      Generic Security Service Application Programming Interface (GSS-API)
                            Naming Extensions

Abstract

   The Generic Security Service Application Programming Interface
   (GSS-API) provides a simple naming architecture that supports name-
   based authorization.  This document introduces new APIs that extend
   the GSS-API naming model to support name attribute transfer between
   GSS-API peers.

Status of This Memo

   This is an Internet Standards Track document.

   This document is a product of the Internet Engineering Task Force
   (IETF).  It represents the consensus of the IETF community.  It has
   received public review and has been approved for publication by the
   Internet Engineering Steering Group (IESG).  Further information on
   Internet Standards is available in Section 2 of RFC 5741.

   Information about the current status of this document, any errata,
   and how to provide feedback on it may be obtained at
   http://www.rfc-editor.org/info/rfc6680.

Table of Contents

1.  Introduction

   As described in [RFC4768], the GSS-API's naming architecture suffers
   from certain limitations.  This document attempts to overcome these
   limitations.

   A number of extensions to the GSS-API [RFC2743] and its C-bindings
   [RFC2744] are described herein.  The goal is to make information
   modeled as "name attributes" available to applications.  Such
   information MAY, for instance, be used by applications to make
   authorization decisions.  For example, Kerberos V authorization data
   elements, both in their raw forms as well as mapped to more useful
   value types, can be made available to GSS-API applications through
   these interfaces.

   The model is that GSS names have attributes.  The attributes of a
   name may be authenticated (e.g., an X509 attribute certificate or
   signed Security Assertion Markup Language (SAML) attribute assertion)
   or may have been set on a GSS name for the purpose of locally
   "asserting" the attribute during credential acquisition or security
   context exchange.  Name attributes' values are network
   representations thereof (e.g., the actual value octets of the
   contents of an X.509 certificate extension, for example) and are
   intended to be useful for constructing portable access control
   facilities.  Applications may often require language- or platform-
   specific data types, rather than network representations of name
   attributes, so a function is provided to obtain objects of such types
   associated with names and name attributes.

   Future updates of this specification may involve adding an attribute
   namespace for attributes that only have application-specific
   semantics.  Note that mechanisms will still need to know how to
   transport such attributes.  The IETF may also wish to add functions
   by which to inquire whether a mechanism(s) understands a given
   attribute name or namespace and to list which attributes or attribute
   namespaces a mechanism understands.  Finally, the IETF may want to
   consider adding a function by which to determine the name of the
   issuer of a name attribute.

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.  Name Attribute Authenticity

   An attribute is "authenticated" if and only if there is a secure
   association between the attribute (and its values) and the trusted
   source of the peer credential.  Examples of authenticated attributes
   are (any part of) the signed portion of an X.509 certificate or
   AD-KDCIssued authorization data elements (Section 5.2.6.2 of
   [RFC4120]) in Kerberos V Tickets, provided, of course, that the
   authenticity of the respective security associations (e.g.,
   signatures) has been verified.

   Note that the fact that an attribute is authenticated does not imply
   anything about the semantics of the attribute nor that the trusted
   credential source was authorized to assert the attribute.  Such
   interpretations SHOULD be the result of applying local policy to the
   attribute.

   An unauthenticated attribute is called _asserted_ in what follows.
   This is not to be confused with other uses of the words "asserted" or
   "assertion" such as "SAML attribute assertion", the attributes of
   which may be authenticated in the sense of this document, for
   instance, if the SAML attribute assertion was signed by a key trusted
   by the peer.

4.  Name Attributes/Values as ACL Subjects

   To facilitate the development of portable applications that make use
   of name attributes to construct and evaluate portable Access Control
   Lists (ACLs), the GSS-API makes name attribute values available in
   canonical network encodings thereof.

5.  Naming Contexts

   Several factors influence the context in which a name attribute is
   interpreted.  One is the trust context.

   As discussed previously, applications apply local policy to determine
   whether a particular peer credential issuer is trusted to make a
   given statement.  Different GSS-API mechanisms and deployments have
   different trust models surrounding attributes they provide about a
   name.

   For example, Kerberos deployments in the enterprise typically trust a
   Key Distribution Center (KDC) to make any statement about principals
   in a realm.  This includes attributes such as group membership.

