Relay Extensions for the Message Session Relay Protocol (MSRP)

Abstract

   Two separate models for conveying instant messages have been defined.
   Page-mode messages stand alone and are not part of a Session
   Initiation Protocol (SIP) session, whereas session-mode messages are
   set up as part of a session using SIP. The Message Session Relay
   Protocol (MSRP) is a protocol for near real-time, peer-to-peer
   exchanges of binary content without intermediaries, which is designed
   to be signaled using a separate rendezvous protocol such as SIP.
   This document introduces the notion of message relay intermediaries
   to MSRP and describes the extensions necessary to use them.

Table of Contents

1. Introduction and Requirements

   There are a number of scenarios in which using a separate protocol
   for bulk messaging is desirable. In particular, there is a need to
   handle a sequence of messages as a session of media initiated using
   SIP [8], just like any other media type. The Message Session Relay
   Protocol (MSRP) [11] is used to convey a session of messages directly
   between two end systems with no intermediaries. With MSRP, messages
   can be arbitrarily large and all traffic is sent over reliable,
   congestion-safe transports.

   This document describes extensions to the core MSRP protocol to
   introduce intermediaries called relays. With these extensions, MSRP
   clients can communicate directly, or through an arbitrary number of
   relays. Each client is responsible for identifying any relays acting
   on its behalf and providing appropriate credentials. Clients that
   can receive new TCP connections directly do not have to implement any
   new functionality to work with these relays.

   The goals of the MSRP relay extensions are listed below:

   o convey arbitrary binary MIME data without modification or transfer
      encoding

   o continue to support client-to-client operation (no relay servers
      required)

   o operate through an arbitrary number of relays for policy
      enforcement

   o operate through relays under differing administrative control

   o allow each client to control which relays are traversed on its
      behalf

   o prevent unsolicited messages (spam), "open relays", and Denial of
      Service (DoS) amplification

   o allow relays to use one or a small number of TCP or TLS [2]
      connections to carry messages for multiple sessions, recipients,
      and senders

   o allow large messages to be sent over slow connections without
      causing head-of-line blocking problems

   o allow transmissions of large messages to be interrupted and
      resumed in places where network connectivity is lost and later
      reestablished

   o offer notification of message failure at any intermediary

   o allow relays to delete state after a short amount of time

2. Conventions and Definitions

   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 [9].

   Below we list several definitions important to MSRP:

   MSRP node: a host that implements the MSRP protocols as a client or a
      relay.

   MSRP client: an MSRP node that is the initial sender or final target
      of messages and delivery status.

   MSRP relay: an MSRP node that forwards messages and delivery status
      and may provide policy enforcement. Relays can fragment and
      reassemble portions of messages.

   Message: arbitrary MIME [13][14] content that one client wishes to
      send to another. For the purposes of this specification, a
      complete MIME body as opposed to a portion of a complete message.

   chunk: a portion of a complete message delivered in a SEND request.

   end-to-end: delivery of data from the initiating client to the final
      target client.

   hop: delivery of data between one MSRP node and an adjacent node.

3. Protocol Overview

   With the introduction of this extension, MSRP has the concept of both
   clients and relays. Clients send messages to relays and/or other
   clients. Relays forward messages and message delivery status to
   clients and other relays. Clients that can open TCP connections to
   each other without intervening policy restrictions can communicate
   directly with each other. Clients who are behind firewalls or who
   need to use intermediaries for policy reasons can use the services of
   a relay. Each client is responsible for enlisting the assistance of
   one or more relays for its side of the communication.

   Clients that use a relay operate by first opening a TLS connection
   with a relay, authenticating, and retrieving an msrps: URI (from the
   relay) that the client can provide to its peers to receive messages

later. There are several steps for doing this. First, the client
opens a TLS connection to its first relay, and verifies that the name
in the certificate matches the name of the relay to which it is
trying to connect. Such verification is performed according to the
procedures defined in Section 9.2. After verifying that it has
connected to the proper host, the client authenticates itself to the
relay using an AUTH request containing appropriate authentication
credentials. In a successful AUTH response, the relay provides an
msrps: URI associated with the path back to the client. The client
can then give this URI to other clients for end-to-end message
delivery.

When clients wish to send a short message, they issue a SEND request
with the entire contents of the message. If any relays are required,
they are included in the To-Path header. The leftmost URI in the To-
Path header is the next hop to deliver a request or response. The
rightmost URI in the To-Path header is the final target.

SEND requests contain headers that indicate how they are acknowledged
in a hop-by-hop form and in an end-to-end form. The default is that
SEND messages are acknowledged hop-by-hop. (Each relay that receives
a SEND request acknowledges receipt of the request before forwarding
the content to the next relay or the final target.) All other
requests are acknowledged end-to-end.

With the introduction of relays, the subtle semantics of the To-Path
header and the From-Path header become more relevant. The To-Path in
both requests and responses is the list of URIs that need to be
visited in order to reach the final target of the request or
response. The From-Path is the list of URIs that indicate how to get
back to the original sender of the request or response. These
headers differ from the To and From headers in SIP, which do not
"swap" from request to response. (Note that sometimes a request is
sent to or from an intermediary directly.)

When a relay forwards a request, it removes its address from the To-
Path header and inserts it as the first URI in the From-Path header.
For example, if the path from Alice to Bob is through relays A and B,
when B receives the request it contains path headers that look like
the following. (Note that MSRP does not permit line folding. A "\"
in the examples shows a line continuation due to limitations in line
length of this document. Neither the backslash nor the extra CRLF is
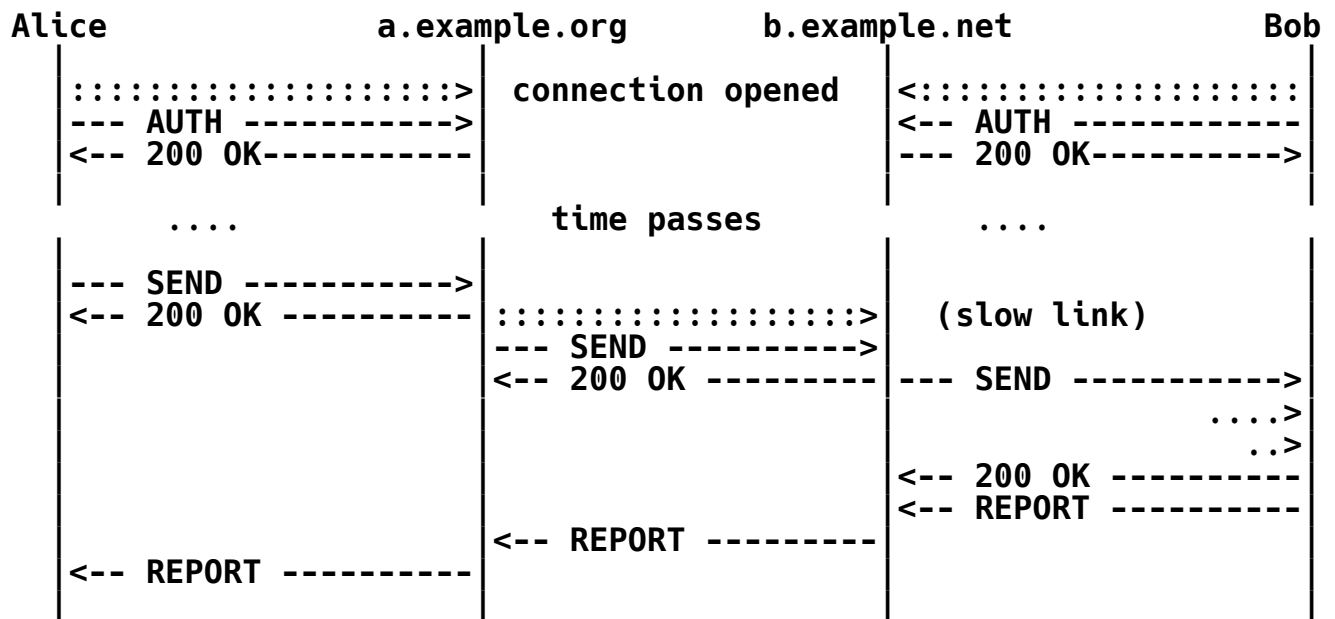included in the actual request or response.)

```
To-Path:   msrps://B.example.com/bbb;tcp \
           msrps://Bob.example.com/bob;tcp
From-Path: msrps://A.example.com/aaa;tcp \
           msrps://Alice.example.com/alice;tcp
```

