

Network Working Group
Internet Draft
Intended status: Informational
Expires: April 21, 2012

H. Kaplan
Acme Packet
October 15, 2011

A Taxonomy of Session Initiation Protocol (SIP)
Back-to-Back User Agents
draft-kaplan-dispatch-b2bua-taxonomy-00

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on April 15, 2011.

Copyright and License Notice

Copyright (c) 2010 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 BSD License.

There are numerous types of SIP Back-to-Back User Agents (B2BUAs), performing different roles in different ways. This document identifies several common B2BUA roles, in order to provide taxonomy other documents can use and reference.

Table of Contents

1.	Terminology.....	2
2.	Introduction.....	3
3.	B2BUA Role Types.....	3
3.1.	Signaling-plane B2BUA Roles.....	4
3.1.1	Proxy-B2BUA.....	4
3.1.2	Signaling-only.....	4
3.2.	Media-plane B2BUA Roles.....	4
3.2.1	Media-relay.....	5
3.2.2	Media-aware.....	5
3.2.3	Media-termination.....	5
4.	Mapping SIP Device Types to B2BUA Roles.....	6
4.1.	SIP PBXs and Softswitches.....	6
4.2.	Application Servers.....	6
4.3.	Session Border Controllers.....	6
4.4.	Transcoders.....	7
4.5.	Conference Servers.....	7
4.6.	P-CSCF and IBCF Functions.....	7
4.7.	S-CSCF Function.....	7
5.	Security Considerations.....	7
6.	IANA Considerations.....	8
7.	Acknowledgments.....	8
8.	References.....	8
8.1.	Informative References.....	8
	Author's Address.....	8

1. Terminology

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. The terminology in this document conforms to RFC 2828, "Internet Security Glossary".

B2BUA: a SIP Back-to-Back User Agent, which is the logical combination of a User Agent Server (UAS) and User Agent Client (UAC).

UAS: a SIP User Agent Server.

UAC: a SIP User Agent Client.

2. Introduction

In many SIP deployments, SIP entities exist in the SIP signaling path between the originating UAC and final terminating UAS, which go beyond the definition of a Proxy, performing functions not defined in standards-track RFCs. The only term for such devices provided in [RFC3261] is for a Back-to-Back User Agent (B2BUA), which is defined as the logical concatenation of a User Agent Server (UAS) and User Agent Client (UAC).

In practice, there are numerous forms of B2BUAs, operating at various layers of the protocol stack, and for various purposes, and with widely varying behavior from a SIP protocol perspective. Some act as pure SIP Proxies and only change to the role of B2BUA in order to generate BYEs to terminate dead sessions. Some are full User Agent stacks with only high-level event and application logic binding the UAS and UAC sides. Some B2BUAs operate only in the SIP signaling plane, while others participate in the media plane as well.

As more and more SIP domains get deployed and interconnect, the odds of a SIP session crossing such media-plane B2BUAs increases, as well as the number of such B2BUAs any given SIP session may go through. In other words, any given SIP session may cross any number of B2BUAs both in the SIP signaling plane as well as media-plane.

This document provides a taxonomy of several common B2BUA roles, so that other documents may refer to them using their given names without re-defining them in each document.

3. B2BUA Role Types

Within the context of this document, the terms refer to a B2BUA role, not a particular system type. A given system type may change its role in the middle of a SIP session, for example when a Stateful Proxy tears-down a session by generating BYEs; or for example when an SBC performs transcoding or REFER termination.

Furthermore, this document defines 'B2BUA' following the definition provided in [RFC3261], which is the logical concatenation of a UAS and UAC. A typical centralized conference server, for example, is not a B2BUA because it is the target UAS of multiple UACs, whereby the UACs individually and independently initiate separate SIP sessions to the central conference server. Likewise, a one-armed third-party call control transcoder as described in section 3.1 of

[RFC5369] is not a B2BUA; whereas an inline transcoder based on [RFC5370] is a B2BUA.

3.1. Signaling-plane B2BUA Roles

A Signaling-plane B2BUA is one that operates only on the SIP message protocol layer, and only with SIP messages and headers but not the media itself in any way. This implies it does not modify SDP bodies, although it may save them and/or operate on other MIME body types. This category is further subdivided into specific roles as described in this section.

3.1.1 Proxy-B2BUA

A Proxy-B2BUA is one that appears, from a SIP protocol perspective, to be a SIP Proxy based on [RFC3261] and its extensions, except that it maintains sufficient dialog state to generate in-dialog SIP messages on its own and does so in specific cases. The most common example of this is a SIP Proxy which can generate BYE requests to tear-down a dead session. Another example would be a Proxy which can generate UPDATE requests to test the liveness of a session.

A Proxy-B2BUA does not modify the received header fields such as the To, From, or Contact, and only modifies the Via and Record-Route header fields following the rules in [RFC3261] and its extensions. A Proxy-B2BUA neither modifies nor inspects MIME bodies (including SDP), does not have any awareness of media, will forward any Method type, etc.

3.1.2 Signaling-only

A Signaling-only B2BUA is one that operates at the SIP layer but in ways beyond those of [RFC3261] Proxies, even for normally forwarded requests. Such a B2BUA may or may not replace the Contact URI, modify or remove all Via and Record-Route headers, modify the To and From header fields, modify or inspect specific MIME bodies, etc. No SIP header field is guaranteed to be copied from the received request on the UAS side to the generated request on the UAC side.

An example of such a B2BUA would be some forms of Application Servers and PBXs, such as a server which locally processes REFER requests and generates new INVITEs on behalf of the REFER's target. Another example would be an [RFC3323] Privacy Service Proxy performing the 'header' privacy function.

3.2. Media-plane B2BUA Roles

A Media-plane B2BUA is one that operates at both the SIP and media planes, not only on SIP messages but also SDP and potentially

RTP/RTCP or other media. Such a B2BUA may or may not replace the Contact URI, modify or remove all Via and Record-Route headers, modify the To and From header fields, etc. No SIP header field is guaranteed to be copied from the received request on the UAS side to the generated request on the UAC side, and SDP will also be modified.

An example of such a B2BUA would be a Session Border Controller performing the functions defined in [RFC5853], a B2BUA transcoder as defined in [RFC5370], a rich-ringtone Application Server, or a recording system. Another example would be an [RFC3323] Privacy Service Proxy performing the 'session' privacy function.