In contrast, in a federated SAML environment, the identity provider
typically exists in a different organization than the acceptor.  In
this case, the set of group memberships or entitlements that the IDP
is permitted to make needs to be filtered by the policy of the
acceptor and federation.

So even an attribute containing the same information, such as email
address, would need to be treated differently by the application in
the context of an enterprise deployment from the context of a
federation.

Another aspect related to trust is the role of the credential issuer
in providing the attribute.  Consider Public Key Cryptography for
Initial Authentication in Kerberos (PKINIT) [RFC4556].  In this
protocol, a public key and associated certificate are used to
authenticate to a Kerberos KDC.  Consider how attributes related to a
PKINIT certificate should be made available in GSS-API
authentications based on the Kerberos ticket.  In some deployments,
the certificate may be fully trusted; by including the certificate
information in the ticket, the KDC permits the acceptor to trust the
information in the certificate just as if the KDC itself had made
these statements.  In other deployments, the KDC may have authorized
a hash of the certificate without evaluating the content of the
certificate or generally trusting the issuing certification
authority.  In this case, if the certificate were included in the
issued ticket, the KDC would only be making the statement that the
certificate was used in the authentication.  This statement would be
authenticated but would not imply that the KDC asserted that
particular attributes of the certificate accurately described the
initiator.

Another aspect of context is encoding of the attribute information.
An attribute containing an ASCII [ANSI.X3-4.1986] or UTF-8 [RFC3629]
version of an email address could not be interpreted the same as an
ASN.1 Distinguished Encoding Rules email address in a certificate.

All of these contextual aspects of a name attribute affect whether
two attributes can be treated the same by an application and thus
whether they should be considered the same name attribute.  In the
GSS-API naming extensions, attributes that have different contexts
MUST have different names so they can be distinguished by
applications.  As an unfortunate consequence of this requirement,
multiple attribute names will exist for the same basic information.
That is, there is no single attribute name for the email address of
an initiator.  Other aspects of how mechanisms describe information
about subjects would already make this true.  For example, some
mechanisms use OIDs to name attributes; others use URIs.

Local implementations or platforms are likely to have sufficient
policy and information to know when contexts can be treated as the
same.  For example, the GSS-API implementation may know that a
particular certification authority can be trusted in the context of a
PKINIT authentication.  The local implementation may have sufficient
policy to know that a particular credential issuer is trusted to make
a given statement.  In order to take advantage of this local
knowledge within the GSS-API implementation, naming extensions
support the concept of local attributes in addition to standard
attributes.  For example, an implementation might provide a local
attribute for email address.  The implementation would specify the
encoding and representation of this attribute; mechanism-specific
standards attributes would be re-encoded if necessary to meet this
representation.  Only email addresses in contexts that meet the
requirements of local policy would be mapped into this local
attribute.

Such local attributes inherently expose a trade-off between
interoperability and usability.  Using a local attribute in an
application requires knowledge of the local implementation.  However,
using a standardized attribute in an application requires more
knowledge of policy and more validation logic in the application.
Sharing this logic in the local platform provides more consistency
across applications as well as reduces implementation costs.  Both
options are needed.

6.  Representation of Attribute Names

Different underlying mechanisms (e.g., SAML or X.509 certificates)
provide different representations for the names of their attributes.
In X.509 certificates, most objects are named by object identifiers
(OIDs).  The type of object (certificate extension, name constraint,
keyPurposeID, etc.) along with the OID is sufficient to identify the
attribute.  By contrast, according to Sections 8.2 and 2.7.3.1 of
[OASIS.saml-core-2.0-os], the name of an attribute has two parts.
The first is a URI describing the format of the name.  The second
part, whose form depends on the format URI, is the actual name.  In
other cases, an attribute might represent a certificate that plays
some particular role in a GSS-API mechanism; such attributes might
have a simple mechanism-defined name.

Attribute names MUST support multiple components.  If there is more
than one component in an attribute name, the more significant
components define the semantics of the less significant components.

Attribute names are represented as OCTET STRING elements in the API
described below.  These attribute names have syntax and semantics
that are understood by the application and by the lower-layer
implementations (some of which are described below).

