme used for the surrounding ticket, some authorization data requires the additional protection provided by the CAMMAC.

Some protocol extensions such as S4U2Proxy allow the KDC to issue a new ticket based on an evidence ticket provided by the service. If the evidence ticket contains authorization data that needs to be preserved in the new ticket, then the KDC MUST revalidate it.

Extracting a CAMMAC from a ticket for use as a credential removes it from the context of the ticket. In the general case, this could turn it into a bearer token, with all of the associated security implications. Also, the CAMMAC does not itself necessarily contain sufficient information to identify the client principal. Therefore, application protocols that rely on extracted CAMMACs might need to duplicate a substantial portion of the ticket contents and include that duplicated information in the authorization data contained within the CAMMAC. The extent of this duplication would depend on the security properties required by the application protocol.

The method for computing the kdc-verifier does not bind it to any authorization data within the ticket but outside of the CAMMAC. At least one (non-standard) authorization data type, AD-SIGNEDPATH,

attempts to bind to other authorization data in a ticket, and it is very difficult for two such authorization data types to coexist.

To minimize ticket size when embedding CAMMACs in Kerberos tickets, a KDC MAY omit the kdc-verifier from the CAMMAC when it is not needed. In this situation, the KDC cannot always determine whether the CAMMAC contents are intact. The KDC MUST NOT create a new CAMMAC from an existing one unless the existing CAMMAC has a valid kdc-verifier, with two exceptions.

Only KDCs for the local realm have knowledge of the local TGS key, so the outer encryption of a local TGT is sufficient to protect the CAMMAC of a local TGT from tampering, assuming that all of the KDCs in the local realm consistently filter out CAMMAC authorization data submitted by clients. The KDC MAY create a new CAMMAC from an existing CAMMAC lacking a kdc-verifier if that CAMMAC is contained within a local TGT and all of the local realm KDCs are configured to filter out CAMMAC authorization data submitted by clients.

An application service might not use the S4U2Proxy extension, or the realm policy might disallow the use of S4U2Proxy by that service. In such situations where there is no flow of authorization data from the service to the KDC, the application service could modify the CAMMAC contents, but such modifications would have no effect on other services. Because of the lack of security impact, the KDC MAY create a new CAMMAC from an existing CAMMAC lacking a kdc-verifier if it is inserting the new CAMMAC into a service ticket for the same service principal as the ticket that contained the existing CAMMAC, but MUST NOT place a kdc-verifier in the new CAMMAC.

The kdc-verifier in CAMMAC does not bind the service principal name to the CAMMAC contents, because the service principal name is not part of the EncTicketPart. An entity that has access to the keys of two different service principals can decrypt a ticket for one service and encrypt it in the key of the other service, altering the svc-verifier to match. Both the kdc-verifier and the svc-verifier would still validate, but the KDC never issued this fabricated ticket. The impact of this manipulation is minor if the CAMMAC contents only communicate attributes related to the client. If an application requires an authenticated binding between the service principal name and the CAMMAC or ticket contents, the KDC MUST include in the CAMMAC some authorization data element that names the service principal.

## 9. Acknowledgements

Shawn Emery, Sam Hartman, Greg Hudson, Ben Kaduk, Zhanna Tsitkov, and Kai Zheng provided helpful technical and editorial feedback on earlier versions of 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, March 1997.

[RFC3961] Raeburn, K., "Encryption and Checksum Specifications for Kerberos 5", RFC 3961, February 2005.

[RFC4120] Neuman, C., Yu, T., Hartman, S., and K. Raeburn, "The Kerberos Network Authentication Service (V5)", RFC 4120, July 2005.

[X.680] ISO, , "Information technology -- Abstract Syntax Notation One (ASN.1): Specification of basic notation -- ITU-T Recommendation X.680 (ISO/IEC International Standard 8824-1:2008)", 2008.

[X.690] ISO, , "Information technology -- ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER) -- ITU-T Recommendation X.690 (ISO/IEC International Standard 8825-1:2008)", 1997.

## 10.2. Informative References

[MS-SFU] Microsoft, "[MS-SFU]: Kerberos Protocol Extensions: Service for User and Constrained Delegation Protocol", January 2013, <http://msdn.microsoft.com/en-us/library/cc246071.aspx>.

# Authors' Addresses

Simo Sorce (editor)
Red Hat

Email: ssorce@redhat.com

Tom Yu (editor)
MIT Kerberos Consortium

Email: tlyu@mit.edu

Thomas Hardjono (editor)
MIT Kerberos Consortium

Email: hardjono@mit.edu