Note that a Media-plane B2BUA need not be instantiated in a single physical system, but may be decomposed into separate signaling and media functions.

The Media-plane B2BUA category is further subdivided into specific roles as described in this section.

3.2.1 Media-relay

A B2BUA which performs a media-relay role is one that terminates the media plane at the IP and UDP/TCP layers on its UAS and UAC sides, but neither modifies nor restricts which forms of UDP or TCP payload are carried within the UDP or TCP packets. Such a role may only support UDP or only TCP or both, as well as other transport types or not. Such a role may involve policing the IP packets to fit within a bandwidth limit, or converting from IPv4 to IPV6 or vice-versa. This is typically similar to a NAT behavior, except a NAT operating in both directions on both the source and destination information; it is often found as the default behavior in SBCs.

3.2.2 Media-aware

A B2BUA which performs a media-aware role is similar to a media-relay except that it inspects and potentially modifies the payload carried in UDP or TCP, such as RTP or RTCP, but not at a codec or higher layer. An example of such a role is an SRTP terminator, which does not need to care about the RTP payload but does care about the RTP header; or a device which monitors RTCP for QoS information; or a device which muxes/de-muxes RTP and RTCP packets on the same 5-tuple.

3.2.3 Media-termination

A B2BUA which performs a media-termination role is one that operates at the media payload layer, such as RTP/RTCP codec or MSRP message layer and higher. Such a role may only terminate/generate specific

RTP media, such as [RFC4733] DTMF packets, or it may convert between media codecs, or act as a Back-To-Back MSRP agent. This is the role performed by transcoders, conference servers, etc.

4. Mapping SIP Device Types to B2BUA Roles

Although the B2BUA role types defined previously do not define a system type, as discussed in section 3, some discussion of what common system types perform which defined roles may be appropriate. This section provides such a 'mapping' for general cases, to aid in understanding of the roles.

4.1. SIP PBXs and Softswitches

SIP-enabled Private Branch eXchanges (SIP PBXs) and Softswitches are market category terms, and not specified in any standard. In general they can perform every role described in this document at any given time, based on their architecture or local policy. Some are based on architectures that make them the equivalent of a SIP UAS and UAC connected with a logical PRI loopback; others are built as a SIP Proxy core with optional modules to "do more".

4.2. Application Servers

Application Servers are a broad marketing term, and not specified in any standard in general, although 3GPP and other organizations specify some specific Application Server functions and behaviors. Common examples of Application Servers functions are message-waiting indication, find-me-follow-me services, privacy services, call-center ACD services, call screening, and VCC services. Some only operate in the signaling plane in either Proxy-B2BUA or Signaling-only B2BUA roles; others operate as full Media-termination B2BUAs, such as when providing IVR, rich-ringtone or integrated voicemail services.

4.3. Session Border Controllers

Session Border Controllers (SBCs) are a market category term, and not specified in any standard. Some of the common functions performed by SBCs are described in [RFC5853], but in general they can perform every role described in this document at any given time, based on local policy. By default, most SBCs are either Media-relay or Media-aware B2BUAs, and replace the Contact URI, remove the Via and Record-Route headers, modify the Call-ID, To, From, and various other headers, and modify SDP. Some SBCs remove all headers, all bodies, and reject all Method types unless explicitly allowed by local policy; other SBCs pass all such elements through unless explicitly forbidden by local policy.

Transcoders perform the function of transcoding one audio or video media codec type to another, such as G.711 to G.729. As such they perform the Media-termination role, although they may only terminate media in specific cases of codec mismatch between the two ends. Although [RFC5369] and [RFC5370] define two types of SIP Transcoders, in practice a popular variant of the [RFC5370] inline model is to behave as a SIP B2BUA without using the resource-list mechanism, but rather simply route a normal INVITE request through an inline transcoder. SIP Transcoders architectures are based on everything from SIP media servers, to SBCs, to looped-back TDM gateways, and thus run the gamut from replacing only specific headers/bodies and SDP content needed to perform their function, to replacing almost all SIP headers and SDP content. Some transcoders save and remove SDP offers in INVITES form the UAC, and wait for an offer in the response from the UAS, similar to a 3PCC model; others just insert additional codecs in SDP offers and only transcode if the inserted codec(s) are selected in the answer.

4.5. Conference Servers

In general Conference Servers do not fall under the term 'B2BUA' as defined by this document, since typically they involve multiple SIP UACs initiating independent SIP sessions to the single conference server UAS. However, a conference server supporting [RFC5366], whereby the received INVITE triggers the conference focus UAS to initiate multiple INVITES as a UAC, would be in a Media-termination B2BUA role when performing that function.

4.6. P-CSCF and IBCF Functions

Proxy-CSCFs and IBCFs are functions defined by 3GPP IMS standards, and when coupled with the IMS media-plane gateways (IMS AGW, TrGW, etc.) typically form a logical Media-relay or Media-aware B2BUA role.

4.7. S-CSCF Function

S-CSCF is a function defined by 3GPP IMS standards, and typically follows a Proxy-B2BUA role.

5. Security Considerations

There are no security implications for this document, as it is only a taxonomy.

6. IANA Considerations

This document makes no request of IANA.

7. Acknowledgments

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).

8. References

8.1. Informative References

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[RFC3323] Peterson, J., "A Privacy Mechanism for the Session Initiation Protocol (SIP)", RFC 3323, November 2002.

[RFC4733] Schulzrinne, H., Taylor, T., "RTP Payload for DTMF Digits, Telephony Tones, and Telephony Signals", RFC 4733, December 2006.

[RFC5366] Camarillo, G., Johnston, A., "Conference Establishment Using Request-Contained Lists in the Session Initiation Protocol (SIP)", RFC 5366, October 2008.

[RFC5369] Camarillo, G., "Framework for Transcoding with the Session Initiation Protocol (SIP)", RFC 5369, October 2008.

[RFC5370] Camarillo, G., "The Session Initiation Protocol (SIP) Conference Bridge Transcoding Model", RFC 5370, October 2008.

[RFC5853] Hautakorpi, J, et al, "Requirements from Session Initiation Protocol (SIP) Session Border Control (SBC) Deployments", RFC 5853, April 2010.

