Network Working Group
Internet-Draft
Updates: 8122 (if approved)
Intended status: Standards Track
Expires: July 8, 2019

M. Thomson
E. Rescorla
Mozilla

January 04, 2019

Unknown Key Share Attacks on uses of TLS with the Session Description Protocol (SDP)
draft-ietf-mmusic-sdp-uks-03

Abstract

This document describes unknown key-share attacks on the use of Datagram Transport Layer Security for the Secure Real-Time Transport Protocol (DTLS-SRTP). Similar attacks are described on the use of DTLS-SRTP with the identity bindings used in Web Real-Time Communications (WebRTC) and SIP identity. These attacks are difficult to mount, but they cause a victim to be mislead about the identity of a communicating peer. Simple mitigation techniques are defined for each.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 8, 2019.

Copyright Notice

Table of Contents

1. Introduction

The use of Transport Layer Security (TLS) [TLS13] with the Session Description Protocol (SDP) [SDP] is defined in [FINGERPRINT]. Further use with Datagram Transport Layer Security (DTLS) [DTLS] and the Secure Real-time Transport Protocol (SRTP) [SRTP] is defined as DTLS-SRTP [DTLS-SRTP].

In these specifications, key agreement is performed using TLS or DTLS, with authentication being tied back to the session description (or SDP) through the use of certificate fingerprints. Communication peers check that a hash, or fingerprint, provided in the SDP matches the certificate that is used in the TLS or DTLS handshake.

WebRTC identity (see Section 7 of [WEBRTC-SEC]) and SIP identity [SIP-ID] both provide a mechanism that binds an external identity to the certificate fingerprints from a session description. However, this binding is not integrity-protected and therefore vulnerable to an identity misbinding attack - or unknown key-share (UKS) attack -

where the attacker binds their identity to the fingerprint of another entity. A successful attack leads to the creation of sessions where peers are confused about the identify of the participants.

This document describes a TLS extension that can be used in combination with these identity bindings to prevent this attack.

A similar attack is possible with the use of certificate fingerprints alone. Though attacks in this setting are likely infeasible in existing deployments due to the narrow conditions necessary (see Section 2.1), this document also describes mitigations for this attack.

The mechanisms defined in this document are intended to strengthen the protocol by preventing the use of unknown key shares in combination with other protocol or implementation vulnerabilities.

This document assumes that signaling is integrity protected. However, as Section 7 of [FINGERPRINT] explains, many deployments that use SDP do not guarantee integrity of session signaling and so are vulnerable to other attacks. [FINGERPRINT] offers key continuity mechanisms as a potential means of reducing exposure to attack in the absence of integrity protection. Section 2.2 provides some analysis of the effect of key continuity in relation to the described attacks.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Unknown Key-Share Attack

In an unknown key-share attack [UKS], a malicious participant in a protocol claims to control a key that is in reality controlled by some other actor. This arises when the identity associated with a key is not properly bound to the key.

An endpoint that can acquire the certificate fingerprint of another entity can advertise that fingerprint as their own in SDP. An attacker can use a copy of that fingerprint to cause a victim to communicate with another unaware victim, even though it believes that it is communicating with the attacker.

When the identity of communicating peers is established by higher-layer signaling constructs, such as those in SIP identity [SIP-ID] or WebRTC [WEBRTC-SEC], this allows an attacker to bind their own identity to a session with any other entity.

The attacker obtains an identity assertion for an identity it controls, but binds that to the fingerprint of one peer. The attacker is then able to cause a TLS connection to be established where two endpoints communicate. The victim that has its fingerprint copied by the attack correctly believes that it is communicating with the other victim; however, the other victim incorrectly believes that it is communicating with the attacker.

A similar attack can be mounted without any communications established based on the SDP "fingerprint" attribute [FINGERPRINT].

This attack is an aspect of SDP-based protocols that the technique known as third-party call control (3PCC) [RFC3725] relies on. 3PCC exploits the potential for the identity of a signaling peer to be different than the media peer, allowing the media peer to be selected by the signaling peer. Section 2.3 describes the consequences of the mitigations described here for systems that use 3PCC.

2.1. Limits on Attack Feasibility

The use of TLS with SDP depends on the integrity of session signaling. Assuming signaling integrity limits the capabilities of an attacker in several ways. In particular:

1. An attacker can only modify the parts of the session signaling for a session that they are part of, which is limited to their own offers and answers.

2. No entity will complete communications with a peer unless they are willing to participate in a session with that peer.

