Internet Engineering Task Force (IETF)
Request for Comments: 8122
Obsoletes: 4572
Category: Standards Track
ISSN: 2070-1721

J. Lennox
Vidyo
C. Holmberg
Ericsson

March 2017

Connection-Oriented Media Transport over the Transport Layer Security (TLS) Protocol in the Session Description Protocol (SDP)

Abstract

This document specifies how to establish secure connection-oriented media transport sessions over the Transport Layer Security (TLS) protocol using the Session Description Protocol (SDP). It defines the SDP protocol identifier, 'TCP/TLS'. It also defines the syntax and semantics for an SDP 'fingerprint' attribute that identifies the certificate that will be presented for the TLS session. This mechanism allows media transport over TLS connections to be established securely, so long as the integrity of session descriptions is assured.

This document obsoletes RFC 4572 by clarifying the usage of multiple fingerprints.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc8122.

Table of Contents

1. Introduction

The Session Description Protocol (SDP) [8] provides a general-purpose format for describing multimedia sessions in announcements or invitations. For many applications, it is desirable to establish, as part of a multimedia session, a media stream that uses a connection-oriented transport. RFC 4145, "TCP-Based Media Transport in the Session Description Protocol (SDP)" [7], specifies a general mechanism for describing and establishing such connection-oriented streams; however, the only transport protocol it directly supports is TCP. In many cases, session participants wish to provide confidentiality, data integrity, and authentication for their media sessions. Therefore, this document extends the TCP-Based Media specification to allow session descriptions to describe media sessions that use the Transport Layer Security (TLS) protocol [10].

The TLS protocol allows applications to communicate over a channel that provides confidentiality and data integrity. The TLS specification, however, does not specify how specific protocols establish and use this secure channel; particularly, TLS leaves the question of how to interpret and validate authentication certificates as an issue for the protocols that run over TLS. This document specifies such usage for the case of connection-oriented media transport.

Complicating this issue, endpoints exchanging media will often be unable to obtain authentication certificates signed by a well-known root certification authority (CA). Most certificate authorities charge for signed certificates, particularly host-based certificates; additionally, there is a substantial administrative overhead to obtaining signed certificates, as certification authorities must be able to confirm that they are issuing the signed certificates to the correct party. Furthermore, in many cases the endpoints' IP addresses and host names are dynamic, for example, they may be obtained from DHCP. It is impractical to obtain a CA-signed certificate valid for the duration of a DHCP lease. For such hosts, self-signed certificates are usually the only option. This specification defines a mechanism that allows self-signed certificates to be used securely, provided that the integrity of the SDP description is assured. It allows for endpoints to include a secure hash of their certificate, known as the "certificate fingerprint", within the session description. Provided that the fingerprint of the offered certificate matches the one in the session description, end hosts can trust even self-signed certificates.

The rest of this document is laid out as follows. An overview of the problem and threat model is given in Section 3. Section 4 gives the basic mechanism for establishing TLS-based connected-oriented media in SDP. Section 5 describes the SDP fingerprint attribute, which, assuming that the integrity of the SDP content is assured, allows the secure use of self-signed certificates. Section 6 describes which X.509 certificates are presented and how they are used in TLS. Section 7 discusses additional security considerations.

1.1. Changes from RFC 4572

This document obsoletes RFC 4572 [20] but remains backwards compatible with older implementations. The changes from RFC 4572 [20] are as follows:

- clarifies that multiple 'fingerprint' attributes can be used to carry fingerprints (calculated using different hash functions) associated with a given certificate and to carry fingerprints associated with multiple certificates.
- clarifies the fingerprint matching procedure when multiple fingerprints are provided.
- updates the preferred hash function with a stronger cipher suite and removes the requirement to use the same hash function for calculating a certificate fingerprint and certificate signature.

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 [3].

3. Overview

This section discusses the threat model that motivates TLS transport for connection-oriented media streams. It also discusses, in more detail, the need for end systems to use self-signed certificates.