Author's Address

Hadriel Kaplan
Acme Packet
Email: hkaplan@acmepacket.com

Dispatch
Internet-Draft
Intended status: Standards Track
Expires: April 22, 2012

K. Ono
H. Schulzrinne
Columbia University
October 20, 2011

Referencing and Validating User Attributes
draft-ono-dispatch-attribute-validation-00.txt

Abstract

This document describes a mechanism for referencing and validating user attributes in SIP communication. User attributes, such as an organizational affiliation and role, are helpful for the recipients of a communication request to decide whether or not to grant the sender access to the recipient's resources, especially when the sender identity is unknown to the recipients. This mechanism allows the sender to claim her attributes to recipients using an attribute reference identifier without needing to prove the sender identity. This document defines a new SIP "Sender-References" header field to convey one or more attribute reference identifiers. This mechanism satisfies all the requirements for trait-based authorization defined in RFC 4484, except that it provides only one assertion scheme.

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 22, 2012.

Copyright Notice

Copyright (c) 2011 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	3
2. Terminology	4
3. Architecture	4
3.1. Assumed Trust Relationships among AVS, Caller, and Callee	5
3.2. ARIDs are Loosely Associated with the Owner's Identity in SIP	6
4. Requirements	7
4.1. Differences between Our Requirements and the Requirements for Trait-Based Authorization	7
5. Procedures	8
5.1. Generating an ARID	9
5.2. Obtaining an ARID	10
5.3. Sending an ARID in a Communication Request	12
5.4. Validating an ARID to Retrieve User Attributes	12
6. Sender-References Header Field	14
7. Relationship to Existing Mechanisms	14
8. Security Considerations	17
8.1. Man in the Middle Attacks	17
8.2. Replay Attacks Using a Received ARID	17
8.3. Denial of Service Attacks on the AVS	18
8.4. Phishing Attacks on the AVS	18
9. IANA Considerations	18
10. References	19
10.1. Normative References	19
10.2. Informative References	19
Authors' Addresses	20

1. Introduction

Ascertaining a person's attributes is often useful to determine the trustworthiness of the person when two people first meet each other. These user attributes include, for example, an organizational affiliation, a role in a professional society, age, holding certificates or licenses, and being a customer of a bank, an employee, or a student. If user attributes are available with a communication request, these attributes can help the recipient determine how to handle the communication request by estimating whether the communication is important enough to be established.

A caller identifier (ID) authenticated by the SIP Identity mechanism [RFC4474], when used alone, can be a helpful user attribute, but only in limited cases. Only if a caller ID is in a SIP-URI [RFC3261] and is authenticated by the domain of a trusted organization can the caller ID be perceived as evidence that the caller belongs to the trusted organization. However, if a caller ID in a SIP-URI belongs to an untrusted domain regarding user admission policy, such as a free voice over IP service provider, or if a caller ID does not contain any domain name, such as a tel-URI [RFC3966], the caller ID does not indicate the caller's trustworthiness to the callee who has never seen the caller ID before. Thus, even if a caller has multiple contact addresses, the caller needs to use a contact address issued by a trusted domain for authorization purposes. To offer a flexible choice of which contact address to use, our referencing mechanism introduces another piece of information, an attribute reference ID (ARID), that enables a caller to refer to her attributes without needing to rely on the caller ID. A caller can use multiple ARIDs if the caller wants to prove multiple attributes associated with different organizations. This referencing mechanism, unlike the caller ID, allows a caller to use multiple ARIDs to declare multiple user attributes in a single communication request.

If an authenticated caller ID does not provide sufficient information, the callee can look up further user attributes through directory services. However, a reference integrity problem arises when a directory service does not allow queriers to look up user attributes by the user's contact address. Additionally, when a directory service allows queries by a user's contact address, but is offered by a third party, not the issuer of contact addresses, the authenticity of the information is unreliable. For example, DoctorFinder service offered by the American Medical Association provides information about certified medical doctors. When making a query, a querier cannot use the doctor's phone number, but needs to use doctor's common name, street address or specialty, which is available to the public. If a doctor makes a call (or sends an email message) that includes such query information and a reference to the

DoctorFinder service, the callee (or the recipient) is not convinced of the certainty. To solve this reference integrity problem, our referencing mechanism allows an organization to generate a short-lived ARID upon a caller request. This ARID is effective only for a specific communication by limiting the lifetime and encoding designated destinations, namely designated queriers. In addition, the ARID can be used only for retrieving the attributes that the caller selects to disclose to the specific queriers.

2. Terminology

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. Architecture

Figure 1 depicts an overview of the service architecture where an attribute validation server (AVS) operates to reference and validate user attributes for an organization. For each user, the AVS maintains the username and credentials to authenticate the user for remote access, in addition to other information such as a user number and role which the organization assigns, the users's common name, affiliation, street address, and electronic contact addresses. Note that the AVS stores a user's contact addresses, but it neither guarantees that the user owns the contact addresses nor can be reached by their addresses.

We provide an example for illustration. Alice, a user of services provided by the organization, "example.org", is about to make a call to Bob at "bob@example.com". Alice first requests an ARID from the AVS using HTTP [RFC2616] over TLS [RFC5246]. When sending the request, Alice authenticates the AVS using its X.509 Public Key Certificate (PKC) [RFC5280] which is delivered in the TLS handshake and is signed by a trusted Certificate Authority (CA). In turn, when generating an ARID for Alice, the AVS authenticates her using any credentials supported by the AVS, such as a password or a client's X.509 PKC. Upon successfully obtaining an ARID, Alice makes a call to Bob using SIP [RFC3261] over TLS. The SIP INVITE request includes the ARID. When Bob receives an ARID, he queries the validity of the ARID to the AVS using HTTP over TLS. Bob authenticates the AVS using its X.509 PKC in the same way that Alice does. Based on the query results, Bob determines whether or not to answer the call from Alice and adjusts his communication stance accordingly.