After forwarding the request, the path headers look like this:

```
To-Path: msrps://Bob.example.com/bob;tcp
From-Path: msrps://B.example.com/bbb;tcp \
           msrps://A.example.com/aaa;tcp \
           msrps://Alice.example.com/alice;tcp
```

The sending of an acknowledgment for SEND requests is controlled by the Success-Report and Failure-Report headers and works the same way as in the base MSRP protocol. When a relay receives a SEND request, if the Failure-Report is set to "yes", it means that the previous hop is running a timer and the relay needs to send a response to the request. If the final response conveys an error, the previous hop is responsible for constructing the error report and sending it back to the original sender of the message. The 200 response acknowledges receipt of the request so that the previous hop knows that it is no longer responsible for the request. If the relay knows that it will not be able to deliver the request and the Failure-Report is set to any value other than "no", then it sends a REPORT to tell the sender about the error. If the Failure-Report is set to "yes", then after the relay is done sending the request to the next hop it starts running a timer; if the timer expires before a response is received from the next hop, the relay assumes that an error has happened and sends a REPORT to the sender. If the Failure-Report is not set to "yes", there is no need for the relay to run this timer.

The following example shows a typical MSRP session. The AUTH requests are explained in a later section but left in the example for call flow completeness.

```
   Alice                 a.example.org       b.example.net          Bob
     |::::::::::::::::::::>| connection opened |<:::::::::::::::::::::|
     |--- AUTH ----------->|                   |<-- AUTH -------------|
     |<-- 200 OK-----------|                   |--- 200 OK---------->|
     |                     |                   |                     |
     |    ....             |    time passes    |    ....             |
     |                     |                   |                     |
     |--- SEND ----------->|                   |                     |
     |<-- 200 OK ----------|::::::::::::::::::::>| (slow link)        |
     |                     |--- SEND ---------->|                     |
     |                     |<-- 200 OK ---------|--- SEND ----------->|
     |                     |                   |                ....>|
     |                     |                   |                  ..>|
     |                     |                   |<-- 200 OK ----------|
     |                     |                   |<-- REPORT ----------|
     |                     |<-- REPORT ---------|                     |
     |<-- REPORT ----------|                   |                     |
     |                     |                   |                     |
```

   The SEND and REPORT messages are shown below to illustrate the To-
   Path and From-Path headers. (Note that MSRP does not permit line
   folding. A "\" in the examples shows a line continuation due to
   limitations in line length of this document. Neither the backslash,
   nor the extra CRLF is included in the actual request or response.)

```
  MSRP 6aef SEND
  To-Path: msrps://a.example.org:9000/kjfjan;tcp \
           msrps://b.example.net:9000/aeiug;tcp \
           msrps://bob.example.net:8145/foo;tcp
  From-Path: msrps://alice.example.org:7965/bar;tcp
  Success-Report: yes
  Byte-Range: 1-*/*
  Message-ID: 87652
  Content-Type: text/plain

  Hi Bob, I'm about to send you file.mpeg
  -------6aef$


  MSRP 6aef 200 OK
  To-Path: msrps://alice.example.org:7965/bar;tcp
  From-Path: msrps://a.example.org:9000/kjfjan;tcp
  Message-ID: 87652
  -------6aef$
```

```
MSRP juh76 SEND
To-Path: msrps://b.example.net:9000/aeiug;tcp \
         msrps://bob.example.net:8145/foo;tcp
From-Path: msrps://a.example.org:9000/kjfjan;tcp \
           msrps://alice.example.org:7965/bar;tcp
Success-Report: yes
Message-ID: 87652
Byte-Range: 1-*/*
Content-Type: text/plain

Hi Bob, I'm about to send you file.mpeg
-------juh76$


MSRP juh76 200 OK
To-Path: msrps://a.example.org:9000/kjfjan;tcp
From-Path: msrps://b.example.net:9000/aeiug;tcp
Message-ID: 87652
-------juh76$

MSRP xght6 SEND
To-Path: msrps://bob.example.net:8145/foo;tcp
From-Path: msrps://b.example.net:9000/aeiug;tcp \
           msrps://a.example.org:9000/kjfjan;tcp \
           msrps://alice.example.org:7965/bar;tcp
Success-Report: yes
Message-ID: 87652
Byte-Range: 1-*/*
Content-Type: text/plain

Hi Bob, I'm about to send you file.mpeg
-------xght6$


MSRP xght6 200 OK
To-Path: msrps://b.example.net:9000/aeiug;tcp
From-Path: msrps://bob.example.net:8145/foo;tcp
Message-ID: 87652
```

```
   MSRP yh67 REPORT
   To-Path: msrps://b.example.net:9000/aeiug;tcp \
           msrps://a.example.org:9000/kjfjan;tcp \
           msrps://alice.example.org:7965/bar;tcp
   From-Path: msrps://bob.example.net:8145/foo;tcp
   Message-ID: 87652
   Byte-Range: 1-39/39
   Status: 000 200 OK
   -------yh67$


   MSRP yh67 REPORT
   To-Path: msrps://a.example.org:9000/kjfjan;tcp \
           msrps://alice.example.org:7965/bar;tcp
   From-Path: msrps://b.example.net:9000/aeiug;tcp \
             msrps://bob.example.net:8145/foo;tcp
   Message-ID: 87652
   Byte-Range: 1-39/39
   Status: 000 200 OK
   -------yh67$


   MSRP yh67 REPORT
   To-Path: msrps://alice.example.org:7965/bar;tcp
   From-Path: msrps://a.example.org:9000/kjfjan;tcp \
             msrps://b.example.net:9000/aeiug;tcp \
             msrps://bob.example.net:8145/foo;tcp
   Message-ID: 87652
   Byte-Range: 1-39/39
   Status: 000 200 OK
   -------yh67$
```

   When sending large content, the client may split up a message into
   smaller pieces; each SEND request might contain only a portion of the
   complete message. For example, when Alice sends Bob a 4-GB file
   called "file.mpeg", she sends several SEND requests each with a
   portion of the complete message. Relays can repack message fragments
   en route. As individual parts of the complete message arrive at the
   final destination client, the receiving client can optionally send
   REPORT requests indicating delivery status.

   MSRP nodes can send individual portions of a complete message in
   multiple SEND requests. As relays receive chunks, they can
   reassemble or re-fragment them as long as they resend the resulting
   chunks in order. (Receivers still need to be prepared to receive
   out-of-order chunks, however.) If the sender has set the Success-
   Report header to "yes", once a chunk or complete message arrives at
   the destination client, the destination will send a REPORT request

indicating that a chunk arrived end-to-end. This request travels
back along the reverse path of the SEND request. Unlike the SEND
request, which can be acknowledged along every hop, REPORT requests
are never acknowledged.

The following example shows a message being re-chunked through two
relays:

```
Alice                   a.example.org       b.example.net           Bob
  |                         |                     |                   |
  |--- SEND 1-3 ------->    |                     |                   |
  |<-- 200 OK ----------    |              (slow link)                |
  |--- SEND 4-7 ------->|--- SEND 1-5 ------>|                        |
  |<-- 200 OK ----------|<-- 200 OK ---------|--- SEND 1-3 ------>    |
  |--- SEND 8-10 ------>|--- SEND 6-10 ----->|                 ....>  |
  |<-- 200 OK ----------|<-- 200 OK ---------|                   ..>  |
  |                     |                    |<-- 200 OK ----------    |
  |                     |                    |<-- REPORT 1-3 ------    |
  |                     |<-- REPORT 1-3 -----|--- SEND 4-7 ------->    |
  |<-- REPORT 1-3 ------|                    |                 ...>    |
  |                     |                    |<-- REPORT 4-7 ----->    |
  |                     |<-- REPORT 4-7 -----|--- SEND 8-10 ------>    |
  |<-- REPORT 4-7 ------|                    |                  ..>    |
  |                     |                    |<-- 200 OK ----------    |
  |                     |<-- REPORT done-----|<-- REPORT done -----    |
  |<-- REPORT done -----|                    |                        |
  |                     |                    |                        |
```

Relays only keep transaction states for a short time for each chunk.
Delivery over each hop should take no more than 30 seconds after the
last byte of data is sent. Client applications define their own
implementation-dependent timers for end-to-end message delivery.