3.1. SDP Operational Modes

There are two principal operational modes for multimedia sessions: advertised and offer-answer. Advertised sessions are the simpler mode. In this mode, a server publishes, in some manner, an SDP session description of a multimedia session it is making available. The classic example of this mode of operation is the Session Announcement Protocol (SAP) [15], in which SDP session descriptions are periodically transmitted to a well-known multicast group.

Traditionally, these descriptions involve multicast conferences, but unicast sessions are also possible. (Obviously, connection-oriented media cannot use multicast.) Recipients of a session description connect to the addresses published in the session description. These recipients may not have been previously known to the advertiser of the session description.

Alternatively, SDP conferences can operate in offer-answer mode [4]. This mode allows two participants in a multimedia session to negotiate the multimedia session between them. In this model, one participant offers the other a description of the desired session from its perspective, and the other participant answers with the desired session from its own perspective. In this mode, each of the participants in the session has knowledge of the other one. This is the mode of operation used by the Session Initiation Protocol (SIP) [17].

3.2. Threat Model

Participants in multimedia conferences often wish to guarantee confidentiality, data integrity, and authentication for their media sessions. This section describes various types of attackers and the ways they attempt to violate these guarantees. It then describes how the TLS protocol can be used to thwart the attackers.

The simplest type of attacker is one who listens passively to the traffic associated with a multimedia session. This attacker might, for example, be on the same local-area or wireless network as one of the participants in a conference. This sort of attacker does not threaten a connection's data integrity or authentication, and almost any operational mode of TLS can provide media-stream confidentiality.

More sophisticated is an attacker who can send his own data traffic over the network, but who cannot modify or redirect valid traffic. In SDP's 'advertised' operational mode, this can barely be considered an attack; media sessions are expected to be initiated from anywhere on the network. In SDP's offer-answer mode, however, this type of attack is more serious. An attacker could initiate a connection to one or both of the endpoints of a session, thus impersonating an endpoint or acting as a man in the middle to listen in on their communications. To thwart these attacks, TLS uses endpoint certificates. So long as the certificates' private keys have not been compromised, the endpoints have an externally trusted mechanism (most commonly, a mutually trusted certification authority) to validate certificates. Because the endpoints know what certificate identity to expect, endpoints can be certain that such an attack has not taken place.

Finally, the most serious type of attacker is one who can modify or redirect session descriptions: for example, a compromised or malicious SIP proxy server. Neither TLS itself nor any mechanisms that use it can protect an SDP session against such an attacker. Instead, the SDP description itself must be secured through some mechanism; SIP, for example, defines how S/MIME [22] can be used to secure session descriptions.

3.3. The Need for Self-Signed Certificates

SDP session descriptions are created by any endpoint that needs to participate in a multimedia session. In many cases, such as SIP phones, such endpoints have dynamically configured IP addresses and host names and must be deployed with nearly zero configuration. For such an endpoint, it is, for practical purposes, impossible to obtain a certificate signed by a well-known certification authority.

If two endpoints have no prior relationship, self-signed certificates cannot generally be trusted, as there is no guarantee that an attacker is not launching a man-in-the-middle attack. Fortunately, however, if the integrity of SDP session descriptions can be assured, it is possible to consider those SDP descriptions themselves as a prior relationship: certificates can be securely described in the session description itself. This is done by providing a secure hash of a certificate, or "certificate fingerprint", as an SDP attribute; this mechanism is described in Section 5.

3.4. Example SDP Description for TLS Connection

Figure 1 illustrates an SDP offer that signals the availability of a T.38 fax session over TLS. For the purpose of brevity, the main portion of the session description is omitted in the example, showing only the ’m’ line and its attributes. (This example is the same as the first one in RFC 4145 [7], except for the proto parameter and the fingerprint attribute.) See the subsequent sections for explanations of the example’s TLS-specific attributes.