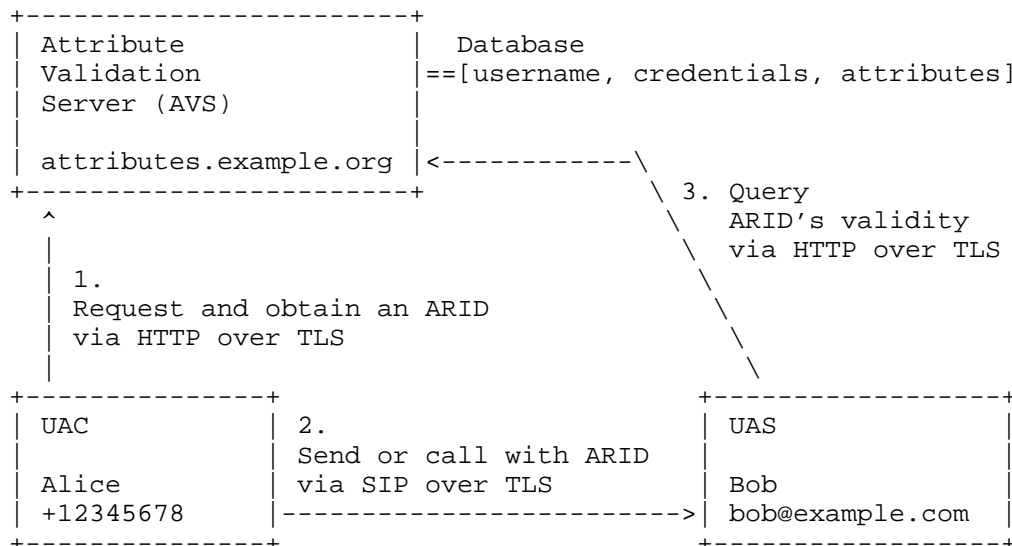


Figure 1: Architecture

3.1. Assumed Trust Relationships among AVS, Caller, and Callee

We assume that the AVS and the caller, Alice, trust each other regarding the attribute validation service for an organization, "example.org." They share Alice's username and credentials for remote access, and her attributes. Alice trusts the AVS to properly maintain her attributes and to disclose the attributes she selects only to queriers whom she specifies. In turn, the AVS trusts Alice as a user in the organization and trusts her attributes which it knows first-hand, such as "Alice is an IEEE student member." However, the AVS does not know the authenticity of her attributes that are not issued by the organization, such as her common name, affiliation, and contact addresses.

We also assume that Bob knows "example.org" as the domain name of an organization that has a user admission policy he trusts, whether or not he belongs to the organization. Bob also trusts the AVS to properly perform its attribute validation service.

Alice finds Bob worth making a call and disclosing her attributes to establish a communication with him. In turn, Bob does not have sufficient information about Alice's trustworthiness based solely on her identity in a SIP communication request.

3.2. ARIDs are Loosely Associated with the Owner's Identity in SIP

An ARID generated upon Alice's request can be used only to retrieve her attributes, but the ARID is not tightly linked with her identity used in a SIP communication request, namely the caller ID in a call. When Bob receives an ARID in a SIP communication request where the message integrity is protected by TLS, the callee can perceive the ARID to be associated with the caller ID. Bob can loosely link an ARID with the owner's identity only because of the fact that these two pieces of information are sent in the same message. Other than the presence of these two pieces of information in the same message, there is no linkage between the ARID and the caller ID. Bob does not need to provide Alice's caller ID to validate a received ARID. The user attributes Bob retrieves upon the success of the validation do not contain the owner's contact address. This loose linkage is a natural consequence of the general fact that user attributes and the user identifier in a communication are often issued separately by different organizations or services.

This loose linkage, however, makes it difficult for Bob to detect impersonation using a stolen ARID. Bob cannot detect this impersonation by providing the AVS with the owner's caller ID or by being presented the caller ID in user attributes. When issuing an ARID, the AVS cannot easily authenticate her caller ID since the caller ID is issued by a different administrative domain. Additionally, Bob cannot always authenticate the caller ID. The cases where no authentication of the caller ID is available include where a caller ID is in a SIP-URI issued by the domain which does not deploy the SIP Identity mechanism, where a caller ID is in a tel-URI which is sent without any other authentication mechanisms, such as a digital signature in S/MIME [RFC5751], and where a caller ID is anonymized. Thus, although tightening this linkage can protect from impersonation attacks, it makes the service deployment more difficult and limits the caller's choice of caller IDs.

To mitigate the vulnerability to impersonation attacks using a stolen ARID without tying an ARID to an authenticated caller ID, a countermeasure is devised for each vulnerable target. To prevent a man in the middle from eavesdropping on an ARID, all the connection links to convey an ARID need to be protected with TLS. To detect that an ARID was stolen from the owner, the recipients, or intermediaries, such as a SIP proxy server, an ARID can be used to retrieve user attributes only a limited number of times, for a limited time period, and by limited queriers. Yet even with these protections, this mechanism cannot prevent the owner of an ARID from giving her own ARID to others. To keep this mechanism simple, we do not include any additional mechanisms that discourage the owner from giving her own ARID. As a result, this mechanism allows the owner of

an ARID to informally delegate her attributes to others without proving the chain of authorizing delegation. However, a legitimate recipient cannot impersonate Alice's attributes by forwarding a received ARID.

4. Requirements

This section first identifies the requirements of a mechanism for referencing and validating attributes, and then identifies differences between these requirements and the requirements for Trait-Based Authorization (TBA) for SIP [RFC4484].

Our requirements are:

- REQ-1: The mechanism must enable a user to prove one or more attributes by presenting an attribute reference ID (ARID).
- REQ-2: The mechanism must allow a user to prove her attributes in one or more organizations in a single communication request.
- REQ-3: The mechanism must allow a user to specify her attributes to be disclosed for each communication session.
- REQ-4: The mechanism must allow a user to restrict queriers who can retrieve her attributes to the recipients of a communication request.
- REQ-5: This mechanism must adapt to various attribute policies; thus, an ARID must be temporary rather than persistent.
- REQ-6: The mechanism must allow the recipients of an ARID to easily validate a received ARID.
- REQ-7: The mechanism must prevent the recipients of an ARID from impersonation by forwarding a received ARID.
- REQ-8: The mechanism must protect user's private information, such as communication history, even against an AVS.
- REQ-9: This mechanism should provide flexibility for deployment; thus, an ARID should be unique across different organizations deployed on a single AVS.

We intentionally omit the following requirements:

- o The mechanism does not need to prevent a user from giving her ARID to others.
- o The mechanism does not need to support a user who delegates the ARID with proving the chain of authorizing delegation.
- o The mechanism does not need to bind an ARID to the SIP signaling path or SIP identity.