For client-to-client communication, the sender of a message typically
opens a new TCP connection (with or without TLS) if one is needed.
Relays reuse existing connections first, but can open new connections
(typically to other relays) to deliver requests such as SEND or
REPORT. Responses can only be sent over existing connections.

The relationship between MSRP and signaling protocols (such as SIP)
is unchanged by this document, and is as described in [11]. An
example of an SDP exchange for an MSRP session involving relays is
shown in Section 11.

3.1. Authorization Overview

   A key element of this protocol is that it cannot introduce open
   relays, with all the associated problems they create, including DoS
   attacks. A message is only forwarded by a relay if it is either
   going to or coming from a client that has authenticated to the relay
   and been authorized for relaying messages on that particular session.
   Because of this, clients use an AUTH message to authenticate to a
   relay and get a URI that can be used for forwarding messages.

   If a client wishes to use a relay, it sends an AUTH request to the
   relay. The client authenticates the relay using the relay's TLS
   certificate. The client uses HTTP Digest authentication [1] to
   authenticate to the relay. When the authentication succeeds, the
   relay returns a 200 response that contains the URI that the client
   can use in the MSRP path for the relay.

   A typical challenge response flow is shown below:

```
Alice                     a.example.org
  │                           │
  │:::::::::::::::::::::>      │
  │--- AUTH ------------>      │
  │<- 401 Unauthorized -      │
  │--- AUTH ------------>      │
  │<-- 200 OK-----------       │
  │                           │
```

   The URI that the client should use is returned in the Use-Path header
   of the 200.

   Note that URIs returned to the client are effectively secret tokens
   that should be shared only with the other MSRP client in a session.
   For that reason, the client MUST NOT reuse the same URI for multiple
   sessions, and needs to protect these URIs from eavesdropping.

4. New Protocol Elements

4.1. The AUTH Method

   AUTH requests are used by clients to create a handle they can use to
   receive incoming requests. AUTH requests also contain credentials
   used to authenticate a client and authorization policy used to block
   Denial of Service attacks.

   In response to an AUTH request, a successful response contains a Use-
   Path header with a list of URIs that the client can give to its peers
   to route responses back to the client.

4.2. The Use-Path Header

The Use-Path header is a list of URIs provided by an MSRP relay in
response to a successful AUTH request. This list of URIs can be used
by the MSRP client that sent the AUTH request to receive MSRP
requests and to advertise this list of URIs, for example, in a
session description. URIs in the Use-Path header MUST include a
fully qualified domain name (as opposed to a numeric IP address) and
an explicit port number.

The URIs in the Use-Path header are in the same order that the
authenticating client uses them in a To-Path header. Instructions on
forming To-Path headers and SDP [7] path attributes from information
in the Use-Path header are provided in Section 5.1.

4.3. The HTTP Authentication "WWW-Authenticate" Header

The "WWW-Authenticate" header contains a challenge token used in the
HTTP Digest authentication procedure (from RFC 2617 [1]). The usage
of HTTP Digest authentication in MSRP is described in detail in
Section 5.1.

4.4. The HTTP Authentication "Authorization" Header

The "Authorization" header contains authentication credentials for
HTTP Digest authentication (from RFC 2617 [1]). The usage of HTTP
Digest authentication in MSRP is described in detail in Section 5.1.

4.5. The HTTP Authentication "Authentication-Info" Header

The "Authentication-Info" header contains future challenges to be
used for HTTP Digest authentication (from RFC 2617 [1]). The usage
of HTTP Digest authentication in MSRP is described in detail in
Section 5.1.

4.6. Time-Related Headers

The Expires header in a request provides a relative time after which
the action implied by the method of the request is no longer of
interest. In a request, the Expires header indicates how long the
sender would like the request to remain valid. In a response, the
Expires header indicates how long the responder considers this
information relevant. Specifically, an Expires header in an AUTH
request indicates how long the provided URIs will be valid.

The Min-Expires header contains the minimum duration a server will
permit in an Expires header. It is sent only in 423 "Interval Out-
of-Bounds" responses. Likewise, the Max-Expires header contains the
maximum duration a server will permit in an Expires header.

5. Client Behavior

5.1. Connecting to Relays Acting on Your Behalf

Clients that want to use the services of a relay or list of relays
need to send an AUTH request to each relay that will act on their
behalf. (For example, some organizations could deploy an "intra-org"
relay and an "extra-org" relay.) The inner relay is used to tunnel
the AUTH requests to the outer relay. For example, the client will
send an AUTH to intra-org and get back a path that can be used for
forwarding through intra-org. The client would then send a second
AUTH destined to extra-org but sent through intra-org. The intra-org
relay forwards this to extra-org and extra-org returns a path that
can be used to forward messages from another destination to extra-org
to intra-org and then on to this client. Each relay authenticates
the client. The client authenticates the first relay and each relay
authenticates the next relay.

Clients can be configured (typically, through discovery or manual
provisioning) with a list of relays they need to use. They MUST be
able to form a connection to the first relay and send an AUTH command
to get a URI that can be used in a To-Path header. The client can
authenticate its first relay by looking at the relay's TLS
certificate. The client MUST authenticate itself to each of its
relays using HTTP Digest authentication [1] (see Section 9.1 for
details).

The relay returns a URI, or list of URIs, in the "Use-Path" header of
a success response. Each URI SHOULD be used for only one unique
session. These URIs are used by the client in the path attribute
that is sent in the SDP to set up the session, and in the To-Path
header of outgoing requests. To form the To-Path header for outgoing
requests, the client takes the list of URIs in the Use-Path header
after the outermost authentication and appends the list of URIs
provided in the path attribute in the peer's session description. To
form the SDP path attribute to provide to the peer, the client
reverses the list of URIs in the Use-Path header (after the outermost
authentication), and appends the client's own URI.

   For example, "A" has to traverse its own relays "B" and "C", and
   then relays "D" and "E" in domain2 to reach "F". Client "A" will
   authenticate with its relays "B" and "C" and eventually receive a
   Use-Path header containing "B C". Client "A" reverses the list

(now "C B") and appends its own URI (now "C B A"), and provides
this list to "F" in a path SDP attribute. Client "F" sends its
SDP path list "D E F", which client "A" appends to the Use-Path
list it received "B C". The resulting To-Path header is "B C D E
F".

```
   domain 1                        domain 2
 ----------------              -----------------

 client     relays          relays     client
   A ----- B -- C -------- D -- E ----- F

 Use-Path returned by C:          B C
 path: attribute generated by A:  C B A
 path: attribute received from F: D E F
 To-Path header generated by A:   B C D E F
```

The initial AUTH request sent to a relay by a client will generally
not contain an Authorization header, since the client has no
challenge to which it can respond. In response to an AUTH request
that does not contain an Authorization header, a relay MUST respond
with a "401 Unauthorized" response containing a WWW-Authenticate
header. The WWW-Authenticate header is formed as described in RFC
2617 [1], with the restrictions and modifications described in
Section 9.1. The realm chosen by the MSRP relay in such a challenge
is a matter of administrative policy. Because a single relay does
not have multiple protection spaces in MSRP, it is not unreasonable
to always use the relay's hostname as the realm.

Upon receiving a 401 response to a request, the client SHOULD fetch
the realm from the WWW-Authenticate header in the response and retry
the request, including an Authorization header with the correct
credentials for the realm. The Authorization header is formed as
described in RFC 2617 [1], with the restrictions and modifications
described in Section 9.1.