If an attribute name contains a space (ASCII 0x20), the first space
separates the most significant or primary component of the name from
the remainder.  We may refer to the primary component of the
attribute name as the attribute name's "prefix".  If there is no
space, the primary component is the entire name; otherwise, it
defines the interpretation of the remainder of the names.

If the primary component contains a ":" (ASCII 0x3a), then the
primary component is a URI.  Otherwise, the attribute is a local
attribute and the primary component has meaning to the implementation
of GSS-API or to the specific configuration of the application.
Local attribute names with an "at" sign ("@") in them are reserved
for future allocation by the IETF.

Since attribute names are split at the first space into prefix and
suffix, there is a potential for ambiguity if a mechanism blindly
passes through a name attribute whose name it does not understand.
In order to prevent such ambiguities, the mechanism MUST always
prefix raw name attributes with a prefix that reflects the context of
the attribute.

Local attribute names under the control of an administrator or a
sufficiently trusted part of the platform need not have a prefix to
describe context.

7.  API

7.1.  SET OF OCTET STRING

The construct "SET OF OCTET STRING" occurs once in RFC 2743
[RFC2743], where it is used to represent a set of status strings in
the GSS_Display_status call.  The Global Grid Forum has defined SET
OF OCTET STRING as a buffer set type in GFD.024 [GFD.024], which also
provides one API for memory management of these structures.  The
normative reference to GFD.024 [GFD.024] is for the buffer set
functions defined in Section 2.5 and the associated buffer set C
types defined in Section 6 (namely gss_buffer_set_desc,
gss_buffer_set_t, gss_create_empty_buffer_set,
gss_add_buffer_set_member, gss_release_buffer_set).  Nothing else
from GFD.024 is required to implement this document.  In particular,
that document specifies changes to the behavior of existing GSS-API

functions in Section 3: implementing those changes are not required
to implement this document.  Any implementation of SET OF OCTET
STRING for use by this specification MUST preserve order.

## 7.2.  Const Types

The C-bindings for the new APIs use some types from [RFC5587] to
avoid issues with the use of "const".  The normative reference to
[RFC5587] is for the C types specified in Figure 1 of Section 3.4.6.
Nothing else from that document is required to implement this
document.

## 7.3.  GSS_Display_name_ext()

Inputs:

o   name INTERNAL NAME

o   display_as_name_type OBJECT IDENTIFIER

Outputs:

o   major_status INTEGER

o   minor_status INTEGER

o   display_name OCTET STRING -- caller must release with
    GSS_Release_buffer()

Return major_status codes:

o   GSS_S_COMPLETE indicates no error.

o   GSS_S_UNAVAILABLE indicates that the given name could not be
    displayed using the syntax of the given name type.

o   GSS_S_FAILURE indicates a general error.

This function displays a given name using the given name syntax, if
possible.  This operation may require mapping Mechanism Names (MNs)
to generic name syntaxes or generic name syntaxes to mechanism-
specific name syntaxes.  Such mappings may not always be feasible and
MAY be inexact or lossy; therefore, this function may fail.

7.3.1.  C-Bindings

   The display_name buffer is de-allocated by the caller with
   gss_release_buffer.

   OM_uint32 gss_display_name_ext(
      OM_uint32                      *minor_status,
      gss_const_name_t               name,
      gss_const_OID                  display_as_name_type,
      gss_buffer_t                   display_name
   );

7.4.  GSS_Inquire_name()

   Inputs:

   o  name INTERNAL NAME

   Outputs:

   o  major_status INTEGER

   o  minor_status INTEGER

   o  name_is_MN BOOLEAN

   o  mn_mech OBJECT IDENTIFIER

   o  attrs SET OF OCTET STRING -- the caller is responsible for de-
      allocating memory using GSS_Release_buffer_set

   Return major_status codes:

   o  GSS_S_COMPLETE indicates no error.

   o  GSS_S_FAILURE indicates a general error.

   This function outputs the set of attributes of a name.  It also
   indicates if a given name is an Mechanism Name (MN) or not and, if it
   is, the mechanism of which it's an MN.