4.1. Differences between Our Requirements and the Requirements for Trait-Based Authorization

Our requirements described above are similar to the TBA requirements for SIP, but two differences exist. First, we do not require support

for optional assertion schemes other than an ARID defined in Section 5 while the TBA includes the following requirement:

7. The mechanism MUST have a single baseline mandatory-to-implement authorization assertion scheme. The mechanism MUST also allow support of other assertion schemes, which would be optional to implement. One example of an assertion scheme is Security Assertion Markup Language (SAML) [6] and another is RFC 3281 X.509 Attribute Certificates [7].

Our mechanism currently does not support other assertion schemes, such as SAML [SAML] or X.509 Attribute Certificates (AC) [RFC5755], as mentioned above. Such mechanisms that protect assertion integrity by signing using the issuer's private key requires that recipients verify the integrity using the issuer's public key in the application layer. The recipients also need to authenticate the issuer of an assertion. On the other hand, our mechanism relies on transport layer security, namely TLS, to protect message integrity and authenticate the issuer of an ARID. Although our mechanism does not separately protect the integrity of user attributes or the linkage between user attributes and their owner, our mechanism instead protects the integrity of a whole message including these attributes. As long as intermediaries such as an HTTP and SIP proxy servers can be trusted to properly transfer messages for this attribute referencing service, this security with TLS is simpler, and strong enough against message tampering and server impersonation.

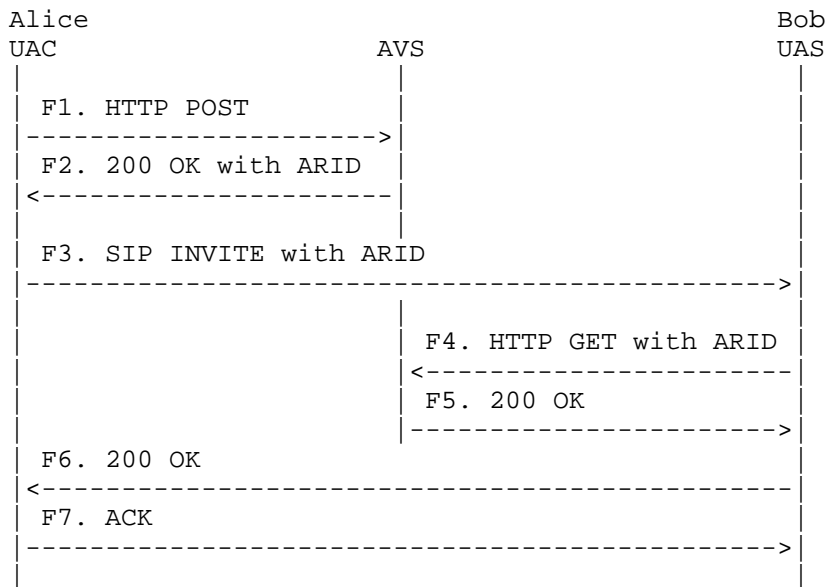
The second difference is that our requirements include an additional requirement for protecting user's privacy described in REQ-8. Although an authorization service or AVS needs to limit designated queriers to the designated destinations of a SIP request, the authorization service has to know neither user's communication history nor plans containing routable contact addresses to do so even for a short term during the lifetime of an assertion or ARID. Our mechanism hashes contact addresses to prevent this unnecessary disclosure of the private information of a user.

5. Procedures

Figure 2 illustrates message exchanges among a UAC, the UAS and the AVS for the following procedures:

1. Obtaining an ARID;
2. Sending the ARID when making a call using SIP;
3. Validating the ARID to retrieve user attributes.

Before explaining each procedure, we describe how the AVS typically generates an ARID.



Note: SIP messages to/from SIP proxy servers are omitted since they are not affected by this mechanism.

Figure 2: Message Exchanges

5.1. Generating an ARID

An ARID is a string of URL [RFC3986] characters generated by an AVS upon a user's request. When a single AVS offers this attribute service for multiple organizations, a subdomain or a path in the URL of the AVS website is assigned to each organization as part of an ARID to meet the requirement REQ-9.

We show two examples how an AVS generates an ARID. Note that the AVS does not have to follow these generating mechanisms. The first example is to hash a string of characters by SHA1 [SHA1]. The string of characters is a user number concatenated with the timestamp, a nonce, and hashed contact addresses of one or more desired queriers (REQ-4,8) as shown below. Hashed contact addresses of one or more desired queriers are sent from a user when the user requests an ARID, as described in Section 5.2. The information other than these hashed contact addresses is stored or generated on the AVS.

Generating an ARID by hash:

```
m = user number || timestamp || nonce || hashed querier's contact address
```

If two queriers, querier_1 and querier_2, are specified,

```
m = user number || timestamp || nonce || hashed querier_1's  
contact address;hashed querier_2's contact address
```

```
ARID = URL path/Hash(m)
```

Another example is to encrypt a string of characters with a symmetric key of the AVS using AES [AES]. The string of characters is a user number concatenated with a disclosure mode, the expiry time, hashed contact addresses of desired queriers. The disclosure mode is determined what attributes a user discloses to desired queriers (REQ-3). The expiry time of an ARID needs to be shortly after the time an ARID is generated, such as ten minutes later, to avoid replay attacks (REQ-7).

An appropriate expiry time depends on the service type. For synchronous communication services, such as a voice or video call or real-time text chat, the lifetime needs to be short. For asynchronous services, such as instant messaging, or email communication, the lifetime needs to be longer, such as 24 hours.

Generating an ARID by encryption:

```
m = user number || disclosure mode || expiry time || salt || hashed  
querier's contact address
```

If two queriers, querier_1 and querier_2, are specified,

```
m = user_id || disclosure_mode || expiry time || salt || hashed  
querier_1's contact address;hashed querier_2's contact address
```

```
ARID = URL path/Encrypt(m)
```

When selecting a method for generating an ARID, by hash or encryption, they have the trade-off between the memory cost of storing ARIDs with related data and the computational cost of decrypting ARIDs. When generating an ARID by hash, the AVS needs to store the generated ARID with associated data including the expiry time, the nonce, the hashed contact addresses of desired queriers which the user sent, and the disclosure mode which the user specified. On the other hand, when generating an ARID by encryption, the AVS only needs to remember the salt for decryption, but not any generated ARIDs. Instead, it requires the computational cost of decryption.