When a client wishes to use more than one relay, it MUST send an AUTH
request to each relay it wishes to use. Consider a client A, that
wishes messages to flow from A to the first relay, R1, then on to a
second relay, R2. This client will do a normal AUTH with R1. It
will then do an AUTH transaction with R2 that is routed through R1.
The client will form this AUTH message by setting the To-Path to
msrps://R1;tcp msrps://R2;tcp. R1 will forward this request onward
to R2.

When sending an AUTH request, the client MAY add an Expires header to
request a MSRP URI that is valid for no longer than the provided
interval (a whole number of seconds). The server will include an

Expires header in a successful response indicating how long its URI
from the Use-Path will be valid. Note that each server can return an
independent expiration time.

Note that MSRP does not permit line folding. A "\" in the examples
shows a line continuation due to limitations in line length of this
document. Neither the backslash nor the extra CRLF is included in
the actual request or response.

(Alice opens a TLS connection to intra.example.com and sends an AUTH
request to initiate the authentication process.)

```
 MSRP 49fh AUTH
 To-Path: msrps://alice@intra.example.com;tcp
 From-Path: msrps://alice.example.com:9892/98cjs;tcp
 -------49fh$
```

(Alice's relay challenges the AUTH request.)

```
 MSRP 49fh 401 Unauthorized
 To-Path: msrps://alice.example.com:9892/98cjs;tcp
 From-Path: msrps://alice@intra.example.com;tcp
 WWW-Authenticate: Digest realm="intra.example.com", qop="auth", \
                 nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093"
 -------49fh$
```

(Alice responds to the challenge.)

```
 MSRP 49fi AUTH
 To-Path: msrps://alice@intra.example.com;tcp
 From-Path: msrps://alice.example.com:9892/98cjs;tcp
 Authorization: Digest username="Alice",
                 realm="intra.example.com", \
                 nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093", \
                 qop=auth, nc=00000001, cnonce="0a4f113b", \
                 response="6629fae49393a05397450978507c4ef1"
 -------49fi$
```

(Alice's relay confirms that Alice is an authorized user. As a
matter of local policy, it includes an "Authentication-Info" header
with a new nonce value to expedite future AUTH requests.)

```
   MSRP 49fi 200 OK
   To-Path: msrps://alice.example.com:9892/98cjs;tcp
   From-Path: msrps://alice@intra.example.com;tcp
   Use-Path: msrps://intra.example.com:9000/jui787s2f;tcp
   Authentication-Info: nextnonce="40f2e879449675f288476d772627370a",\
                        rspauth="7327570c586207eca2afae94fc20903d", \
                        cnonce="0a4f113b", nc=00000001, qop=auth
   Expires: 900
   -------49fi$
```

(Alice now sends an AUTH request to her "external" relay through her
"internal" relay, using the URI she just obtained; the AUTH request
is challenged.)

```
   MSRP mnbvw AUTH
   To-Path: msrps://intra.example.com:9000/jui787s2f;tcp \
            msrps://extra.example.com;tcp
   From-Path: msrps://alice.example.com:9892/98cjs;tcp
   -------mnbvw$

   MSRP m2nbvw AUTH
   To-Path: msrps://extra.example.com;tcp
   From-Path: msrps://intra.example.com:9000/jui787s2f;tcp \
            msrps://alice.example.com:9892/98cjs;tcp
   -------m2nbvw$

   MSRP m2nbvw 401 Unauthorized
   To-Path: msrps://intra.example.com:9000/jui787s2f;tcp \
            msrps://alice.example.com:9892/98cjs;tcp
   From-Path: msrps://extra.example.com;tcp
   WWW-Authenticate: Digest realm="extra.example.com", qop="auth", \
                     nonce="Uumu8cAV38FGsEF31VLevIbNXj9HWO"
   -------m2nbvw$

   MSRP mnbvw 401 Unauthorized
   To-Path: msrps://alice.example.com:9892/98cjs;tcp
   From-Path: msrps://intra.example.com:9000/jui787s2f;tcp \
            msrps://extra.example.com;tcp
   WWW-Authenticate: Digest realm="extra.example.com", qop="auth", \
                     nonce="Uumu8cAV38FGsEF31VLevIbNXj9HWO"
   -------mnbvw$
```

(Alice replies to the challenge with her credentials and is then
authorized to use the "external" relay).

```
MSRP m3nbvx AUTH
To-Path: msrps://intra.example.com:9000/jui787s2f;tcp \
        msrps://extra.example.com;tcp
From-Path: msrps://alice.example.com:9892/98cjs;tcp
Authorization: Digest username="Alice",
              realm="extra.example.com", \
              nonce="Uumu8cAV38FGsEF31VLevIbNXj9HWO", \
              qop=auth, nc=00000001, cnonce="85a0dca8", \
              response="cb06c4a77cd90918cd7914432032e0e6"
-------m3nbvx$

MSRP m4nbvx AUTH
To-Path: msrps://extra.example.com;tcp
From-Path: msrps://intra.example.com:9000/jui787s2f;tcp \
        msrps://alice.example.com:9892/98cjs;tcp
Authorization: Digest username="Alice",
              realm="extra.example.com", \
              nonce="Uumu8cAV38FGsEF31VLevIbNXj9HWO", \
              qop=auth, nc=00000001, cnonce="85a0dca8", \
              response="cb06c4a77cd90918cd7914432032e0e6"
-------m4nbvx$

MSRP m4nbvx 200 OK
To-Path: msrps://intra.example.com:9000/jui787s2f;tcp \
        msrps://alice.example.com:9892/98cjs;tcp
From-Path: msrps://extra.example.com;tcp
Use-Path: msrps://intra.example.com:9000/jui787s2f;tcp \
        msrps://extra.example.com:9000/mywdEe1233;tcp
Authentication-Info: nextnonce="bz8V080GEA2sLyEDpITF2AZCq7gIkc", \
                    rspauth="72f109ed2755d7ed0d0a213ec653b3f2", \
                    cnonce="85a0dca8", nc=00000001, qop=auth
Expires: 1800
-------m4nbvx$

MSRP m3nbvx 200 OK
To-Path: msrps://alice.example.com:9892/98cjs;tcp
From-Path: msrps://intra.example.com:9000/jui787s2f;tcp \
        msrps://extra.example.com;tcp
Use-Path: msrps://extra.example.com:9000/mywdEe1233;tcp \
        msrps://extra.example.com:9000/mywdEe1233;tcp
Authentication-Info: nextnonce="bz8V080GEA2sLyEDpITF2AZCq7gIkc", \
                    rspauth="72f109ed2755d7ed0d0a213ec653b3f2", \
                    cnonce="85a0dca8", nc=00000001, qop=auth
Expires: 1800
-------m3nbvx$
```

5.2. Sending Requests

The procedure for forming SEND and REPORT requests is identical for
clients whether or not relays are involved. The specific procedures
are described in Section 7 of the core MSRP protocol.

As usual, once the next-hop URI is determined, the client MUST find
the appropriate address, port, and transport to use and then check if
there is already a suitable existing connection to the next-hop
target. If so, the client MUST send the request over the most
suitable connection. Suitability MAY be determined by a variety of
factors such as measured load and local policy, but in most simple
implementations a connection will be suitable if it exists and is
active.

5.3. Receiving Requests

The procedure for receiving requests is identical for clients whether
or not relays are involved.

5.4. Managing Connections

Clients should open a connection whenever they wish to deliver a
request and no suitable connection exists. For connections to
relays, the client should leave a connection up until no sessions
have used it for a locally defined period of time, which defaults to
5 minutes for foreign relays and one hour for the client's relays.