The combination of these two constraints make the spectrum of possible attacks quite limited. An attacker is only able to switch its own certificate fingerprint for a valid certificate that is acceptable to its peer. Attacks therefore rely on joining two separate sessions into a single session.

However, the second condition might not be necessary when using an identity binding such as those defined in [WEBRTC] or [SIP-ID]. When using an identity binding, the threat model assumes the possibility of attack by an entity with access to the signaling channel. Removing this constraint makes attacks considerably more feasible.

2.2. Interactions with Key Continuity

Systems that use key continuity might be able to detect an unknown key-share attack if a session with either the attacker or the geniune peer (i.e., the victim whose fingerprint was copied by an attacker)

was established in the past. Whether this is possible depends on how key continuity is implemented.

Implementations that maintain a single database of identities with an index on peer keys could discover that the identity saved for the peer key does not match the claimed identity. Such an implementation could notice the disparity between the actual keys (those copied from a victim) and the expected keys (those of the attacker).

In comparison, implementations that first match based on peer identity could treat an unknown key-share attack as though their peer had used a newly-configured device. The apparent addition of a new device could generate user-visible notices (e.g., "Mallory appears to have a new device"). However, such an event is not always considered alarming; some implementations might silently save a new key.

2.3. Third-Party Call Control

Third-party call control (3PCC) [RFC3725] is a technique where a signaling peer establishes a call that is terminated by a different entity. This attack is very similar to the 3PCC technique, except where the TLS peers are aware of the use of 3PCC.

For 3PCC to work with the proposed mechanisms, TLS peers need to be aware of the signaling so that they can correctly generate (and check) the extension. It is understood that this technique will prevent the use of 3PCC if peers are not able to access signaling.

3. Attack on Identity Bindings

The identity assertions used for WebRTC (Section 7 of [WEBRTC-SEC]) and the SIP PASSPoRT using in SIP identity ([SIP-ID], [PASSPoRT]) are bound to the certificate fingerprint of an endpoint. An attacker causes an identity binding to be created that binds an identity they control to the fingerprint of a victim.

An attacker can thereby cause a victim to believe that they are communicating with an attacker-controlled identity, when they are really talking to another entity of the attacker's choice. The attacker only needs to create an identity assertion that covers a certificate fingerprint of their choosing.

The problem might appear to be caused by the fact that the entity that certifies the identity binding is not required to verify that the entity requesting the binding controls the keys associated with the fingerprints. Both SIP and WebRTC identity providers are not required to perform this validation. This is not an issue because verifying control of the associated keys is not a necessary condition

for a secure protocol, nor would it be sufficient to prevent attack [SIGMA].

A simple solution to this problem is suggested by [SIGMA]. The identity of endpoints is included under a message authentication code (MAC) during the cryptographic handshake. Endpoints then validate that their peer has provided an identity that matches their expectations. In TLS, the Finished message provides a MAC over the entire handshake, so that including the identity in a TLS extension is sufficient to implement this solution.

Rather than include a complete identity binding - which could be sizeable - a collision- and pre-image-resistant hash of the binding is included in a TLS extension. Endpoints then need only validate that the extension contains a hash of the identity binding they received in signaling. If the identity binding is successfully validated, the identity of a peer is verified and bound to the session.

The same technique can be used to cause two victims to both believe they are talking to the attacker when they are talking to each other.

3.1. Example

In the example shown in Figure 1, it is assumed that the attacker also controls the signaling channel.

Mallory (the attacker) presents two victims, Norma and Patsy, with two separate sessions. In the first session, Patsy is presented with the option to communicate with Norma; a second session with Mallory is presented to Norma.

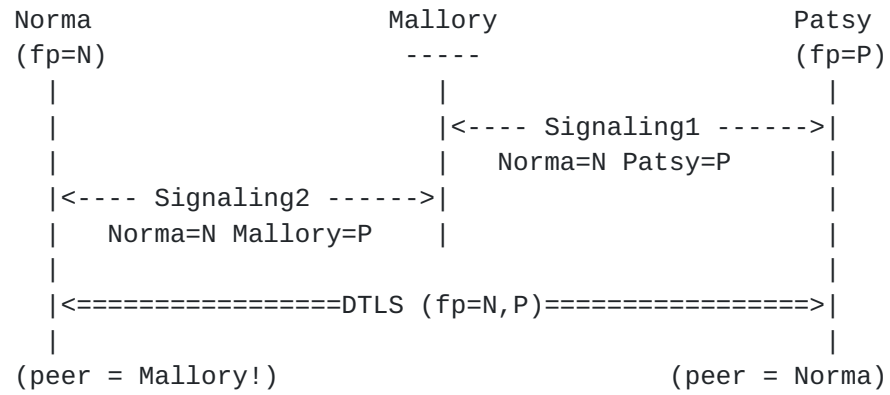

Figure 1: Example Attack on Identity Bindings

The attack requires that Mallory obtain an identity binding for their own identity with the fingerprints presented by Patsy (P). This false binding is then presented to Norma.

