Private Extensions to the Session Initiation Protocol (SIP) for Asserter Identification within Trusted Networks
draft-kaplan-sip-asserter-identity-01

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/1id-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 September 8, 2008.

Copyright Notice

Copyright (c) 2009 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 in effect on the date of publication of this document (http://trustee.ietf.org/license-info). Please review these documents carefully, as they describe your rights and restrictions with respect to this document.
Abstract

This document describes private extensions to the Session Initiation Protocol (SIP) that enable a network of trusted SIP servers to identify the assessor of private user identity defined in RFC 3325. The use of these extensions is only applicable inside a set of administrative domains with previously agreed-upon policies for generation, transport and usage of such information. This document does NOT offer a general identity model suitable for use between different trust domains, or use in the Internet at large.

Table of Contents

1. Introduction...........................................2
1.1. The Problem...........................................3
1.2. The Solution...........................................3
1.3. Uses for P-Asserter....................................4
2. Terminology.............................................4
3. Applicability............................................4
4. Definitions..............................................5
5. Overview of Operation....................................5
6. PASS Generator Behavior...............................6
7. PASS Verifier Behavior..................................8
8. Proxy Behavior..........................................10
9. Header Syntax..........................................10
10. Handling Errors and Failures..........................13
11. Example................................................14
12. Security Considerations..............................15
13. IANA Considerations....................................16
14. Acknowledgements......................................16
15. References.............................................16
15.1. Normative References...............................16
15.2. Informative References.............................16
Author’s Address...........................................17

1. Introduction

Many SIP service providers use the P-Asserted-Identity (termed "PAI" in this document) mechanism defined in [RFC3325], to identify the calling party. PAI was not meant to cross Trust-domain boundaries, where the definition of "Trust-domain" is defined such that all parties must adhere to the same behavior assumptions. When two administrative domains interconnect, in order for each to be able to use the PAI header value generated by the other, they must agree and conform to the same Trust-domain policies, in order to be in the same "Trust-domain". Such agreement and trust may be possible for two directly attached
administrative domains, but is much more difficult to achieve in large-scale multi-domain arrangements.

1.1. The Problem

The issues identified in [identity-important] expose the weaknesses in such a multi-domain PAI model. As more administrative domains are joined to the Trust-domain, the PAI mechanism becomes brittle.

One concern is that entry points into the Trust-domain may be improperly provisioned or malfunctioning, such that the assertions they make in PAI header values may not be based on strong authentication of the originating user identity. It only takes one open relay, or poorly provisioned entry point to the Trust-domain, to corrupt the mechanism. Historically, email deployments experienced such issues from "open relays", which were exploited by spam senders to forward their messages through trusted, legitimate email domains.

Unfortunately, the PAI header field does not identify the asserter of the PAI value. Without such information, it is difficult to track down the source of invalid assertions. The assertions will most likely only be identified as invalid at the final terminating target domain, for example if users report incorrect or malicious caller-ID indications. Since the PAI value can be generated by any node or domain along the signaling path, efforts to resolve abuse or malfunction are very difficult in current SIP deployments.

Clearly, one solution would be to have the asserter of PAI value(s) identity itself or its domain by inserting its identity in SIP requests or responses. However, if such an "asserter identifier" is unauthenticated, anyone could insert it claiming to be any domain. The mechanism would thus suffer the same issues as PAI.

1.2. The Solution

This draft proposes a solution which addresses the problem in a secure manner: asserter identity. The basic concept is the identity of the asserting node or domain which generates the PAI header value(s) is inserted into a new 'P-Asserter' header field in the same SIP message which it inserts a PAI header field into. This allows users to both identify the root of erroneous PAI assertion quickly, and to form a basis for a reputation system of which asserters and assertions to believe or not-believe until such time as the asserter can be fixed.
Because asserter identities can easily be faked, impacting the reputation of the legitimate asserting domain, this draft also proposes a cryptographic means by which to assert and verify the assertion. Readers familiar with SIP Identity [RFC4474] will note a distinct similarity with that mechanism. Indeed, the general concept of [RFC4474] is used for this draft’s mechanism, with subtle but important differences. The two are not at odds, however: both can be used at the same time, or not, but this draft is specifically aimed at solving a different problem — one of strong, identifiable P-Asserted-Identity assertion.