7.4.1.  C-Bindings

```
OM_uint32 gss_inquire_name(
  OM_uint32                     *minor_status,
  gss_const_name_t              name,
  int                           *name_is_MN,
  gss_OID                       *MN_mech,
  gss_buffer_set_t              *attrs
);
```

   The gss_buffer_set_t is used here as the C representation of SET OF
   OCTET STRING.  This type is used to represent a set of attributes and
   is a NULL-terminated array of gss_buffer_t.  The gss_buffer_set_t
   type and associated API is defined in GFD.024 [GFD.024].  The "attrs"
   buffer set is de-allocated by the caller using
   gss_release_buffer_set().

7.5.  GSS_Get_name_attribute()

   Inputs:

   o  name INTERNAL NAME

   o  attr OCTET STRING

   Outputs:

   o  major_status INTEGER

   o  minor_status INTEGER

   o  authenticated BOOLEAN -- TRUE if and only if authenticated by the
      trusted peer credential source

   o  complete BOOLEAN -- TRUE if and only if this represents a complete
      set of values for the name

   o  values SET OF OCTET STRING -- the caller is responsible for de-
      allocating memory using GSS_Release_buffer_set

   o  display_values SET OF OCTET STRING -- the caller is responsible
      for de-allocating memory using GSS_Release_buffer_set

   Return major_status codes:

   o  GSS_S_COMPLETE indicates no error.

o GSS_S_UNAVAILABLE indicates that the given attribute OID is not
  known or set.

o GSS_S_FAILURE indicates a general error.

This function outputs the value(s) associated with a given GSS name
object for a given name attribute.

The complete flag denotes that (if TRUE) the set of values represents
a complete set of values for this name.  The peer being an
authoritative source of information for this attribute is a
sufficient condition for the complete flag to be set by the peer.

In the federated case, when several peers may hold some of the
attributes about a name, this flag may be highly dangerous and SHOULD
NOT be used.

NOTE: This function relies on the GSS-API notion of "SET OF" allowing
for order preservation; this has been discussed on the KITTEN WG
mailing list, and the consensus seems to be that, indeed, that was
always the intention.  It should be noted, however, that the order
presented does not always reflect an underlying order of the
mechanism-specific source of the attribute values.

7.5.1.  C-Bindings

The C-bindings of GSS_Get_name_attribute() require one function call
per attribute value for multi-valued name attributes.  This is done
by using a single gss_buffer_t for each value and an input/output
integer parameter to distinguish initial and subsequent calls and to
indicate when all values have been obtained.

The "more" input/output parameter should point to an integer variable
whose value, on first call to gss_get_name_attribute(), MUST be -1
and whose value upon function call return will be non-zero to
indicate that additional values remain or zero to indicate that no
values remain.  The caller should not modify this parameter after the
initial call.  The status of the complete and authenticated flags
MUST NOT change between multiple calls to iterate over values for an
attribute.

The output buffers "value" and "display_value" are de-allocated by
the caller using gss_release_buffer().

```
OM_uint32 gss_get_name_attribute(
  OM_uint32                       *minor_status,
  gss_const_name_t                name,
  gss_const_buffer_t              attr,
  int                             *authenticated,
  int                             *complete,
  gss_buffer_t                    value,
  gss_buffer_t                    display_value,
  int                             *more
);
```

7.6.  GSS_Set_name_attribute()

   Inputs:

   o  name INTERNAL NAME

   o  complete BOOLEAN -- TRUE if and only if this represents a complete
      set of values for the name

   o  attr OCTET STRING

   o  values SET OF OCTET STRING

   Outputs:

   o  major_status INTEGER

   o  minor_status INTEGER

   Return major_status codes:

   o  GSS_S_COMPLETE indicates no error.

   o  GSS_S_UNAVAILABLE indicates that the given attribute NAME is not
      known or could not be set.

   o  GSS_S_FAILURE indicates a general error.

   When the given NAME object is an MN, this function MUST fail (with
   GSS_S_FAILURE) if the mechanism for which the name is an MN does not
   recognize the attribute name or the namespace it belongs to.  This is
   because name attributes generally have some semantics that mechanisms
   must understand.

   On the other hand, when the given name is not an MN, this function
   MAY succeed even if none of the available mechanisms understand the
   given attribute, in which subsequent credential acquisition attempts

   (via GSS_Acquire_cred() or GSS_Add_cred()) with the resulting name
   MUST fail for mechanisms that do not understand any one or more name
   attributes set with this function.  Applications may wish to use a
   non-MN, then acquire a credential with that name as the desired name.
   The acquired credentials will have elements only for the mechanisms
   that can carry the name attributes set on the name.