5.2. Obtaining an ARID

To obtain an ARID which can be used for a communication with Bob, Alice first needs to connect to the AVS using a SIP UA which supports this mechanism. When connecting, the SIP UAC MUST authenticate the AVS using its X.509 PKC sent in the TLS handshake. In turn, the AVS MUST authenticate Alice using her username and credentials. For user authentication, HTTP Basic or Digest authentication [RFC2617], a

client's PKC, or other mechanisms SHOULD be used. Upon successful user authentication, the SIP UAC MUST send the AVS an HTTP POST request with setting hashed Bob's contact address as a desired querier, and a disclosure mode in a message body, as shown in the following example. Each hashed contact address of a desired querier SHOULD be attached as a JSON [RFC4627] object, or MAY be in XML [XML]. When a communication request has multiple destinations, such as a conference call, multiple "destination" fields SHOULD be included to contain multiple hashed contact addresses of the desired queriers.

F1. HTTP POST sent from Alice to AVS:

```
POST /requestARID HTTP/1.1
HOST:attributes.example.org
Content-Type:application/json
```

```
{"destination":"2cf6aleda3b5205005d25a7d5dcf13bb200fc26a",\
"disclosure_mode":"details"}
```

Note: Mandatory HTTP or SIP headers unrelated to this mechanism are not shown here and the following example messages.

Hashing the contact address of a desired querier is to limit acceptable queriers without revealing communication history to the AVS (REQ-8). The SIP UA supporting this mechanism MUST implement and use SHA1, and MAY support any other hash algorithms. To prevent re-identification based on hashed contact addresses collected on the AVS, the SIP UAC MUST generate a salt, which is a random string of characters, and concatenate it with a contact address as follows:

Hash(salt || contact address)

In the example above, the destination field, "2cf6aleda3b5205005d25a7d5dcf13bb200fc26a", is generated by SHA1("dmvblp03" || "sips:bob@example.com").

When the AVS successfully generates an ARID for Alice, the AVS responds to her with a 200 OK response including the ARID and its expiry time in the same data format used in the received HTTP request. The HTTP messages MUST be sent over TLS to protect message confidentiality and integrity. In the following example, the ARID is attached as a JSON object. The "arid" field consists of the URL of the website for the ARID validation, "https://attributes.example.org/", and the ARID, "17750c5cbac9979171991d505d2e634e727d8d9b."

F2. 200 OK sent from AVS to Alice:

```
HTTP/1.1 200 OK
Content-Type:application/json
```

```
{ "arid":"https://attributes.example.org/17750c5cbac997917199\
1d505d2e634e727d8d9b", "expires":"2011-08-24T16:20:20Z" }
```

5.3. Sending an ARID in a Communication Request

When Alice makes a call to Bob with an ARID, she needs to specify the ARID associated with the URL of the website for validating the ARID in a SIP UA. The SIP UA MUST generate a new SIP header called "Sender-Reference" including a URI, "type", "salt", and "hash_alg" parameters to convey the ARID in the path of an HTTP URL, specify this service, and the salt and the hash algorithm which were used for hashing the querier's contact address described in Section 5.2, respectively. If Alice wants to specify multiple ARIDs, this Sender-References header field includes multiple set of an ARID and related parameters concatenating a comma separator. The SIP UA then sends an SIP INVITE request including the Sender-Reference as shown in the following example. The INVITE request MUST be sent over TLS to protect message confidentiality and integrity.

Instead of defining a new SIP header field, the existing Call-Info header field can be set to an ARID by defining a new value of the purpose parameter, such as "sender-attributes." However, to convey a salt and the hash algorithm, we also need to define two more parameters. To avoid complexing the Call-Info parameter structure, we rather define a new SIP header field.

F3. SIP INVITE from Alice to Bob:

```
INVITE sips:bob@example.com SIP/2.0
From:Alice <tel:+12345678>
To:Bob <sips:bob@example.com>
Sender-References:<https://attributes.example.org/\
17750c5cbac9979171991d505d2e634e727d8d9b>;type="avs";\
salt="dmvblp03";hash_alg="SHA1"
```

5.4. Validating an ARID to Retrieve User Attributes

When Bob, the recipient of one or more ARIDs, wants to retrieve the caller attributes, the SIP UAS needs to test the validity of the ARIDs on the corresponding AVSes. By prompting Bob or based on his preconfigured information, the SIP UAS first needs to determine whether or not he trusts each domain name of the AVS in the Sender-References header in received SIP INVITE request. Only for trusted AVSes, the SIP UAS looks up the received ARIDs on the corresponding AVSes to retrieve the caller's attributes by using an HTTP GET request as shown in the following example. HTTP messages MUST sent

over TLS for security as well as messages between the SIP UAC the AVS. Bob authenticates the AVS using its X.509 PKC delivered in the TLS handshake.

For this validation, the SIP UAS MUST send the ARID found in the Sender-References header field and a hashed querier's contact address generated by the hash algorithm and salt also found in the Sender-References header field. To generate a hashed querier's contact address, the SIP UAS needs to know the original destination address by extracting from the To header or by Bob's pre-configuration especially when he enables call forwarding services. In the following example, the hashed querier's contact address, "2cf6aleda3b5205005d25a7d5dcf13bb200fc26a", is generated by SHA1("dmvb1p03"||"sips:bob@example.com"). This validation MAY be invoked by a SIP inbound proxy on behalf of the UAS.

F4. HTTP GET from Bob to AVS:

```
GET /17750c5cbac9979171991d505d2e634e727d8d9b/\
2cf6aleda3b5205005d25a7d5dcf13bb200fc26a HTTP/1.1
HOST:attributes.example.org
```

If no AVSes are trusted by Bob, the SIP UAS MUST ignore the Sender-Reference header field and stop any further validation process. If the SIP UAS does not support a hash algorithm specified in the Sender-References header field, or if the SIP UAS does not support the header field, it SHOULD also ignore the header field and continue normal processing of the received SIP request.

If the ARID is valid at the queried time and with the querier's contact address, the AVS MUST respond to the querier with 200 OK in HTTP having the attributes based on the disclosure mode which Alice specifies in the message body, as shown in the following example. The attributes SHOULD be attached as a JSON object or MAY be in XML. If the query is done later than the expiry time, the AVS SHOULD respond with 408 Request Timeout in HTTP. If the querier is not included in the list of desired queriers specified earlier by Alice, the AVS SHOULD respond with 403 Forbidden in HTTP. If the ARID is invalid for other reasons, the AVS MUST respond with 404 Not Found in HTTP.