This document does NOT offer a general identity model suitable for inter-domain use outside of the Trust-domain, or use in the Internet at large. Its assumptions about the trust relationship between the user and the network, and between networks, may not apply in many applications. For example, these extensions do not accommodate a model whereby end users can independently assert their PAI value by use of the extensions defined here.

1.3. Uses for P-Asserter

It should be noted that the P-Asserter mechanism proposed in this document does not guarantee the PAI value is correct. It only provides an assurance that the asserter identified in the P-Asserter field generated the PAI value, with the fields covered by the PASS signature.

The PASS mechanism does, though, lead to some obvious inferences that can be made based on the PAI and asserter identities. For example, if the P-Asserter URI domain matches both the PAI URI domain and From domain, an inference regarding user-identification similar to [RFC4474] and [RFC4916] can be made. Some nodes may even decide that if the From URI is a telephone number, and the signed PAI has the same number, that the caller-id may be inferable. This document makes no claims or statements regarding what may or may not be inferred from such things at this time.

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 RFC 2119. The terminology in this document conforms to RFC 2828, "Internet Security Glossary".

3. Applicability

This draft proposes enhancements relying on the P-Asserted-Identity mechanism defined in [RFC3325].
4. Definitions

PAI: the P-Asserted-Identity header field, defined in [RFC3325].

PASS header: the "P-Asserter" header field defined in this document.

PASS mechanism: the mechanism proposed in this document.

5. Overview of Operation

The mechanism proposed in this document works very similarly to [RFC4474], except it signs the P-Asserted-Identity header field based on [RFC3325], and a new ‘P-Original-To’ header value, instead of the From and To header field URI values. It also does not sign the Call-Id or CSeq header values, and instead creates a specific new "seq" sequence parameter for that role. Lastly, it does not sign the Contact header value nor the entire message body. The Date header value is still used and signed.

The mechanism relies on three new header fields named ‘P-Asserter’, ‘P-Asserter-Info’, and ‘P-Original-To’. The ‘P-Asserter’ header field value contains a URI (commonly a SIP URI), and a sequence number parameter; for example:

P-Asserter: <sip:node12-if6@example.com>;seq=12834

A proxy server which inserts a P-Asserted-Identity header field into a SIP request or response and forwards it to other trusted nodes as defined in [RFC3325] and [update-pai], can insert this new P-Asserter header field identifying itself or its domain, along with a unique sequence number, and sign it, as defined later.

The sequence parameter contains a number which will be unique for all requests or responses from the PASS URI, for a time window, as defined later. The Verifier uses this to detect replay attacks, similar to the combination of Call-Id and CSeq in [RFC4474].

The asserter also copies the far-end target of the request or response, typically the To URI for requests and From URI for responses, into a new ‘P-Original-To’ header field. This is also included in the signature calculation, and is used by the verifier to validate the target and prevent replay and [draft-baiting] type attacks. The ‘To’ header field is not directly used for the signature, because it is often changed along the path to the target in real-world deployments today.
Although the SIP message bodies in the signed message may be entirely included in the signature calculation, the PASS mechanism allows the signer to choose not to do so, and optionally to sign specific portions of the body. This is accomplished by additional parameters in the P-Assertor-Info header field, as defined later. For example, the assessor may decide to only sign Fingerprint attributes for [DTLS-SRTP] verification, and leave the rest of an SDP body unsigned so that intermediaries can change it. Or the Asserter may decide to sign MESSAGE method bodies, or PUBLISH bodies, or whatever.

The Verifier performs validation of the message in a similar fashion as [RFC4474], by validating the signature for the message given the values received. Instead of using the received Call-Id and Cseq pair to check a local cache for a unique signature, a PASS Verifier uses the ‘seq’ sequence number parameter value for such a cache instead.

Note that this draft discusses signing and validation of SIP messages, and thus responses as well as requests.

6. PASS Generator Behavior

This document defines a mechanism by which the assessor of PAI based on [RFC3325] identifies itself or its domain; this role is called a PASS Generator. Any entity which performs the role of PASS Generator MUST possess the private key of a domain certificate, matching the domain name it will use in the P-Assertor URI.

The PASS mechanism relies on the node or domain asserting the PAI header value(s) to perform some form of authentication for the PAI identity it asserts. For example, it may digest-challenge a request before asserting PAI, or rely on TLS for response authentication. Even if the PAI assertion is weak, however, the PASS mechanism is useful in determining the source of weak assertions, for example for reputation systems or fraud resolution, or simply to determine malfunctioning asserters.