   Note that this means that all name attributes are locally critical:
   the mechanism(s) must understand them.  The reason for this is that
   name attributes must necessarily have some meaning that the mechanism
   must understand, even in the case of application-specific attributes
   (in which case the mechanism must know to transport the attribute to
   any peer).  However, there is no provision to ensure that peers
   understand any given name attribute.  Individual name attributes may
   be critical with respect to peers, and the specification of the
   attribute will have to indicate whether the mechanism's protocol or
   the application is expected to enforce criticality.

   The complete flag denotes that (if TRUE) the set of values represents
   a complete set of values for this name.  The peer being an
   authoritative source of information for this attribute is a
   sufficient condition for the complete flag to be set by the peer.

   In the federated case, when several peers may hold some of the
   attributes about a name, this flag may be highly dangerous and SHOULD
   NOT be used.

   NOTE: This function relies on the GSS-API notion of "SET OF" allowing
   for order preservation; this has been discussed on the KITTEN WG
   mailing list, and the consensus seems to be that, indeed, that was
   always the intention.  It should be noted that underlying mechanisms
   may not respect the given order.

7.6.1.  C-Bindings

   The C-bindings of GSS_Set_name_attribute() requires one function call
   per attribute value for multi-valued name attributes.  Each call adds
   one value.  To replace an attribute's every value, delete the
   attribute's values first with GSS_Delete_name_attribute().

```
   OM_uint32 gss_set_name_attribute(
     OM_uint32                     *minor_status,
     gss_const_name_t              name,
     int                           complete,
     gss_const_buffer_t            attr,
     gss_const_buffer_t            value
   );
```

7.7.  GSS_Delete_name_attribute()

   Inputs:

   o  name INTERNAL NAME

   o  attr OCTET STRING

   Outputs:

   o  major_status INTEGER

   o  minor_status INTEGER

   Return major_status codes:

   o  GSS_S_COMPLETE indicates no error.

   o  GSS_S_UNAVAILABLE indicates that the given attribute NAME is not
      known.

   o  GSS_S_UNAUTHORIZED indicates that a forbidden delete operation was
      attempted, such as deleting a negative attribute.

   o  GSS_S_FAILURE indicates a general error.

   Deletion of negative authenticated attributes from NAME objects MUST
   NOT be allowed and must result in a GSS_S_UNAUTHORIZED.

7.7.1.  C-Bindings

```
   OM_uint32 gss_delete_name_attribute(
     OM_uint32                       *minor_status,
     gss_const_name_t                name,
     gss_const_buffer_t              attr
   );
```

7.8.  GSS_Export_name_composite()

   Inputs:

   o  name INTERNAL NAME

   Outputs:

   o  major_status INTEGER

   o  minor_status INTEGER

   o  exp_composite_name OCTET STRING -- the caller is responsible for
      de-allocating memory using GSS_Release_buffer

   Return major_status codes:

   o  GSS_S_COMPLETE indicates no error.

   o  GSS_S_FAILURE indicates a general error.

   This function outputs a token that can be imported with
   GSS_Import_name(), using GSS_C_NT_COMPOSITE_EXPORT as the name type
   and that preserves any name attribute information (including the
   authenticated/complete flags) associated with the input name (which
   GSS_Export_name() may well not).  The token format is not specified
   here as this facility is intended for inter-process communication
   only; however, all such tokens MUST start with a two-octet token ID,
   hex 04 02, in network byte order.

   The OID for GSS_C_NT_COMPOSITE_EXPORT is 1.3.6.1.5.6.6.

7.8.1.  C-Bindings

   The "exp_composite_name" buffer is de-allocated by the caller with
   gss_release_buffer.

   OM_uint32 gss_export_name_composite(
     OM_uint32                     *minor_status,
     gss_const_name_t              name,
     gss_buffer_t                  exp_composite_name
   );

8.  IANA Considerations

   IANA has registered a new name-type OID in "SMI Security for Name
   System Designators Codes (nametypes)":

      6  gss-composite-export [RFC6680]

   (The absolute OID is 1.3.6.1.5.6.6.)