F5. HTTP 200 OK from AVS to Bob:

```
HTTP/1.1 200 OK
Content-Type:application/json

{ "user_status":"student member" }
```

If Bob receives a 200 OK in HTTP from the AVS, he is informed that the ARID is valid and attached information is the caller's attributes, for the example above, the caller is a student member in "example.org". With any other responses, Bob knows nothing about the caller's attributes. Based on this information, he determines whether or not to answer the call and adjusts his communication stance accordingly.

6. Sender-References Header Field

The SIP "Sender-References" header field is newly defined to provide the reference information about the sender or the caller. The field consists of one or more sender-ref information. Each sender-ref information consists of three parts: an absolute URI, sender-ref-type, and avs-params. The absolute URI contains the URI of the AVS website including an ARID in the path. The sender-ref-type indicates the service type of using this header field. For this referencing service, it MUST be "avs." The avs-params consists of two parameters: one is to specify a salt and another is for a hash algorithm. Both parameters are used for hashing a contact address to be presented for validation.

The syntax of the Sender-References header field in the ABNF [RFC5234] is as follows:

```
Sender-References = "Sender-References" HCOLON sender-ref
                  *(COMMA sender-ref)
sender-ref         = LAQUOT absoluteURI RAQUOT sender-ref-type
                  SEMI avs-params
sender-ref-type    = "type" EQUAL ( "avs" / quoted-string )
avs-params         = salt-param SEMI hash-alg-param
salt-param        = "salt" EQUAL quoted-string
hash-alg-param    = "hash-alg" EQUAL ( "SHA1" / quoted-string )
```

This Sender-References header field is optionally set in any SIP requests and responses.

7. Relationship to Existing Mechanisms

This section discusses why this referencing mechanism does not use existing mechanisms that provide an attribute assertion or third-party authentication, such as an X.509 Attribute Certificate (AC), a Security Assertion Markup Language (SAML) assertion, Vouch by Reference [RFC5518], OAuth [RFC5849] or Kerberos [RFC4120]. The following table compares our mechanism using an AVS with these existing mechanisms in terms of the trust model they assume, whether

or not to need to bind the assertion to the sender ID, and applicable services.

	AVS	X.509 AC	SAML assertion	VBR	OAuth	Ker- beros
trust model	described in Sec 3.1	the same as AVS			different from AVS	
need for binding to sender ID	no	yes	optional	yes, with the sender domain	no	yes
apps.	SIP	any	any	email	Web	any

apps. = applications

An X.509 Attribute Certificate (AC) provides a superset of features we need for our equivalent trust model. However, an X.509 AC, unlike our mechanism, requires the AC holder information, namely a user's identity, to be bound to the user's attributes. This binding is protected by being digitally signed with the AC issuer's private key. However, the AC issuer does not always have the right to sign the binding since the AC issuer cannot authenticate the user identity issued by a different organization as described in Section 3.2. Authenticating the user identity requires either the user's PKC or other mechanisms, such as the SIP identity mechanism where the user identity is a user identifier in SIP. These mechanisms are difficult to deploy for each reason. Users' PKCs have not been widely deployed because of difficulty in managing the pair of public and private keys across multiple devices. The sender ID in SIP is usually issued by an administrative domain different from the AC issuer. For these reasons, we need a new mechanism to allow a looser linkage between the sender ID and attributes.

Similar to an X.509 AC, a SAML assertion provides a superset of features we need for our equivalent trust model. Unlike an X.509 AC, a SAML assertion does not require the binding between user attributes and the user identity. Including the user identity into a SAML assertion is optional. To limit queriers, a SAML assertion can restrict its audience by addressing the URIs of specific entities, but they are currently not allowed to address by their hashed names. Thus, with a minor modification in the form of the restricted audience, we can use an XML-based SAML assertion to convey user attributes instead of a JSON object described in Section 5.4. However, using a SAML assertion requires a digital signature by its

issuer, which is an application layer protection against message tampering and server impersonation. As discussed in Section 4.1, we prefer a simple transport layer protection to an application layer one, namely protecting a whole message by TLS rather than protecting part of a message by a digital signature.

Vouch by Reference (VBR) defines a simple mechanism that vouches a specific type of content claimed by the sender's domain of an email message. This mechanism uses a new email "VBR-Info" header and a DNS-based server of a third party certification service. If the recipient finds a trusted domain from the certification service providers set in the VBR-info header in a received email message, he looks up an entry of the sender domain on the DNS-based server of a trusted certification service provider. Since VBR assumes the same trust model as ours, it is possible to extend this VBR mechanism to vouch a user's attributes instead of certifying a specific content type for the sender's domain. However, VBR requires the authentication of the sender domain since the server domain is used as a query key. Additionally, it is difficult for a DNS-based server to restrict queriers for each record, mainly private attributes. Consequently, we cannot apply VBR to referencing user attributes.

OAuth is a third-party authentication model for Web services. OAuth uses three tokens to delegate limited permissions of user's resource to another entity called a Consumer. With the OAuth terminology, a caller is a User, the callee is a Consumer, and the AVS is a Service Provider, which is a third-party authenticator. In OAuth, unlike our trust model, the AVS and the callee share a Consumer ID and a key to authenticate the Consumer when the AVS provides one of these three tokens, a Request Token. An unauthorized Request Token is generated upon the Consumer's request. After the Consumer is authorized by a User, it is provided with an authorized Request Token. Since this authorized Request Token has a restricted scope and limited lifetime to access the Users' resources, this Request Token can be used to query the caller's attributes once the caller authorizes that. However, to obtain an authorized Request Token, the callee, who is a Consumer in OAuth, needs to obtain an unauthorized Request Token from the AVS beforehand. To resolve these differences in the trust models and the procedures, it is possible to omit using both an unauthorized and authorized Request Tokens. In addition, using the third token in OAuth, an Access Token, which is exchanged with an authorized Request Token, is not needed. Thus, we do not need unnecessary complexities of using these three types of tokens. Rather, we prefer a simple mechanism, using a single token for SIP.