The role of the PASS Generator is to perform the assertion of PAI and identify itself in the PASS header field, and sign the message. The specific steps it MUST perform are:

Step 1:

The PASS Generator MUST insert a PAI header value(s), based on the identity of the request or response originator, as defined in [RFC3325] or [update-pai], or other relevant specifications. In order to generate PASS information, the PAI assertion MUST be
based on some knowledge as to the authenticity of the identity, such that the message sender can claim to represent the identity. Such knowledge can be achieved, for example, by digest authentication, TLS certificate verification, policies defining allowable identities, etc.

If the PASS Generator cannot verify the originating identity claim, it MAY still generate a PAI, but MUST NOT generate or insert PASS header fields. Doing otherwise is not in the best interest of the PASS Generator, as it would be signing an assertion for an identity it does not know to be accurate, and thus may lead to impacting the reputation of its assertions.

Step 2:

The PASS Generator MUST ensure that any preexisting Date header value in the message is accurate, to within ten minutes. If the Date header value contains a time outside of the ten minute window, it MUST either replace the value with an accurate one, or it MAY reject the message if it is a request. If no Date header is present in the message, the PASS Generator MUST insert one. The Date header value is included in the signature calculation, and is used by the PASS Verifier to detect stale signatures and prevent replay/cut-paste attacks.

Step 3:

The PASS Generator MUST insert a ‘P-Original-To’ header field, using the URI value of the ultimate target of the request or response. For SIP requests, this would typically be the value of the received To header URI value, unless local policy dictates otherwise. One rationale for using a different value would be if the request’s To-URI value needed to be modified due to number translation rules, or converted from a SIP URI to a TEL URI, for example. For SIP responses, the P-Original-To URI value would typically be the URI of the From header field. Details on the generation of this header are provided in Section 10.
Step 4:

The PASS Generator MUST insert a ‘P-Asserter’ header field value identifying itself or its domain in the form of a URI, and include a unique sequence number in the ‘seq’ header parameter. Details on the generation of both of these headers are provided in Section 10.

Step 5:

The PASS Generator MUST form the PASS signature using its appropriate private key of its certificate, and insert a ‘P-Asserter-Info’ header field with this value, along with other mandatory PASS-Info values as defined later.

Finally, the PASS Generator MUST forward the message normally.

7. PASS Verifier Behavior

This document introduces a new logical role for SIP entities called a PASS Verifier. When a Verifier receives a SIP message containing a PAI and PASS header fields, it may inspect the signature in the PASS-Info header field to verify the identity of the PAI asserter. Typically, the results of the verification are provided as input to an authorization process that is outside the scope of this document.

If P-Asserter and P-Asserter-Info header fields are not present in a request, and one is required by local policy, then a 400 ‘Use PASS Signature’ response MUST be sent, with a Reason header pass-cause code of "1", as defined later in this document. If PASS and PASS-Info header fields are not present in a response, and one is required by local policy, then any associated state created by the original request MUST be removed (e.g., by sending a BYE or CANCEL), depending on the state of the dialog. The tear-down request MUST include the Reason header pass-cause code of "1", as defined later as well.

In order to verify the identity of the sender of a message, an entity acting as a verifier MUST perform the following steps, in the order here specified.

Step 1:

The verifier MUST acquire the certificate for the signing domain, following the same behavior defined in [RFC4474] for SIP Identity Verifiers, except using the P-Asserter-Info header field values. Certificates cached for PASS verification should be indexed by the
certificate location URI given in the P-Asserter-Info header field value.

If the certificate location URI scheme in the P-Asserter-Info header of a SIP request cannot be dereferenced, then a 400 'Bad PASS-Info' response MUST be returned, with a Reason header field pass-cause code of "2", as defined later. If a SIP response cannot be so dereferenced, then the Verifier MUST either remove the PAI, PASS and PASS-Info header fields and forward the response, or it can CANCEL the original request. If the request is canceled, the CANCEL request MUST contain a Reason header with the pass-cause code of "2", as defined later.

Step 2:

The verifier MUST verify the signature in the P-Asserter-Info header field, following the procedures for generating the hashed digest-string described in Section 10.