6. Relay Behavior

6.1. Handling Incoming Connections

When a relay receives an incoming connection on a port configured for
TLS, it includes a client CertificateRequest in the same record in
which it sends its ServerHello. If the TLS client provides a
certificate, the server verifies it and continues if the certificate
is valid and rooted in a trusted authority. If the TLS client does
not provide a certificate, the server assumes that the client is an
MSRP endpoint and invokes Digest authentication. Once a TCP or TLS
channel is negotiated, the server waits for up to 30 seconds to
receive an MSRP request over the channel. If no request is received
in that time, the server closes the connection. If no successful
requests are sent during this probationary period, the server closes
the connection. Likewise, if several unsuccessful requests are sent
during the probation period and no requests are sent successfully,
the server SHOULD close the connection.

6.2. Generic Request Behavior

   Upon receiving a new request, relays first verify the validity of the
   request. Relays then examine the first URI in the To-Path header and
   remove this URI if it matches a URI corresponding to the relay. If
   the request is not addressed to the relay, the relay immediately
   drops the corresponding connection over which the request was
   received.

6.3. Receiving AUTH Requests

   When a relay receives an AUTH request, the first thing it does is to
   authenticate and authorize the previous hop and the client at the far
   end. If there are no other relays between this relay and the client,
   then these are the same thing.

   When the previous hop is a relay, authentication is done with TLS
   using mutual authentication. If the TLS client presented a host
   certificate, the relay checks that the subjectAltName in the
   certificate of the TLS client matches the hostname in the first From-
   Path URI. If the TLS client doesn't provide a host certificate, the
   relay assumes the TLS client is an MSRP client and sends it a
   challenge.

   Authorization is a matter of local policy at the relay. Many relays
   will choose to authorize all relays that can be authenticated,
   possibly in conjunction with a blacklisting mechanism. Relays
   intended to operate only within a limited federation may choose to
   authorize only those relays whose identity appears in a provisioned
   list. Other authorization policies may also be applied.

   When the previous hop is a client, the previous hop is the same as
   the identity of the client. The relay checks the credentials
   (username and password) provided by the client in the Authorization
   header and checks if this client is allowed to use the relay. If the
   client is not authorized, the relay returns a 403 response. If the
   client has requested a particular expiration time in an Expires
   header, the relay needs to check that the time is acceptable to it
   and, if not, return an error containing a Min-Expires or Max-Expires
   header, as appropriate.

   Next the relay will generate an MSRP URI that allows messages to be
   forwarded to or from this previous hop. If the previous hop was a
   relay authenticated by mutual TLS, then the URI MUST be valid to
   route across any connection the relay has to the previous hop relay.
   If the previous hop is a client, then the URI MUST only be valid to

route across the same connection over which the AUTH request was
received. If the client's connection is closed and then reopened,
the URI MUST be invalidated.

If the AUTH request contains an Expires header, the relay MUST ensure
that the URI is invalidated after the expiry time. The URI MUST
contain at least 64 bits of cryptographically random material so that
it is not guessable by attackers. If a relay is requested to forward
a message for which the URI is not valid, the relay MUST discard the
message and MAY send a REPORT indicating that the AUTH URI was bad.

A successful AUTH response returns a Use-Path header that contains an
MSRP URI that the client can use. It also returns an Expires header
that indicates how long the URI will be valid (expressed as a whole
number of seconds).

If a relay receives several unsuccessful AUTH requests from a client
that is directly connected to it via TLS, the relay SHOULD terminate
the corresponding connection. Similarly, if a relay forwards several
failed AUTH requests to the same destination that originate from a
client that is directly connected to it via TLS, the relay SHOULD
terminate the corresponding connection. Determination of a remote
AUTH failure can be made by observing an AUTH request containing an
Authorization header that triggers a 401 response without a
"stale=TRUE" indication. These preventive measures apply only to a
connection between a relay and a client; a relay SHOULD NOT use
excessive AUTH request failures as a reason to terminate a connection
with another relay.

6.4. Forwarding

Before any request is forwarded, the relay MUST check that the first
URI in the To-Path header corresponds to a URI that this relay has
created and handed out in the Use-Path header of an AUTH request.
Next it verifies that either 1) the next hop is the next hop back
toward the client that obtained this URI, or 2) the previous hop was
the correct previous hop coming from the client that obtained this
URI.

Since transact-id values are not allowed to conflict on a given
connection, a relay will generally need to construct a new transact-
id value for any request that it forwards.

6.4.1. Forwarding SEND Requests

If an incoming SEND request contains a Failure-Report header with a
value of "yes", an MSRP relay that receives that SEND request MUST
respond with a final response immediately. A 200-class response
indicates the successful receipt of a message fragment but does not
mean that the message has been forwarded on to the next hop. The
final response to the SEND MUST be sent only to the previous hop,
which could be an MSRP relay or the original sender of the SEND
request.

If the Failure-Report header is "yes", then the relay MUST run a
timer to detect if transmission to the next hop fails. The timer
starts when the last byte of the message has been sent to the next
hop. If after 30 seconds the next hop has not sent any response,
then the relay MUST construct a REPORT with a status code of 408 to
indicate a timeout error happened sending the message, and send the
REPORT to the original sender of the message.

If the Failure-Report header is "yes" or "partial", and if there is a
problem processing the SEND request or if an error response is
received for that SEND request, then the relay MUST respond with an
appropriate error response in a REPORT back to the original source of
the message.

The MSRP relay MAY further break up the message fragment received in
the SEND request into smaller fragments and forward them to the next
hop in separate SEND requests. It MAY also combine message fragments
received before or after this SEND request, and forward them out in a
single SEND request to the next hop identified in the To-Path header.
The MSRP relay MUST NOT combine message fragments from SEND requests
with different values in the Message-ID header.

The MSRP relay MAY choose whether to further fragment the message, or
combine message fragments, or send the message as is, based on some
policy that is administered, or based on the network speed to the
next hop, or any other mechanism.

If the MSRP relay has knowledge of the byte range that it will
transmit to the next hop, it SHOULD update the Byte-Range header in
the SEND request appropriately.

Before forwarding the SEND request to the next hop, the MSRP relay
MUST inspect the first URI in the To-Path header. If it indicates
this relay, the relay removes this URI from the To-Path header and
inserts this URI in the From-Path header before any other URIs. If
it does not indicate this relay, there has been an error in

forwarding at a previous hop. In this case, the relay SHOULD discard
the message, and if the Failure-Report header is set to "yes", the
relay SHOULD generate a failure report.

6.4.2. Forwarding Non-SEND Requests

An MSRP relay that receives any request other than a SEND request
(including new methods unknown to the relay) first follows the
validation and authorization rules for all requests. Then the relay
moves its URI from the beginning of the To-Path headers to the
beginning of the From-Path header and forwards the request on to the
next hop. If it already has a connection to the next hop, it SHOULD
use this connection and not form a new connection. If no suitable
connection exists, the relay opens a new connection.

Requests with an unknown method are forwarded as if they were REPORT
requests. An MSRP node MAY be configured to block unknown methods
for security reasons.

6.4.3. Handling Responses

Relays receiving a response first verify that the first URI in the
To-Path corresponds to itself; if not, the response SHOULD be
dropped. Likewise, if the message cannot be parsed, the relay MUST
drop the response. Next the relay determines if there are additional
URIs in the To-Path. (For responses to SEND requests there will be
no additional URIs, whereas responses to AUTH requests have
additional URIs directing the response back to the client.)

If the response matches an existing transaction, then that
transaction is completed and any timers running on it can be removed.
If the response is a non 200 response, and the original request was a
SEND request that had a Failure-Report header with a value other than
"no", then the relay MUST send a REPORT indicating the nature of the
failure. The response code received by the relay is used to form the
status line in the REPORT that the relay sends.

If there are additional URIs in the To-Path header, the relay MUST
then move its URI from the To-Path header, insert its URI in front of
any other URIs in the From-Path header, and forward the response to
the next URI in the To-Path header. The relay sends the request over
the best connection that corresponds to the next URI in the To-Path
header. If this connection has closed, then the response is silently
discarded.

6.5. Managing Connections

   Relays should keep connections open as long as possible. If a
   connection has not been used in a significant time (more than one
   hour), it MAY be closed. If the relay runs out of resources and can
   no longer establish new connections, it SHOULD start closing existing
   connections. It MAY choose to close the connections based on a least
   recently used basis.