Note: due to RFC formatting conventions, this document splits SDP across lines whose content would exceed 72 characters. A backslash character marks where this line folding has taken place. This backslash and its trailing CRLF and whitespace would not appear in actual SDP content.

```
m=image 54111 TCP/TLS t38
c=IN IP4 192.0.2.2
a=setup:passive
a=connection:new
a=fingerprint:SHA-256 \
   12:DF:3E:5D:49:6B:19:E5:7C:AB:4A:AD:B9:B1:3F:82:18:3B:54:02:12:DF: \
   3E:5D:49:6B:19:E5:7C:AB:4A:AD
a=fingerprint:SHA-1 \
   4A:AD:B9:B1:3F:82:18:3B:54:02:12:DF:3E:5D:49:6B:19:E5:7C:AB
```

Figure 1: Example SDP Description Offering a TLS Media Stream

4. Protocol Identifiers

The 'm' line in SDP specifies, among other items, the transport protocol to be used for the media in the session. See the "Media Descriptions" section of SDP [8] for a discussion on transport protocol identifiers.

This specification defines the protocol identifier, 'TCP/TLS', which indicates that the media described will use the Transport Layer Security protocol [10] over TCP. (Using TLS over other transport protocols is not discussed in this document.) The 'TCP/TLS' protocol identifier describes only the transport protocol, not the upper-layer protocol. An 'm' line that specifies 'TCP/TLS' MUST further qualify the protocol using an fmt identifier to indicate the application being run over TLS.

Media sessions described with this identifier follow the procedures defined in RFC 4145 [7]. They also use the SDP attributes defined in that specification, 'setup' and 'connection'.

5. Fingerprint Attribute

Parties to a TLS session indicate their identities by presenting authentication certificates as part of the TLS handshake procedure. Authentication certificates are X.509 [2] certificates, as profiled by RFCs 3279 [5], 5280 [11], and 4055 [6].

In order to associate media streams with connections and to prevent unauthorized barge-in attacks on the media streams, endpoints MUST provide a certificate fingerprint. If the X.509 certificate presented for the TLS connection matches the fingerprint presented in the SDP, the endpoint can be confident that the author of the SDP is indeed the initiator of the connection.

A certificate fingerprint is a secure one-way hash of the Distinguished Encoding Rules (DER) form of the certificate. (Certificate fingerprints are widely supported by tools that manipulate X.509 certificates; for instance, the command "openssl x509 -fingerprint" causes the command-line tool of the openssl package to print a certificate fingerprint, and the certificate managers for Mozilla and Internet Explorer display them when viewing the details of a certificate.)

A fingerprint is represented in SDP as an attribute (an 'a' line). It consists of the name of the hash function used, followed by the hash value itself. The hash value is represented as a sequence of uppercase hexadecimal bytes, separated by colons. The number of bytes is defined by the hash function. (This is the syntax used by openssl and by the browsers' certificate managers. It is different from the syntax used to represent hash values in, for example, HTTP digest authentication [24], which uses unseparated lowercase hexadecimal bytes. Consistency with other applications of fingerprints was considered more important.)

The formal syntax of the fingerprint attribute is given in Augmented Backus-Naur Form [9] in Figure 2. This syntax extends the BNF syntax of SDP [8].

```
attribute              =/ fingerprint-attribute

fingerprint-attribute  =  "fingerprint" ":" hash-func SP fingerprint

hash-func              =  "sha-1" / "sha-224" / "sha-256" /
                          "sha-384" / "sha-512" /
                          "md5" / "md2" / token
                          ; Additional hash functions can only come
                          ; from updates to RFC 3279

fingerprint            =  2UHEX *(":" 2UHEX)
                          ; Each byte in upper-case hex, separated
                          ; by colons.

UHEX                   =  DIGIT / %x41-46 ; A-F uppercase
```

Figure 2: Augmented Backus-Naur Syntax for the Fingerprint Attribute

Following RFC 3279 [5] as updated by RFC 4055 [6], the defined hash functions are 'SHA-1' [1] [16], 'SHA-224' [1], 'SHA-256' [1], 'SHA-384' [1], 'SHA-512' [1], 'MD5' [13], and 'MD2' [23], with 'SHA-256' preferred. A new IANA registry, named "Hash Function Textual Names",

specified in Section 8, allows for the addition of future tokens, but they may only be added if they are included in RFCs that update or obsolete RFC 3279 [5].