Patsy could be similarly duped, but in this example, a correct binding between Norma's identity and fingerprint (N) is faithfully presented by Mallory.

The resulting DTLS session is established directly between Norma and Patsy. Patsy correctly believes that they are communicating with Norma. However, Norma incorrectly believes they are talking to Mallory.

In order for this attack to work without compromising signaling integrity, it is likely that the attacker also needs to subvert the session as described in Section 4. Endpoints can use the "external_session_id" extension (see Section 4.3) in addition to this so that two calls between the same parties can't be altered by an attacker.

3.2. The external_id_hash TLS Extension

The "external_id_hash" TLS extension carries a hash of the identity assertion that communicating peers have exchanged.

The "extension_data" for the "external_id_hash" extension contains a "ExternalIdentityHash" struct, described below using the syntax defined in [TLS13]:

```
struct {
   opaque binding_hash<0..32>;
} ExternalIdentityHash;
```

A WebRTC identity assertion is provided as a JSON [JSON] object that is encoded into a JSON text. The resulting string is then encoded using UTF-8 [UTF8]. The content of the "external_id_hash" extension are produced by hashing the resulting octets with SHA-256 [SHA]. This produces the 32 octets of the "binding_hash" parameter, which is the sole contents of the extension.

The SDP "identity" attribute includes the base64 [BASE64] encoding of the same octets that were input to the hash. The "external_id_hash" extension is validated by performing base64 decoding on the value of the SDP "identity" attribute, hashing the resulting octets using SHA-256, and comparing the results with the content of the extension.

Where a PASSPoRT is used, the compact form of the PASSPoRT MUST be expanded into the full form. The base64 encoding used in the

Identity (or 'y') header field MUST be decoded then used as input to SHA-256. This produces the 32 octet "binding_hash" value used for creating or validating the extension.

Note: Should SHA-256 prove to be inadequate at some point in the future (see [AGILITY]), a new TLS extension can be defined that uses a different hash function.

Identity bindings in either form might be provided by only one peer. An endpoint that does not produce an identity binding MUST generate an empty "external_id_hash" extension in its ClientHello. This allows its peer to include a hash of its identity binding. An endpoint without an identity binding MUST include an empty "external_id_hash" extension in its ServerHello or EncryptedExtensions message, to indicate support for the extension.

A peer that receives an "external_id_hash" extension that does not match the value of the identity binding from its peer MUST immediately fail the TLS handshake with an error. This includes cases where the binding is absent, in which the extension MUST be present and empty.

An "external_id_hash" extension that is any length other than 0 or 32 is invalid and MUST cause the receiving endpoint to generate a fatal "decode_error" alert.

A peer that receives an identity binding, but does not receive an "external_id_hash" extension MAY choose to fail the connection, though it is expected that implementations written prior to the definition of the extensions in this document will not support both for some time.

In TLS 1.3, the "external_id_hash" extension MUST be sent in the EncryptedExtensions message.

4. Unknown Key-Share with Fingerprints

A similar attack can create a session where there is confusion about the communicating endpoints by substituting the fingerprint of a communicating endpoint.

An endpoint that is configured to reuse a certificate can be attacked if it is willing to initiate two calls at the same time, one of which is with an attacker. The attacker can arrange for the victim to incorrectly believe that is calling the attacker when it is in fact calling a second party. The second party correctly believes that it is talking to the victim.

As with the attack on identity bindings, this can be used to cause two victims to both believe they are talking to the attacker when they are talking to each other.

4.1. Example

In this example, two sessions are created with the same endpoint at the same time. One of those sessions is initiated with the attacker, the second session is created toward another honest endpoint. The attacker convinces the endpoint that their session has completed, and that the session with the other endpoint has succeeded.

In addition to the constraints described in Section 2.1, the attacker in this example also needs to the ability to view and drop packets between victims. That is, the attacker is on-path.

The attack shown in Figure 2 depends on a somewhat implausible set of conditions. It is intended to demonstrate what sort of attack is possible and what conditions are necessary to exploit this weakness in the protocol.