If a verifier determines that the signature on the message does not correspond to the reconstructed digest-string, then for requests it MUST generate a 400 'Invalid PASS Signature' response, with a Reason header pass-code parameter value of '3'. For responses it MAY choose to terminate any dialog created by the response (e.g., through a BYE or CANCEL), but MUST add a Reason header field with a pass-code parameter value of '3' if doing so; or it may forward the response upstream but then MUST remove the PAI, PASS, and PASS-Info headers.

Step 3:

The verifier MUST validate the Date header in the manner described in Section 10; recipients that wish to verify PASS signatures MUST support all of the operations described there. It must furthermore ensure that the value of the Date header falls within the validity period of the certificate whose corresponding private key was used to sign the PASS header.

Step 4:

The verifier MUST validate that the URI in the ‘P-Original-To’ header field either identifies the resource it will forward the message to, or that the resource it will forward the message to is willing to accept messages addressed for the URI. How he verifier determines the P-Original-To URI is of such a type is beyond the scope of this document. One example would be using a list of URI’s the forwarded-to user has populated on the verifier through a service/account portal.
8. Proxy Behavior

A proxy that is about to forward a message to a next-hop that it does not trust MUST remove the PASS, P-Asserter-Info, and P-Original-To header fields if the PAI header is removed - for example if the user requested privacy per [RFC3325]. Note that a legacy [RFC3325] proxy only removing the PAI header without removing the PASS headers will still keep the sensitive information private, but also invalidate the signature. Therefore there is no security concern with a failure to remove the information in such a case, but not doing so could trigger validation check failures.

9. Header Syntax

This document specifies two new SIP headers: P-Asserter and P-Asserter-Info. Each of these headers can appear only once in a SIP message. The grammar for these two headers is (following the ABNF [6] in [RFC3261]):

```
PASS       = "P-Asserter" HCOLON Pass-value
Pass-value = Pass-uri SEMI seq-param *( SEMI generic-param)
Pass-uri   = name-addr / addr-spec
seq-param  = "seq" EQUAL 1*DIGIT

PASS-Info = "P-Asserter-Info" HCOLON cert-loc-info
            *( SEMI pass-info-params )
cert-loc-info = LAQUOT absoluteURI RAQUOT
pass-info-params = signed-pass-digest / signed-pass-bodies
                   / pass-info-alg / generic-param
signed-pass-digest = "sig" EQUAL DQUOT 32LHEX DQUOT
pass-info-alg = "alg" EQUAL token
signed-pass-bodies = "bodies" EQUAL DQUOT body-infos DQUOT
body-infos = body-info *(SEMI body-info)
body-info = full-body-type / sdp-att / body-extension
full-body-type = "full" COLON content-type-name
content-type-name = m-type SLASH m-subtype ; from RFC 3261
sdp-att = "sdp-att" COLON att-field ; from RFC 4566
body-extension = token COLON token
```

The P-Asserter header field pass-uri MUST consist of exactly one name-addr or addr-spec, which MUST identify the logical "node" or domain that inserted the PAI header value(s). Note that only its host portion is relevant - it MUST be a domain name, and MUST match the domain identified by the certificate used for the signature. It may be a Fully Qualified Domain Name (FQDN) identifying the host machine, or a site domain name; in either case, it MUST be a domain name.
case, the URI domain name MUST match the domain name of the Certificate used for the signature. The username portion of the URI, the display-name, and any URI parameters, if present, are essentially for troubleshooting purposes and have no meaning to anyone other than the generator of them.

The sequence number MUST be guaranteed to be unique for each message, within the scope of the P-Asserter URI domain or FQDN name for at least 3600 seconds after the Date header field value time. Message retransmission at the SIP transport layer MUST NOT generate a new sequence number, and thus the same signature value will be used for retransmissions. In other words, PASS authentication and validation occur at a layer above the SIP transport layer.

The P-Asserter-Info header field value contains a URI in the cert-loc-uri field from which its certificate can be acquired, following the same rules and requirements as defined for the Identity-Info header value in [RFC4474]. The P-Asserter-Info header value is not itself contained in the signature calculation, because any modification of it by malicious parties would invalidate the signature anyway.

The P-Asserter-Info header field also indicates which message bodies, if any, were included in the signature process, and which SDP attributes, if any. For every body which is fully signed, its content-type name is added as a "full" type per the ABNF defined. If the SDP is not fully signed, each SDP attribute is identified as an "sdp-att" type in the syntax. See Section 14 for an example of this use.