Implementations compliant with this specification MUST NOT use the MD2 and MD5 hash functions to calculate fingerprints or to verify received fingerprints that have been calculated using them.

Note: The MD2 and MD5 hash functions are listed in this specification so that implementations can recognize them. Implementations that log unused hash functions might log occurrences of these algorithms differently to unknown hash algorithms.

The fingerprint attribute may be either a session-level or a media-level SDP attribute. If it is a session-level attribute, it applies to all TLS sessions for which no media-level fingerprint attribute is defined.

5.1. Multiple Fingerprints

Multiple SDP fingerprint attributes can be associated with an 'm' line. This can occur if multiple fingerprints have been calculated for a certificate using different hash functions. It can also occur if one or more fingerprints associated with multiple certificates have been calculated. This might be needed if multiple certificates will be used for media associated with an 'm' line (e.g., if separate certificates are used for RTP and the RTP Control Protocol (RTCP)) or where it is not known which certificate will be used when the fingerprints are exchanged. In such cases, one or more fingerprints MUST be calculated for each possible certificate.

An endpoint MUST, as a minimum, calculate a fingerprint using both the 'SHA-256' hash function algorithm and the hash function used to generate the signature on the certificate for each possible certificate. Including the hash from the signature algorithm ensures interoperability with strict implementations of RFC 4572 [20]. Either of these fingerprints MAY be omitted if the endpoint includes a hash with a stronger hash algorithm that it knows that the peer supports, if it is known that the peer does not support the hash algorithm, or if local policy mandates use of stronger algorithms.

If fingerprints associated with multiple certificates are calculated, the same set of hash functions MUST be used to calculate fingerprints for each certificate associated with the 'm' line.

An endpoint MUST select the set of fingerprints that use its most preferred hash function (out of those offered by the peer) and verify that each certificate used matches one fingerprint out of that set.

If a certificate does not match any such fingerprint, the endpoint MUST NOT establish the TLS connection.

Note: The SDP fingerprint attribute does not contain a reference to a specific certificate. Endpoints need to compare the fingerprint with a certificate hash in order to look for a match.

6. Endpoint Identification

6.1. Certificate Choice

An X.509 certificate binds an identity and a public key. If SDP describing a TLS session is transmitted over a mechanism that provides integrity protection, a certificate asserting any syntactically valid identity MAY be used. For example, an SDP description sent over HTTP/TLS [14] or secured by S/MIME [22] MAY assert any identity in the certificate securing the media connection.

Security protocols that provide only hop-by-hop integrity protection (e.g., the SIPS scheme [17], SIP over TLS) are considered sufficiently secure to allow the mode in which any valid identity is accepted. However, see Section 7 for a discussion of some security implications of this fact.

In situations where the SDP is not integrity-protected, the certificate provided for a TLS connection MUST certify an appropriate identity for the connection. In these scenarios, the certificate presented by an endpoint MUST certify either the SDP connection address or the identity of the creator of the SDP message, as follows:

- If the connection address for the media description is specified as an IP address, the endpoint MAY use a certificate with an iPAddress subjectAltName that exactly matches the IP in the connection-address in the session description's 'c' line. Similarly, if the connection address for the media description is specified as a fully qualified domain name, the endpoint MAY use a certificate with a dNSName subjectAltName matching the specified 'c' line connection-address exactly. (Wildcard patterns MUST NOT be used.)

- Alternately, if the SDP session description of the session was transmitted over a protocol (such as SIP [17]) for which the identities of session participants are defined by Uniform Resource Identifiers (URIs), the endpoint MAY use a certificate with a uniformResourceIdentifier subjectAltName corresponding to the identity of the endpoint that generated the SDP. The details of

what URIs are valid are dependent on the transmitting protocol. (For more details on the validity of URIs, see Section 7.

Identity matching is performed using the matching rules specified by RFC 5280 [11]. If more than one identity of a given type is present in the certificate (e.g., more than one dNSName name), a match in any one of the set is considered acceptable. To support the use of certificate caches, as described in Section 7, endpoints SHOULD consistently provide the same certificate for each identity they support.