   This document creates a namespace of GSS-API name attributes.
   Attributes are named by URIs, so no single authority is technically
   needed for allocation.  However, future deployment experience may
   indicate the need for an IANA registry for URIs used to reference
   names specified by IETF standards.  It is expected that this will be
   a registry of URNs, but this document provides no further guidance on
   this registry.

9.  Security Considerations

   This document extends the GSS-API naming model to include support for
   name attributes.  The intention is that name attributes are to be
   used as a basis for (among other things) authorization decisions or
   personalization for applications relying on GSS-API security
   contexts.

   The security of the application may be critically dependent on the
   security of the attributes.  This document classifies attributes as
   asserted or authenticated.  Asserted (non-authenticated) attributes
   MUST NOT be used if the attribute has security implications for the
   application (e.g., authorization decisions) since asserted attributes
   may easily be controlled by the peer directly.

   It is important to understand the meaning of "authenticated" in this
   setting.  Authenticated does not imply that any semantic of the
   attribute is claimed to be true.  The only implication is that a
   trusted third party has asserted the attribute as opposed to the
   attribute being asserted by the peer itself.  Any additional
   semantics are always the result of applying policy.  For instance, in
   a given deployment, the mail attribute of the subject may be
   authenticated and sourced from an email system where "authoritative"
   values are kept.  In another situation, users may be allowed to
   modify their mail addresses freely.  In both cases, the "mail"
   attribute may be authenticated by virtue of being included in signed
   SAML attribute assertions or by other means authenticated by the
   underlying mechanism.

   When the underlying security mechanism does not provide a permanent
   unique identity (e.g., anonymous Kerberos), GSS-API naming extensions
   may be used to provide a permanent unique identity attribute.  This
   may be a globally unique identifier, a value unique within the
   namespace of the attribute issuer, or a "directed" identifier that is
   unique per peer acceptor identity.  SAML, to use one example
   technology, offers a number of built-in constructs for this purpose,
   such as a <NameID> with a Format of
   "urn:oasis:names:tc:SAML:2.0:nameid-format:persistent".  SAML
   deployments also typically make use of domain-specific attribute
   types that can serve as identifiers.

10.  References

10.1.  Normative References

   [GFD.024]  Meder, S., Welch, V., Tuecke, S., and D. Engert, "GSS-API
              Extensions", Global Grid Forum GFD.024, June 2004,
              <http://www.ggf.org/documents/GFD.24.pdf>.

   [RFC2119]  Bradner, S., "Key words for use in RFCs to Indicate
              Requirement Levels", BCP 14, RFC 2119, March 1997.

   [RFC2743]  Linn, J., "Generic Security Service Application Program
              Interface Version 2, Update 1", RFC 2743, January 2000.

   [RFC2744]  Wray, J., "Generic Security Service API Version 2 :
              C-bindings", RFC 2744, January 2000.

   [RFC5587]  Williams, N., "Extended Generic Security Service Mechanism
              Inquiry APIs", RFC 5587, July 2009.

10.2.  Informative References

   [ANSI.X3-4.1986]
              American National Standards Institute, "Coded Character
              Set - 7-bit American Standard Code for Information
              Interchange", ANSI X3.4, 1986.

   [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.

   [RFC3629]  Yergeau, F., "UTF-8, a transformation format of ISO
              10646", STD 63, RFC 3629, November 2003.

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

   [RFC4556]  Zhu, L. and B. Tung, "Public Key Cryptography for Initial
              Authentication in Kerberos (PKINIT)", RFC 4556, June 2006.

   [RFC4768]  Hartman, S., "Desired Enhancements to Generic Security
              Services Application Program Interface (GSS-API) Version 3
              Naming", RFC 4768, December 2006.

Authors' Addresses

   Nicolas Williams
   Cryptonector, LLC

   EMail: nico@cryptonector.com


   Leif Johansson
   Swedish University Network
   Thulegatan 11
   Stockholm
   Sweden

   EMail: leifj@sunet.se
   URI:   http://www.sunet.se


   Sam Hartman
   Painless Security

   EMail: hartmans-ietf@mit.edu


   Simon Josefsson
   SJD AB
   Johan Olof Wallins Vaeg 13
   171 64 Solna
   Sweden

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