Kerberos provides strong user-client authentication using a Key Distribution Center (KDC). An Authentication Server in KDC authenticates a user on behalf of a Service Server (SS), and a Ticket

Granting Server (TGS) issues a ticket which is effective only for a session between the user and the SS for a limited time period. The Kerberos features cover all our mechanism needs, but the trust model is different. Kerberos assumes that the TGS and the SS share the SS's secret key to allow the SS to verify a received ticket by decrypting with the SS's key without connecting to the TGS. Since using this ticket is a key feature in Kerberos, we cannot omit sharing the SS's key with the TGS to resolve the difference in the trust model. Because of this difference in assumed trust model, we cannot use Kerberos for referencing and validating user attributes.

8. Security Considerations

8.1. Man in the Middle Attacks

Man in the middle attacks need to be prevented on the AVS, connection links, and the recipients of an ARID. To prevent from impersonating a user on the AVS, the AVS MUST authenticate a user using HTTP Basic or Digest authentication, a client X.509 PKC or other mechanisms. To prevent from eavesdropping and message tampering on connection links, all connection links between UAC and the AVS, UAC and UAS, UAS and the AVS MUST be protected using TLS.

To prevent from impersonating user attributes using a stolen ARID, the AVS MUST limit queriers using a specific ARID based on the hashed contact addresses the original requester of the ARID specifies. SIP UAs MUST support SHA1 to hash a contact address. To mitigate the damage of the impersonation in the case where an ARID is stolen with one of these hashed contact addresses, the AVS MUST limit an ARID's lifetime and MAY also limit the number of times it can be resolved.

Limiting the use times of an ARID strengthens security, but reduces service applicability. When the originator of a communication knows that the communication has multiple recipients, she can specify these multiple destination addresses as designated queriers. The AVS then limits the use times of an ARID to one for each designated querier. However, the difficult case exists where the originator cannot know the number of honest recipients or their addresses, for example, using a forking proxy at the destination side or using a list service that distributes the same message to multiple registered destinations.

8.2. Replay Attacks Using a Received ARID

The recipient of an ARID can exploit impersonation just by forwarding a received ARID to another user since this mechanism does not have a tight link between the username of AVS and the caller ID as described in Section 3.2 nor a link between the SIP signaling path and the

ARID. To prevent this replay or forwarding attack, the AVS MUST limit queriers for each ARID based on the hashed contact addresses that the original requester of the ARID specifies. This is the same way as preventing impersonation using a stolen ARID described in Section 8.1.

Suppose Bob, the recipient of Alice's ARID, forwards the ARID to Carol when sending an instant message to her. In the message, Bob instructs Carol to query his attributes using his hashed contact address, instead of hers. By instructing this wrong way of query, Bob fails in his attempt to masquerade as a user having Alice's attributes. However, despite Alice's original designation, Carol can retrieve Alice's attributes following Bob's wrong instruction, resulting in raising Alice's privacy concerns. This privacy problem is caused by Bob's misbehavior and unavoidable for any attribute mechanisms which others can retrieve the attributes.

8.3. Denial of Service Attacks on the AVS

Another form of potential attacks is denial of service (DoS) attacks by flooding requests to exhaust resources on the AVS. To mitigate the damage from DoS attacks, we need to spare resources for valid requests. For this purpose, the AVS MUST carefully configure TCP and implement user authentication. To detect invalid requests as easily as possible, this mechanism SHOULD use a light query protocol using the RESTful API [REST], which sets a query key in the path of an HTTP URL.

8.4. Phishing Attacks on the AVS

An evil website having a domain name confusingly similar to a well-known AVS makes it possible to steal the password of a user for remote access to the AVS. It is also possible for an evil website to respond to any attribute queries with an HTTP 200 OK response with forged user attributes attached to invalidate the attribute validation service. To prevent these attacks, both a user and the recipient of an ARID MUST use TLS when connecting to the AVS and MUST ensure that the server's PKC has a valid signature for the valid domain name.

9. IANA Considerations

This document defines a new SIP Sender-References header field. This header field needs to be registered by the IANA in the SIP Parameters registry under the Header Fields sub-registry.

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.
- [RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.
- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.
- [RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.
- [SHA1] National Institute of Science and Technology, "Secure Hash Standard", Federal Information Processing Standard (FIPS) 180-2, August 2002.

10.2. Informative References

- [AES] National Institute of Science and Technology, "Specifications for the Advanced Encryption Standard", Federal Information Processing Standard (FIPS) 197, November 2001.
- [REST] Fielding, R. and R. Taylor, "Principled design of the modern Web architecture", ACM Transactions on Internet Technology (TOIT) 2-2, May 2002, <<http://portal.acm.org/citation.cfm?doid=514183.514185>>.
- [RFC2617] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999.
- [RFC3966] Schulzrinne, H., "The tel URI for Telephone Numbers",

RFC 3966, December 2004.

- [RFC4120] Neuman, C., Yu, T., Hartman, S., and K. Raeburn, "The Kerberos Network Authentication Service (V5)", RFC 4120, July 2005.
- [RFC4474] Peterson, J. and C. Jennings, "Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)", RFC 4474, August 2006.
- [RFC4484] Peterson, J., Polk, J., Sicker, D., and H. Tschofenig, "Trait-Based Authorization Requirements for the Session Initiation Protocol (SIP)", RFC 4484, August 2006.
- [RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, July 2006.
- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.
- [RFC5518] Hoffman, P., Levine, J., and A. Hathcock, "Vouch By Reference", RFC 5518, April 2009.
- [RFC5751] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification", RFC 5751, January 2010.
- [RFC5755] Farrell, S., Housley, R., and S. Turner, "An Internet Attribute Certificate Profile for Authorization", RFC 5755, January 2010.
- [RFC5849] Hammer-Lahav, E., "The OAuth 1.0 Protocol", RFC 5849, April 2010.
- [SAML] "Security Assertion Markup Language (SAML) V2.0", March 2005,
<<http://docs.oasis-open.org/security/saml/v2.0/>>.
- [XML] Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E., and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Third Edition)", W3C Recommendation REC-xml-20040204, February 2004.

Authors' Addresses

Kumiko Ono
Department of Computer Science
Columbia University
New York, NY 10027
USA

Email: kumiko@cs.columbia.edu

Henning Schulzrinne
Department of Computer Science
Columbia University
New York, NY 10027
USA

Email: hgs@cs.columbia.edu