6.2. Certificate Presentation

In all cases, an endpoint acting as the TLS server (i.e., one taking the 'setup:passive' role, in the terminology of connection-oriented media) MUST present a certificate during TLS initiation, following the rules presented in Section 6.1. If the certificate does not match the original fingerprint, the client endpoint MUST terminate the media connection with a bad_certificate error.

If the SDP offer/answer model [4] is being used, the client (the endpoint with the 'setup:active' role) MUST also present a certificate following the rules of Section 6.1. The server MUST request a certificate; if the client does not provide one, or if the certificate does not match a provided fingerprint, the server endpoint MUST terminate the media connection with a bad_certificate error.

Note that when the offer/answer model is being used, it is possible for a media connection to outrace the answer back to the offerer. Thus, if the offerer has offered a 'setup:passive' or 'setup:actpass' role, it MUST (as specified in RFC 4145 [7]) begin listening for an incoming connection as soon as it sends its offer. However, it MUST NOT assume that the data transmitted over the TLS connection is valid until it has received a matching fingerprint in an SDP answer. If the fingerprint, once it arrives, does not match the client's certificate, the server endpoint MUST terminate the media connection with a bad_certificate error, as stated in the previous paragraph.

If offer/answer is not being used (e.g., if the SDP was sent over the Session Announcement Protocol [15]), there is no secure channel available for clients to communicate certificate fingerprints to servers. In this case, servers MAY request client certificates, which SHOULD be signed by a well-known certification authority, or MAY allow clients to connect without a certificate.

7. Security Considerations

This entire document concerns itself with security. The problem to be solved is addressed in Section 1, and a high-level overview is presented in Section 3. See the SDP specification [8] for security considerations applicable to SDP in general.

Offering a TCP/TLS connection in SDP (or agreeing to one in the SDP offer/answer mode) does not create an obligation for an endpoint to accept any TLS connection with the given fingerprint. Instead, the endpoint must engage in the standard TLS negotiation procedure to ensure that the TLS stream cipher and MAC algorithm chosen meet the security needs of the higher-level application. (For example, an offered stream cipher of TLS_NULL_WITH_NULL_NULL SHOULD be rejected in almost every application scenario.)

Like all SDP messages, SDP messages describing TLS streams are conveyed in an encapsulating application protocol (e.g., SIP, Media Gateway Control Protocol (MGCP), etc.). It is the responsibility of the encapsulating protocol to ensure the integrity of the SDP security descriptions. Therefore, the application protocol SHOULD either invoke its own security mechanisms (e.g., secure multiparts) or, alternatively, utilize a lower-layer security service (e.g., TLS or IPsec). This security service SHOULD provide strong message authentication as well as effective replay protection.

However, such integrity protection is not always possible. For these cases, end systems SHOULD maintain a cache of certificates that other parties have previously presented using this mechanism. If possible, users SHOULD be notified when an unsecured certificate associated with a previously unknown end system is presented and SHOULD be strongly warned if a different unsecured certificate is presented by a party with which they have communicated in the past. In this way, even in the absence of integrity protection for SDP, the security of this document’s mechanism is equivalent to that of the Secure Shell (SSH) protocol [18], which is vulnerable to man-in-the-middle attacks when two parties first communicate but can detect ones that occur subsequently. (Note that a precise definition of the "other party" depends on the application protocol carrying the SDP message.) Users SHOULD NOT, however, in any circumstances be notified about certificates described in the SDP descriptions sent over an integrity-protected channel.

To aid interoperability and deployment, security protocols that provide only hop-by-hop integrity protection (e.g., the SIPS scheme [17], SIP over TLS) are considered sufficiently secure to allow the mode in which any syntactically valid identity is accepted in a certificate. This decision was made because SIPS is currently the

integrity mechanism most likely to be used in deployed networks in the short to medium term. However, in this mode, SDP integrity is vulnerable to attacks by compromised or malicious middleboxes, e.g., SIP proxy servers. End systems MAY warn users about SDP sessions that are secured in only a hop-by-hop manner, and definitions of media formats running over TCP/TLS MAY specify that only end-to-end integrity mechanisms be used.