7. Formal Syntax

   The following syntax specification uses the Augmented Backus-Naur
   Form (ABNF) as described in RFC 4234 [10].

   ; This ABNF imports all the definitions in the ABNF of RFC 4975.

   header =/ Expires / Min-Expires / Max-Expires / Use-Path /
            WWW-Authenticate / Authorization / Authentication-Info
              ; this adds to the rule in RFC 4975

   mAUTH   = %x41.55.54.48   ; AUTH in caps
   method   =/ mAUTH
                            ; this adds to the rule in RFC 4975

   WWW-Authenticate   = "WWW-Authenticate:" SP "Digest" SP digest-param
                          *("," SP digest-param)

   digest-param   = (realm / nonce / [opaque] / [stale] / [
                          algorithm] / qop-options / [auth-param])

   realm   = "realm=" realm-value
   realm-value   = quoted-string

   auth-param   = token "=" (token / quoted-string)

   nonce   = "nonce=" nonce-value
   nonce-value   = quoted-string
   opaque   = "opaque=" quoted-string
   stale   = "stale=" ("true" / "false")
   algorithm   = "algorithm=" ("MD5" / token)
   qop-options   = "qop=" DQUOTE qop-list DQUOTE
   qop-list   = qop-value *("," qop-value)
   qop-value   = "auth" / token

   Authorization   = "Authorization:" SP credentials

   credentials   = "Digest" SP digest-response
                            *("," SP digest-response)

```
digest-response       = ( username / realm / nonce / response / [
                          algorithm ] / cnonce / [opaque] / message-qop /
                          [nonce-count]  / [auth-param] )

username              = "username=" username-value
username-value        = quoted-string
message-qop           = "qop=" qop-value
cnonce                = "cnonce=" cnonce-value
cnonce-value          = nonce-value
nonce-count           = "nc=" nc-value
nc-value              = 8LHEX
response              = "response=" request-digest
request-digest        = DQUOTE 32LHEX DQUOTE
LHEX                  = DIGIT / %x61-66 ;lowercase a-f

Authentication-Info =   "Authentication-Info:" SP ainfo
                         *("," ainfo)
ainfo               =   nextnonce / message-qop
                         / response-auth / cnonce
                         / nonce-count
nextnonce           =   "nextnonce=" nonce-value
response-auth       =   "rspauth=" response-digest
response-digest     =   DQUOTE *LHEX DQUOTE

Expires     = "Expires:" SP 1*DIGIT
Min-Expires = "Min-Expires:" SP 1*DIGIT
Max-Expires = "Max-Expires:" SP 1*DIGIT

Use-Path = "Use-Path:" SP MSRP-URI *(SP MSRP-URI)
```

8. Finding MSRP Relays

   When resolving an MSRP URI that contains an explicit port number, an
   MSRP node follows the rules in Section 6 of the MSRP base
   specification. MSRP URIs exchanged in SDP and in To-Path and From-
   Path headers in non-AUTH requests MUST have an explicit port number.
   (The only message in this specification that can have an MSRP URI
   without an explicit port number is in the To-Path header in an AUTH
   request.) Similarly, if the authority component of an msrps: URI
   contains an IPv4 address or an IPv6 reference, a port number MUST be
   present.

   The following rules allow MSRP clients to discover MSRP relays more
   easily in AUTH requests. If the authority component contains a
   domain name and an explicit port number is provided, attempt to look
   up a valid address record (A or AAAA) for the domain name. Connect
   using TLS over the default transport (TCP) with the provided port
   number.

If a domain name is provided but no port number, perform a DNS SRV
[4] lookup for the '_msrps' service and '_tcp' transport at the
domain name, and follow the Service Record (SRV) selection algorithm
defined in that specification to select the entry. (An '_msrp'
service is not defined, since AUTH requests are only sent over TLS.)
If no SRVs are found, try an address lookup (A or AAAA) for the
domain name. Connect using TLS over the default transport (TCP) with
the default port number (2855). Note that AUTH requests MUST only be
sent over a TLS-protected channel. An SRV lookup in the example.com
domain might return:

```
;; in example.com.        Pri Wght Port Target
_msrps._tcp    IN SRV     0   1    9000 server1.example.com.
_msrps._tcp    IN SRV     0   2    9000 server2.example.com.
```

If implementing a relay farm, it is RECOMMENDED that each member of
the relay farm have an SRV entry. If any members of the farm have
multiple IP addresses (for example, an IPv4 and an IPv6 address),
each of these addresses SHOULD be registered in DNS as separate A or
AAAA records corresponding to a single target.

9. Security Considerations

This section first describes the security mechanisms available for
use in MSRP. Then the threat model is presented. Finally, we list
implementation requirements related to security.

9.1. Using HTTP Authentication

AUTH requests MUST be authenticated. The authentication mechanism
described in this specification uses HTTP Digest authentication.
HTTP Digest authentication is performed as described in RFC 2617 [1],
with the following restrictions and modifications:

o Clients MUST NOT attempt to use Basic authentication, and relays
   MUST NOT request or accept Basic authentication.

o The use of a qop value of auth-int makes no sense for MSRP.
   Integrity protection is provided by the use of TLS. Consequently,
   MSRP relays MUST NOT indicate a qop of auth-int in a challenge.

o The interaction between the MD5-sess algorithm and the nextnonce
   mechanism is underspecified in RFC 2617 [1]; consequently, MSRP
   relays MUST NOT send challenges indicating the MD5-sess algorithm.

o Clients SHOULD consider the protection space within a realm to be
   scoped to the authority portion of the URI, without regard to the
   contents of the path portion of the URI. Accordingly, relays

   SHOULD NOT send the "domain" parameter on the "WWW-Authenticate"
   header, and clients MUST ignore it if present.

   o Clients and relays MUST include a qop parameter in all "WWW-
      Authenticate" and "Authorization" headers. Note that the value of
      the qop parameter in a "WWW-Authenticate" header is quoted, but
      the value of the qop parameter in an "Authorization" header or
      "Authentication-Info" header is not quoted.

   o Clients MUST send cnonce and nonce-count parameters in all
      "Authorization" headers.

   o The request-URI to be used in calculating H(A2) is the rightmost
      URI in the To-Path header.

   o Relays MUST include rspauth, cnonce, nc, and qop parameters in a
      "Authentication-Info" header for all "200 OK" responses to an AUTH
      request.

   Note that the BNF in RFC 2617 has a number of errors. In particular,
   the value of the uri parameter MUST be in quotes; further, the
   parameters in the Authentication-Info header MUST be separated by
   commas. The BNF in this document is correct, as are the examples in
   RFC 2617 [1].

   The use of the nextnonce and nc parameters is supported as described
   in RFC 2617 [1], which provides guidance on how and when they should
   be used. As a slight modification to the guidance provided in RFC
   2617, implementors of relays should note that AUTH requests cannot be
   pipelined; consequently, there is no detrimental impact on throughput
   when relays use the nextnonce mechanism.

   See Section 5.1 for further information on the procedures for client
   authentication.

9.2. Using TLS

   TLS is used to authenticate relays to senders and to provide
   integrity and confidentiality for the headers being transported.
   MSRP clients and relays MUST implement TLS. Clients MUST send the
   TLS ClientExtendedHello extended hello information for server name
   indication as described in RFC 4366 [5]. A TLS cipher-suite of
   TLS_RSA_WITH_AES_128_CBC_SHA [6] MUST be supported (other cipher-
   suites MAY also be supported). A relay MUST act as a TLS server and
   present a certificate with its identity in the SubjectAltName using
   the choice type of dnsName. Relay-to-relay connections MUST use TLS
   with mutual authentication. Client-to-relay communications MUST use
   TLS for AUTH requests and responses.

The SubjectAltName in the certificate received from a relay MUST
match the hostname part of the URI, and the certificate MUST be valid
according to RFC 3280 [12], including having a date that is valid and
being signed by an acceptable certification authority. After
validating that such is the case, the device that initiated the TLS
connection can assume that it has connected to the correct relay.