The ‘P-Asserter-Info’ header field signed-pass-digest value contains a base-64 encoded signature of the hash of certain portions of the message. To create the contents of the signed-identity-digest, the following elements of a SIP message MUST be placed in a bit-exact string in the order specified here, separated by a vertical line, "\n" or %x7C, character:

1. The entire value, including any parameters, of the P-Asserted-Identity header field(s). If more than one value is present, the values are used in order as they appear in the message, separated by a comma. LAQUOT and RAQUOT separators MUST be added to the URI for the purpose of signature calculation, even if they are not in the received message.
2. The entire value, including any parameters, of the P-Original-To header field. LAQUOT and RAQUOT separators MUST be added to the URI for the purpose of signature calculation, even if they are not in the message.
3. The entire value, including any parameters, of the P-Asserter header field. LAQUOT and RAQUOT separators MUST be added to the URI for the purpose of signature calculation, even if they are not in the message.

4. The Date header field, with exactly one space each for each SP and the weekday and month items case set as shown in BNF in RFC 3261. RFC 3261 specifies that the BNF for weekday and month is a choice amongst a set of tokens. The RFC 2234 rules for the BNF specify that tokens are case sensitive. However, when used to construct the canonical string defined here, the first letter of each week and month MUST be capitalized, and the remaining two letters must be lowercase. This matches the capitalization provided in the definition of each token. All messages that use the PASS mechanism MUST contain a Date header.

5. Any and all body content types identified as "full" in the P-Asserter-Info field, the body content of the message with the bits exactly as they are in the Message (in the ABNF for SIP, the message-body). This includes the named components of multipart message bodies. Note that the message-body does NOT include the CRLF separating the SIP headers from the message-body, but does include everything that follows that CRLF. For multipart bodies, if multiple "full" content types are identified, they are included in the order they appear in the P-Asserter-Info header field.

6. Any and all SDP attributes identified by the "sdp-att" syntax of the P-Asserter-Info header field. For each such identified attribute name, the first matching attribute field name in the SDP is used, with the att-value bits exactly as they are in the SDP, not including the "a=" nor the att-field name itself. Multiple "sdp-att" identified names of the same name identify multiple matching attribute field values. For example, two "sdp-att:fingerprint" in the P-Asserter-Info header field mean the first two "a=fingerprint:<value>" lines found in SDP are used (namely, the <value> portions are used). Multiple SDP attributes are separated by a comma when concatenating for this process.

If the message has no body, then the last two portions will be empty, and the final two "|" will not be followed by any additional characters.

```
Digest-string = PAssertedID-value "|" POriginalTo-value "|"
               PASS-value "|" SIP-date "|" [message-body] "|"
               [att-value*,|att-value)]
```

After the digest-string is formed, it MUST be hashed and signed with the certificate for the domain, in identical fashion as [RFC4474]. The hashing and signing algorithm is specified by the
'alg' parameter of the P-Asserter-Info field, following the same parameter behavior as in [RFC4474].

10. Handling Errors and Failures

SIP Identity in [RFC4474] added specific response codes for handling failures of the mechanism, which the PASS mechanism does not. The same responses cannot be reused, because they are specific to the mechanism in [RFC4474] and only applicable to request failures. Since devices which do not understand this new mechanism will treat any unknown 4xx response as a 400, the PASS mechanism defines using exactly that, and putting the specific failure cause in a Reason header based on [RFC3326]. The Reason header was only specified for use in SIP requests in [RFC3326], however this author cannot find any normative statement they cannot be used in responses as well.

Failures of the PASS mechanism which generate 400 responses as defined in this document, or cause any associated state to be torn down (e.g., with a BYE or CANCEL), MUST include a Reason header with the protocol value of "SIP", and an appropriate pass-cause code value.

The ABNF for this, as defined in [RFC3326], is as follows:

```plaintext
reason-extension = pass-cause / generic-param
pass-cause       = "pass-cause" EQUAL cause
```

The reason-text parameter MAY also be included, but it has no defined value or meaning and is useful for troubleshooting purposes only - it MUST be treated as an opaque string.

The following pass-cause code values are defined:

- "0" = unknown
- "1" = PASS mechanism required; for example the P-Asserter header was not present
- "2" = Bad PASS-info; for example the P-Asserter-Info certificate could not be dereferenced, or he P-Asserter-Info parameters could not be understood
- "3" = Invalid PASS Signature; the received signature hash did not match the one locally generated
11. Example

This example shows how two a=fingerprint lines in SDP would populate the P-Asserted-Info SIP header field. The following is an example of an INVITE created by the endpoint.