Depending on how SDP messages are transmitted, it is not always possible to determine whether or not a subjectAltName presented in a remote certificate is expected for the remote party. In particular, given call forwarding, third-party call control, or session descriptions generated by endpoints controlled by the Gateway Control Protocol [21], it is not always possible in SIP to determine what entity ought to have generated a remote SDP response. In general, when not using authenticity and integrity protection of the SDP descriptions, a certificate transmitted over SIP SHOULD assert the endpoint's SIP Address of Record as a uniformResourceIndicator subjectAltName. When an endpoint receives a certificate over SIP asserting an identity (including an iPAddress or dNSName identity) other than the one to which it placed or received the call, it SHOULD alert the user and ask for confirmation. This applies whether certificates are self-signed or signed by certification authorities; a certificate for "sip:bob@example.com" may be legitimately signed by a certification authority, but it may still not be acceptable for a call to "sip:alice@example.com". (This issue is not one specific to this specification; the same consideration applies for S/MIME-signed SDP carried over SIP.)

This document does not define a mechanism for securely transporting RTP and RTCP packets over a connection-oriented channel. Please see RFC 7850 [19] for more details.

TLS is not always the most appropriate choice for secure connection-oriented media; in some cases, a higher- or lower-level security protocol may be appropriate.

This document improves security from RFC 4572 [20]. It updates the preferred hash function from SHA-1 to SHA-256 and deprecates the usage of the MD2 and MD5 hash functions.

By clarifying the usage and handling of multiple fingerprints, the document also enables hash agility and incremental deployment of newer and more secure hash functions.

8. IANA Considerations

IANA has updated the registrations defined in RFC 4572 [20] to refer to this specification.

This document defines an SDP proto value: 'TCP/TLS'. Its format is defined in Section 4. This proto value has been registered by IANA under the "proto" registry within the "Session Description Protocol (SDP) Parameters" registry.

This document defines an SDP session and media-level attribute: 'fingerprint'. Its format is defined in Section 5. This attribute has been registered by IANA under the "att-field (both session and media level)" registry within the "Session Description Protocol (SDP) Parameters" registry.

The SDP specification [8] states that specifications defining new proto values, like the 'TCP/TLS' proto value defined in this one, must define the rules by which their media format (fmt) namespace is managed. For the TCP/TLS protocol, new formats SHOULD have an associated MIME registration. Use of an existing MIME subtype for the format is encouraged. If no MIME subtype exists, it is RECOMMENDED that a suitable one be registered through the IETF process [12] by production of, or reference to, a Standards Track RFC that defines the transport protocol for the format.

IANA has updated the "Hash Function Textual Names" registry (which was originally created in [20]) to refer to this document.

The names of hash functions used for certificate fingerprints are registered by the IANA. Hash functions MUST be defined by Standards Track RFCs that update or obsolete RFC 3279 [5].

When registering a new hash function textual name, the following information MUST be provided:

- The textual name of the hash function.
- The Object Identifier (OID) of the hash function as used in X.509 certificates.
- A reference to the Standards Track RFC that updates or obsoletes RFC 3279 [5] and defines the use of the hash function in X.509 certificates.

Table 1 contains the initial values of this registry.

Hash Function Name	OID	Reference
"md2"	1.2.840.113549.2.2	RFC 3279
"md5"	1.2.840.113549.2.5	RFC 3279
"sha-1"	1.3.14.3.2.26	RFC 3279
"sha-224"	2.16.840.1.101.3.4.2.4	RFC 4055
"sha-256"	2.16.840.1.101.3.4.2.1	RFC 4055
"sha-384"	2.16.840.1.101.3.4.2.2	RFC 4055
"sha-512"	2.16.840.1.101.3.4.2.3	RFC 4055

Table 1: IANA Hash Function Textual Name Registry

9. References

9.1. Normative References

[1] National Institute of Standards and Technology, "Secure Hash Standard (SHS)", FIPS PUB 180-4, DOI 10.6028/NIST.FIPS.180-4, August 2015, <http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf>.