This document does not define procedures for using mutual
authentication between an MSRP client and an MSRP relay.
Authentication of clients is handled using the AUTH method via the
procedures described in Section 5.1 and Section 6.3. Other
specifications may define the use of TLS mutual authentication for
the purpose of authenticating users associated with MSRP clients.
Unless operating under such other specifications, MSRP clients SHOULD
present an empty certificate list (if one is requested by the MSRP
relay), and MSRP relays SHOULD ignore any certificates presented by
the client.

    This behavior is defined specifically to allow forward-
    compatibility with specifications that define the use of TLS for
    client authentication.

Note: When relays are involved in a session, TCP without TLS is only
used when a user that does not use relays connects directly to the
relay of a user that is using relays. In this case, the client has
no way to authenticate the relay other than to use the URIs that form
a shared secret in the same way those URIs are used when no relays
are involved.

9.3. Threat Model

This section discusses the threat model and the broad mechanism that
needs to be in place to secure the protocol. The next section
describes the details of how the protocol mechanism meets the broad
requirements.

MSRP allows two peer-to-peer clients to exchange messages. Each peer
can select a set of relays to perform certain policy operations for
them. This combined set of relays is referred to as the route set.
A channel outside of MSRP always needs to exist, such as out-of-band
provisioning or an explicit rendezvous protocol such as SIP, that can
securely negotiate setting up the MSRP session and communicate the
route set to both clients. A client may trust a relay with certain
types of routing and policy decisions, but it might or might not
trust the relay with all the contents of the session. For example, a
relay being trusted to look for viruses would probably need to be
allowed to see all the contents of the session. A relay that helped
deal with traversal of the ISP's Network Address Translator (NAT)

would likely not be trusted with the contents of the session but
would be trusted to correctly forward messages.

Clients implicitly trust the relays through which they send and
receive messages to honor the routing indicated in those messages,
within the constraints of the MSRP protocol. Clients also need to
trust that the relays they use do not insert new messages on their
behalf or modify messages sent to or by the clients. It is worth
noting that some relays are in a position to cause a client to
misroute a message by maliciously modifying a Use-Path returned by a
relay further down the chain. However, this is not an additional
security threat because these same relays can also decide to misroute
a message in the first place. If the relay is trusted to route
messages, it is reasonable to trust it not to tamper with the Use-
Path header. If the relay cannot be trusted to route messages, then
it cannot be used.

Under certain circumstances, relays need to trust other relays not to
modify information between them and the client they represent. For
example, if a client is operating through Relay A to get to Relay B,
and Relay B is logging messages sent by the client, Relay B may be
required to authenticate that the messages they logged originate with
the client, and have not been modified or forged by Relay A. This
can be done by having the client sign the message.

Clients need to be able to authenticate that the relay they are
communicating with is the one they trust. Likewise, relays need to
be able to authenticate that the client is the one they are
authorized to forward information to. Clients need the option of
ensuring that information between the relay and the client is
integrity protected and confidential to elements other than the
relays and clients. To simplify the number of options, traffic
between relays is always integrity protected and encrypted regardless
of whether or not the client requests it. There is no way for the
clients to tell the relays what strength of cryptographic mechanisms
to use between relays other than to have the clients choose relays
that are administered to require an adequate level of security.

The system also needs to stop messages from being directed to relays
that are not supposed to see them. To keep the relays from being
used in Denial of Service (DoS) attacks, the relays never forward
messages unless they have a trust relationship with either the client
sending or the client receiving the message; further, they only
forward a message if it is coming from or going to the client with
which they have the trust relationship. If a relay has a trust
relationship with the client that is the destination of the message,
it should not send the message anywhere except to the client that is
the destination.

Some terminology used in this discussion: SClient is the client
sending a message and RClient is the client receiving a message.
SRelay is a relay the sender trusts and RRelay is a relay the
receiver trusts. The message will go from SClient to SRelay1 to
SRelay2 to RRelay2 to RRelay1 to RClient.

9.4. Security Mechanism

Confidentiality and privacy from elements not in the route set is
provided by using TLS on all the transports. Relays always use TLS.
A client can use unprotected TCP for peer-to-peer MSRP, but any time
a client communicates with its relay, it MUST use TLS.

The relays authenticate to the clients using TLS (but don't have to
do mutual TLS). Further, the use of the rspauth parameter in the
Authentication-Info header provides limited authentication of relays
to which the client is not directly connected. The clients
authenticate to the relays using HTTP Digest authentication. Relays
authenticate to each other using TLS mutual authentication.

By using Secure/Multipurpose Internet Mail Extensions (S/MIME) [3]
encryption, the clients can protect their actual message contents so
that the relays cannot see the contents. End-to-end signing is also
possible with S/MIME.

The complex part is making sure that relays cannot successfully be
instructed to send messages to a place where they should not. This
is done by having the client authenticate to the relay and having the
relay return a token. Messages that contain this token can be
relayed if they come from the client that got the token or if they
are being forwarded towards the client that got the token. The
tokens are the URIs that the relay places in the Use-Path header.
The tokens contain random material (defined in Section 6.3) so that
they are not guessable by attackers. The tokens need to be protected
so they are only ever seen by elements in the route set or other
elements that at least one of the parties trusts. If some third
party discovers the token that RRelay2 uses to forward messages to
RClient, then that third party can send as many messages as they want
to RRelay2 and it will forward them to RClient. The third party
cannot cause them to be forwarded anywhere except to RClient,
eliminating the open relay problems. SRelay1 will not forward the
message unless it contains a valid token.

When SClient goes to get a token from SRelay2, this request is
relayed through SRelay1. SRelay2 authenticates that it really is
SClient requesting the token, but it generates a token that is only
valid for forwarding messages to or from SRelay1. SRelay2 knows it
is connected to SRelay1 because of the mutual TLS.

The tokens are carried in the resource portion of the MSRP URIs. The
length of time the tokens are valid for is negotiated using the
Expire header in the AUTH request. Clients need to re-negotiate the
tokens using a new offer/answer [15] exchange (e.g., a SIP re-invite)
before the tokens expire.

Note that this scheme relies on relays as trusted nodes, acting on
behalf of the users authenticated to them. There is no security
mechanism to prevent relays on the path from inserting forged
messages, manipulating the contents of messages, sending messages in
a session to a party other than that specified by the sender, or from
copying them to a third party. However, the one-to-one binding
between session identifiers and sessions helps mitigate any damage
that can be caused by rogue relays by limiting the destinations to
which forged or modified messages can be sent to the two parties
involved in the session, and only for the duration of the session.
Additionally, the use of S/MIME encryption can be employed to limit
the utility of redirecting messages. Finally, clients can employ
S/MIME signatures to guarantee the authenticity of messages they
send, making it possible under some circumstances to detect relay
manipulation or the forging of messages.

Clients are not the only actors in the network who need to trust
relays to act in non-malicious ways. If a relay does not have a
direct TLS connection with the client on whose behalf it is acting
(i.e. There are one or more intervening relays), it is at the mercy
of any such intervening relays to accurately transmit the messages
sent to and from the client. If a stronger guarantee of the
authentic origin of a message is necessary (e.g. The relay is
performing logging of messages as part of a legal requirement), then
users of that relay can be instructed by their administrators to use
detached S/MIME signatures on all messages sent by their client. The
relay can enforce such a policy by returning a 415 response to any
SEND requests using a top-level MIME type other than "multipart/
signed". Such relays may choose to make policy decisions (such as
terminating sessions and/or suspending user authorization) if such
signatures fail to match the contents of the message.

10. IANA Considerations

10.1. New MSRP Method

This specification defines a new MSRP method, to be added to the
Methods sub-registry under the MSRP Parameters registry: AUTH. See
Section 5.1 for details on the AUTH method.

10.2. New MSRP Headers

This specification defines several new MSRP header fields, to be
added to the header-field sub-registry under the MSRP Parameters
registry:

o Expires
o Min-Expires
o Max-Expires
o Use-Path
o WWW-Authenticate
o Authorization
o Authentication-Info

10.3. New MSRP Response Codes

This specification defines one new MSRP status code, to be added to
the Status-Code sub-registry under the MSRP Parameters registry:

The 401 response indicates that an AUTH request contained no
credentials, an expired nonce value, or invalid credentials. The
response includes a "WWW-Authenticate" header containing a challenge
(among other fields); see Section 9.1 for further details. The
default response phrase for this response is "Unauthorized".