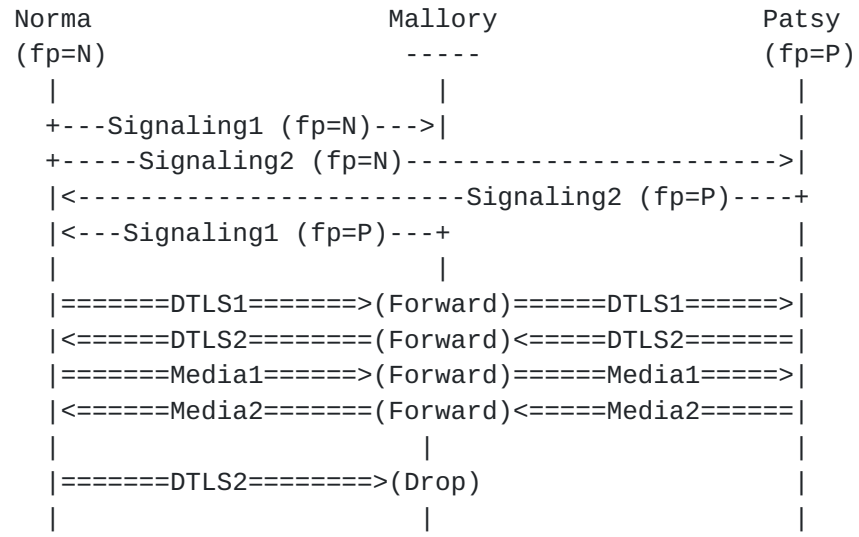

Figure 2: Example Attack Scenario using Fingerprints

In this scenario, there are two sessions initiated at the same time by Norma. Signaling is shown with single lines ('-'), DTLS and media with double lines ('=').

The first session is established with Mallory, who falsely uses Patsy's certificate fingerprint (denoted with 'fp=P'). A second session is initiated between Norma and Patsy. Signaling for both sessions is permitted to complete.

Once signaling is complete on the session that is ostensibly between Mallory and Norma is complete. Mallory begins forwarding DTLS and media packets sent to her by Norma to Patsy. These packets denoted 'DTLS1' because Norma associates these with the first signaling session ('signaling1').

Mallory also intercepts packets from Patsy and forwards those to Norma at the transport address that Norma associates with Mallory. These packets are denoted 'DTLS2' to indicate that Patsy associates these with the second signaling session ('signaling2'), however Norma will interpret these as being associated with the first signaling session ('signaling1').

The second signaling exchange - 'signaling2', between Norma and Patsy - is permitted to continue to the point where Patsy believes that it has succeeded. This ensures that Patsy believes that she is communicating with Norma. In the end, Norma believes that she is communicating with Mallory, when she is really communicating with Patsy.

Though Patsy needs to believe that the second signaling session has been successfully established, Mallory has no real interest in seeing that session complete. Mallory only needs to ensure that Patsy does not abandon the session prematurely. For this reason, it might be necessary to permit the signaling from Patsy to reach Norma to allow Patsy to receive a call completion signal, such as a SIP ACK. Once the second session completes, Mallory might cause DTLS packets sent by Norma to Patsy to be dropped, though these will likely be discarded by Patsy.

For the attacked session to be sustained beyond the point that Norma detects errors in the second session, Mallory also needs to block any signaling that Norma might send to Patsy asking for the call to be abandoned. Otherwise, Patsy might receive a notice that the call is failed and thereby abort the call.

This attack creates an asymmetry in the beliefs about the identity of peers. However, this attack is only possible if the victim (Norma) is willing to conduct two sessions nearly simultaneously, if the attacker (Mallory) is on the network path between the victims, and if the same certificate - and therefore SDP "fingerprint" attribute value - is used in both sessions.

Where ICE [ICE] is used, Mallory also needs to ensure that connectivity between Patsy and Norma succeed, either by forwarding checks or answering and generating the necessary messages.

4.2. Unique Session Identity Solution

An attack on DTLS-SRTP is possible because the identity of peers involved is not established prior to establishing the call. Endpoints use certificate fingerprints as a proxy for authentication, but as long as fingerprints are used in multiple calls, they are vulnerable to attack.

The solution to this problem is to assign a new identifier to communicating peers. Each endpoint assigns their peer a unique identifier during call signaling. The peer echoes that identifier in the TLS handshake, binding that identity into the session. Including this new identity in the TLS handshake means that it will be covered by the TLS Finished message, which is necessary to authenticate it (see [SIGMA]). Validating that peers use the correct identifier then means that the session is established between the correct two endpoints.