[2] International Organization for Standardization, "Information technology -- Open Systems Interconnection -- The Directory -- Part 8: Public-key and attribute certificate frameworks", ISO/IEC 9594-8:2014, March 2014, <https://www.iso.org/standard/64854.html>.

[3] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[4] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, DOI 10.17487/RFC3264, June 2002, <http://www.rfc-editor.org/info/rfc3264>.

[5] Bassham, L., Polk, W., and R. Housley, "Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3279, DOI 10.17487/RFC3279, April 2002, <http://www.rfc-editor.org/info/rfc3279>.

[6] Schaad, J., Kaliski, B., and R. Housley, "Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 4055, DOI 10.17487/RFC4055, June 2005, <http://www.rfc-editor.org/info/rfc4055>.

[7] Yon, D. and G. Camarillo, "TCP-Based Media Transport in the Session Description Protocol (SDP)", RFC 4145, DOI 10.17487/RFC4145, September 2005, <http://www.rfc-editor.org/info/rfc4145>.

[8] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, DOI 10.17487/RFC4566, July 2006, <http://www.rfc-editor.org/info/rfc4566>.

[9] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <http://www.rfc-editor.org/info/rfc5234>.

[10] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <http://www.rfc-editor.org/info/rfc5246>.

[11] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <http://www.rfc-editor.org/info/rfc5280>.

[12] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <http://www.rfc-editor.org/info/rfc6838>.

9.2. Informative References

[13] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, DOI 10.17487/RFC1321, April 1992, <http://www.rfc-editor.org/info/rfc1321>.

[14] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <http://www.rfc-editor.org/info/rfc2818>.

[15] Handley, M., Perkins, C., and E. Whelan, "Session Announcement Protocol", RFC 2974, DOI 10.17487/RFC2974, October 2000, <http://www.rfc-editor.org/info/rfc2974>.

[16] Eastlake 3rd, D. and P. Jones, "US Secure Hash Algorithm 1 (SHA1)", RFC 3174, DOI 10.17487/RFC3174, September 2001, <http://www.rfc-editor.org/info/rfc3174>.

[17] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <http://www.rfc-editor.org/info/rfc3261>.

[18] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Protocol Architecture", RFC 4251, DOI 10.17487/RFC4251, January 2006, <http://www.rfc-editor.org/info/rfc4251>.

[19] Lazzaro, J., "Framing Real-time Transport Protocol (RTP) and RTP Control Protocol (RTCP) Packets over Connection-Oriented Transport", RFC 4571, DOI 10.17487/RFC4571, July 2006, <http://www.rfc-editor.org/info/rfc4571>.

[20] Lennox, J., "Connection-Oriented Media Transport over the Transport Layer Security (TLS) Protocol in the Session Description Protocol (SDP)", RFC 4572, DOI 10.17487/RFC4572, July 2006, <http://www.rfc-editor.org/info/rfc4572>.

[21] Taylor, T., "Reclassification of RFC 3525 to Historic", RFC 5125, DOI 10.17487/RFC5125, February 2008, <http://www.rfc-editor.org/info/rfc5125>.

[22] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification", RFC 5751, DOI 10.17487/RFC5751, January 2010, <http://www.rfc-editor.org/info/rfc5751>.

[23] Turner, S. and L. Chen, "MD2 to Historic Status", RFC 6149, DOI 10.17487/RFC6149, March 2011, <http://www.rfc-editor.org/info/rfc6149>.

[24] Shekh-Yusef, R., Ed., Ahrens, D., and S. Bremer, "HTTP Digest Access Authentication", RFC 7616, DOI 10.17487/RFC7616, September 2015, <http://www.rfc-editor.org/info/rfc7616>.

Acknowledgments

This document included significant contributions by Cullen Jennings, Paul Kyzivat, Roman Shpount, and Martin Thomson. Elwyn Davies performed the Gen-ART review of the document.

Authors' Addresses

Jonathan Lennox
Vidyo

Email: jonathan@vidyo.com

Christer Holmberg
Ericsson

Email: christer.holmberg@ericsson.com