11. Example SDP with Multiple Hops

The following section shows an example SDP that could occur in a SIP
message to set up an MSRP session between Alice and Bob where Bob
uses a relay. Alice makes an offer with a path to Alice.

```
c=IN IP4 a.example.com
m=message 1234 TCP/MSRP *
a=accept-types: message/cpim text/plain text/html
a=path:msrp://a.example.com:1234/agic456;tcp
```

In this offer, Alice wishes to receive MSRP messages at
a.example.com. She wants to use TCP as the transport for the MSRP
session. She can accept message/cpim, text/plain, and text/html
message bodies in SEND requests. She does not need a relay to set up
the MSRP session.

To this offer, Bob's answer could look like:

```
c=IN IP4 bob.example.com
m=message 1234 TCP/TLS/MSRP *
a=accept-types: message/cpim text/plain
a=path:msrps://relay.example.com:9000/hjdhfha;tcp  \
        msrps://bob.example.com:1234/fuige;tcp
```

Here Bob wishes to receive the MSRP messages at bob.example.com. He
can accept only message/cpim and text/plain message bodies in SEND
requests and has rejected the text/html content offered by Alice. He
wishes to use a relay called relay.example.com for the MSRP session.

12. Acknowledgments

Many thanks to Avshalom Houri, Hisham Khartabil, Robert Sparks,
Miguel Garcia, Hans Persson, and Orit Levin, who provided detailed
proofreading and helpful text. Thanks to the following members of
the SIMPLE WG for spirited discussions on session mode: Chris
Boulton, Ben Campbell, Juhee Garg, Paul Kyzivat, Allison Mankin, Aki
Niemi, Pekka Pessi, Jon Peterson, Brian Rosen, Jonathan Rosenberg,
and Dean Willis.

13. References

13.1. Normative References

[1]   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.

[2]   Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS)
       Protocol Version 1.1", RFC 4346, April 2006.

[3]   Ramsdell, B., "Secure/Multipurpose Internet Mail Extensions
       (S/MIME) Version 3.1 Message Specification", RFC 3851, July
       2004.

[4]   Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for
       specifying the location of services (DNS SRV)", RFC 2782,
       February 2000.

   [5]   Blake-Wilson, S., Nystrom, M., Hopwood, D., Mikkelsen, J., and
         T. Wright, "Transport Layer Security (TLS) Extensions", RFC
         4366, April 2006.

   [6]   Chown, P., "Advanced Encryption Standard (AES) Ciphersuites for
         Transport Layer Security (TLS)", RFC 3268, June 2002.

   [7]   Handley, M., Jacobson, V., and C. Perkins, "SDP: Session
         Description Protocol", RFC 4566, July 2006.

   [8]   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.

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

   [10] Crocker, D. and P. Overell, "Augmented BNF for Syntax
         Specifications: ABNF", RFC 4234, October 2005.

   [11] Campbell, B., Ed., Mahy, R., Ed., and C. Jennings, Ed., "The
         Message Session Relay Protocol (MSRP)", RFC 4975, September
         2007.

   [12] Housley, R., Polk, W., Ford, W., and D. Solo, "Internet X.509
         Public Key Infrastructure Certificate and Certificate
         Revocation List (CRL) Profile", RFC 3280, April 2002.

13.2. Informative References

   [13] Freed, N. and N. Borenstein, "Multipurpose Internet Mail
         Extensions (MIME) Part One: Format of Internet Message Bodies",
         RFC 2045, November 1996.

   [14] Freed, N. and N. Borenstein, "Multipurpose Internet Mail
         Extensions (MIME) Part Two: Media Types", RFC 2046, November
         1996.

   [15] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with
         Session Description Protocol (SDP)", RFC 3264, June 2002.

Appendix A. Implementation Considerations

   This text is not necessary in order to implement MSRP in an
   interoperable way, but is still useful as an implementation
   discussion for the community. It is purely an implementation detail.

   Note: The idea has been proposed of having a relay return a base URI
   that the client can use to construct more URIs, but this allows third
   parties that have had a session with the client to know URIs that the
   relay will use for forwarding after the session with the third party
   has ended. Effectively, this reveals the secret URIs to third
   parties, which compromises the security of the solution, so this
   approach is not used.

   An alternative to this approach causes the relays to return a URI
   that is divided into an index portion and a secret portion. The
   client can encrypt its identifier and its own opaque data with the
   secret portion, and concatenate this with the index portion to create
   a plurality of valid URIs. When the relay receives one of these
   URIs, it could use the index to look up the appropriate secret,
   decrypt the client portion, and verify that it contains the client
   identifier. The relay can then forward the request. The client does
   not need to send an AUTH request for each URI it uses. This is an
   implementation detail that is out of the scope of this document.

   It is possible to implement forwarding requirements in a farm without
   the relay saving any state. One possible implementation that a relay
   might use is described in the rest of this section. When a relay
   starts up, it could pick a cryptographically random 128-bit password
   (K) and 128-bit initialization vector (IV). If the relay was
   actually a farm of servers with the same DNS name, all the machines
   in the farm would need to share the same K. When an AUTH request is
   received, the relay forms a string that contains the expiry time of
   the URI, an indication if the previous hop was mutual TLS
   authenticated or not, and if it was, the name of the previous hop,
   and if it was not, the identifier for the connection that received
   the AUTH request. This string would be padded by appending a byte
   with the value 0x80 then adding zero or more bytes with the value of
   0x00 until the string length is a multiple of 16 bytes long. A new
   random IV would be selected (it needs to change because it forms the
   salt) and the padded string would be encrypted using AES-CBC with a
   key of K. The IV and encrypted data and an SPI (security parameter
   index) that changes each time K changes would be base 64 encoded and
   form the resource portion of the request URI. The SPI allows the key
   to be changed and for the system to know which K should be used.
   Later when the relay receives this URI, it could decrypt it and check
   that the current time was before the expiry time and check that the
   message was coming from or going to the connection or location

specified in the URI. Integrity protection is not required because
it is extremely unlikely that random data that was decrypted would
result in a valid location that was the same as the one the message
was routing to or from. When implementing something like this,
implementors should be careful not to use a scheme like EBE that
would allows portions of encrypted tokens to be cut and pasted into
other URIs.

Authors' Addresses

Cullen Jennings
Cisco Systems, Inc.
170 West Tasman Dr.
MS: SJC-21/2
San Jose, CA 95134
USA

Phone: +1 408 421-9990
EMail: fluffy@cisco.com

Rohan Mahy
Plantronics
345 Encincal Street
Santa Cruz, CA 95060
USA

EMail: rohan@ekabal.com

Adam Roach
Estacado Systems
17210 Campbell Rd.
Suite 250
Dallas, TX 75252
USA

Phone: sip:adam@estacado.net
EMail: adam@estacado.net

Full Copyright Statement

Intellectual Property

    The IETF takes no position regarding the validity or scope of any
    Intellectual Property Rights or other rights that might be claimed to
    pertain to the implementation or use of the technology described in
    this document or the extent to which any license under such rights
    might or might not be available; nor does it represent that it has
    made any independent effort to identify any such rights. Information
    on the procedures with respect to rights in RFC documents can be
    found in BCP 78 and BCP 79.

    Copies of IPR disclosures made to the IETF Secretariat and any
    assurances of licenses to be made available, or the result of an
    attempt made to obtain a general license or permission for the use of
    such proprietary rights by implementers or users of this
    specification can be obtained from the IETF on-line IPR repository at
    http://www.ietf.org/ipr.

    The IETF invites any interested party to bring to its attention any
    copyrights, patents or patent applications, or other proprietary
    rights that may cover technology that may be required to implement
    this standard. Please address the information to the IETF at
    ietf-ipr@ietf.org.