(lines folded for readability)

INVITE sip:bob@biloxi.example.org SIP/2.0
Via: SIP/2.0/TLS pc33.example.com;branch=z9hG4bKnashds8
Via: SIP/2.0/TLS hal9k.example.com;branch=z9hG4bKnorm
To: Bob <sip:bob@biloxi.example.org>
From: Alice <sip:alice@example.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Max-Forwards: 70
Date: Thu, 21 Feb 2002 13:02:03 GMT
Contact: <sip:alice@pc33.atlanta.example.com>
P-Asserted-Identity: Alice <sip:alice@example.com>
P-Asserted-Identity: tel:+17815551212
P-Original-To: <sip:bob@biloxi.example.org>
P-Asserter: <sip:daisy@hal9k.example.com>;seq=2001
P-Assertor-Info: <https://hal9k.example.com/hal9k.cer>
  ;alg=rsa-sha1
  ;bodies="sdp-att:fingerprint;sdp-att:fingerprint"
  ;sig="ZYNBbHC00VMZr2kZt6VmCvPonWJMGvQTBDbghoWeLxJfzB2a"
Content-Type: application/sdp
Content-Length: xxx

v=0
o=- 6418913922105372816 2105372818 IN IP4 192.0.2.1
s=example2
c=IN IP4 192.0.2.1
m=audio 54113 UDP/TLS/RTP/SAVP 0
a=fingerprint:SHA-1
m=video 54115 UDP/TLS/RTP/SAVP 0
a=fingerprint:SHA-1

In the above example, the signature shown is not an accurate value, but simply shown for descriptive purpose.
The digest-string, based on the above example, would be the following inside the quotes, as one long line:

"Alice <sip:alice@example.com>,<tel:+17815551212>|<sip:bob@biloxi.example.org>|<sip:daisy@hal9k.example.com>
;seq=2001|Thu, 21 Feb 2002 13:02:03 GMT||SHA-1

Note the empty full message-body portion, shown by two "|" side-by-side.

12. Security Considerations

The considerations in [RFC4474] apply to this document’s proposed mechanism as well, and will not be repeated here. There are several additional security consideration when using this mechanism, however, as follows:

1) The PASS mechanism does not sign the Contact URI value, and thus a malicious party can change the value without detection. For most SIP use scenarios, this is no worse than [RFC4474], since Record-Route and Path header fields can be added into [RFC4474] signed SIP requests as well to accomplish the same malicious goal. The Contact URI is usable, however, in cases where Record-Route and Path do not apply, for example to generate subsequent out-of-dialog requests to a GRUU Contact; in such cases the PASS mechanism is weaker than [RFC4474].

2) The PASS mechanism does not typically sign the entire SIP message SDP body, and therefore much of the SDP can be changed without detection. Although [RFC4474] only signs bodies when they are in requests, which is not always the case for SDP, if the SDP body *is* in the request then [RFC4474] assures the Verifier that it has not been changed by any node beyond the Authenticator. For SDP, such assurance does not guarantee media identity (see [draft-baiting]), but [RFC4474] is better than nothing. The PASS mechanism allows the signer to decide what to sign, by design, specifically to allow intermediaries the ability to change SDP because this is required for many deployments.

3) The PASS mechanism allows any node with a valid verifiable signature to sign a PAI asserted identity of any user of any domain. [RFC4474] only allows the domain identified by the From URI addr-spec value to sign the request. For example, thief.com could assert a PAI for bank.com, and as long as thief.com’s certificate is valid, the verification would pass. This behavior is essentially by design: the PASS mechanism
validates the *asserter* of PAI, not the PAI’s value itself. The goal of the mechanism is to provide easy tracking of who asserted PAI, and potentially provide a reputation system for such. For example thief.com would be identifiable as the source of malicious PAI assertion, and their future requests could be rejected.

13. IANA Considerations

This document makes no request of Internet Assigned Numbers Authority (IANA) currently.

14. Acknowledgements

Thanks to John Elwell, Dan Wing, Dean Willis, and Kai Fischer for discussion regarding the idea behind this draft.

15. References

15.1. Normative References


15.2. Informative References


Author’s Address

Hadriel Kaplan
Acme Packet
71 Third Ave.
Burlington, MA 01803, USA
Email: hkaplan@acmepacket.com