This solution relies on the unique identifier given to DTLS sessions using the SDP "tls-id" attribute [DTLS-SDP]. This field is already required to be unique. Thus, no two offers or answers from the same client will have the same value.

A new "external_session_id" extension is added to the TLS or DTLS handshake for connections that are established as part of the same call or real-time session. This carries the value of the "tls-id" attribute and provides integrity protection for its exchange as part of the TLS or DTLS handshake.

4.3. The external_session_id TLS Extension

The "external_session_id" TLS extension carries the unique identifier that an endpoint selects. When used with SDP, the value includes the "tls-id" attribute from the SDP that the endpoint generated when negotiating the session. This document only defines use of this extension for SDP; other methods of external session negotiation can use this extension to include a unique session identifier.

The "extension_data" for the "external_session_id" extension contains a ExternalSessionId struct, described below using the syntax defined in [TLS13]:

```
struct {
   opaque session_id<20..255>;
} ExternalSessionId;
```

For SDP, the "session_id" field of the extension includes the value of the "tls-id" SDP attribute as defined in [DTLS-SDP] (that is, the

"tls-id-value" ABNF production). The value of the "tls-id" attribute is encoded using ASCII [ASCII].

Where RTP and RTCP [RTP] are not multiplexed, it is possible that the two separate DTLS connections carrying RTP and RTCP can be switched. This is considered benign since these protocols are usually distinguishable. RTP/RTCP multiplexing is advised to address this problem.

The "external_session_id" extension is included in a ClientHello and either ServerHello (for TLS and DTLS versions less than 1.3) or EncryptedExtensions (for TLS 1.3). In TLS 1.3, the "external_session_id" extension MUST NOT be included in a ServerHello.

Endpoints MUST check that the "session_id" parameter in the extension that they receive includes the "tls-id" attribute value that they received in their peer's session description. Endpoints can perform string comparison by ASCII decoding the TLS extension value and comparing it to the SDP attribute value, or compare the encoded TLS extension octets with the encoded SDP attribute value. An endpoint that receives a "external_session_id" extension that is not identical to the value that it expects MUST abort the connection with a fatal "handshake_failure" alert.

An endpoint that is communicating with a peer that does not support this extension will receive a ClientHello, ServerHello or EncryptedExtensions that does not include this extension. An endpoint MAY choose to continue a session without this extension in order to interoperate with peers that do not implement this specification.

In TLS 1.3, the "external_session_id" extension MUST be sent in the EncryptedExtensions message.

5. Consequences of Session Concatenation

Use of session identifiers does not prevent an attacker from establishing two concurrent sessions with different peers and forwarding signaling from those peers to each other. Concatenating two signaling sessions creates a situation where both peers believe that they are talking to the attacker when they are talking to each other.

This kind of attack is prevented by systems that enable peer authentication such as WebRTC identity [WEBRTC-SEC] or SIP identity [SIP-ID]. However, session concatention remains possible at higher

layers: an attacker can establish two independent sessions and simply forward any data it receives from one into the other.

In the absence of any higher-level concept of peer identity, the use of session identifiers does not prevent session concatenation. The value to an attacker is limited unless information from the TLS connection is extracted and used with the signaling. For instance, a key exporter [EXPORTER] might be used to create a shared secret or unique identifier that is used in a secondary protocol.

If a secondary protocol uses the signaling channel with the assumption that the signaling and TLS peers are the same then that protocol is vulnerable to attack unless they also validate the identity of peers at both layers. Use of the "external_session_id" does not guarantee that the identity of the peer at the TLS layer is the same as the identity of the signaling peer.

It is important to note that multiple connections can be created within the same signaling session. An attacker might concatenate only part of a session, choosing to terminate some connections (and optionally forward data) while arranging to have peers interact directly for other connections. It is even possible to have different peers interact for each connection. This means that the actual identity of the peer for one connection might differ from the peer on another connection.

Information extracted from a TLS connection therefore MUST NOT be used in a secondary protocol outside of that connection if that protocol relies on the signaling protocol having the same peers. Similarly, data from one TLS connection MUST NOT be used in other TLS connections even if they are established as a result of the same signaling session.

6. Security Considerations

This entire document contains security considerations.

7. IANA Considerations

This document registers two extensions in the TLS "ExtensionType Values" registry established in [TLS13]:

- The "external_id_hash" extension defined in Section 3.2 has been assigned a code point of TBD; it is recommended and is marked as "Encrypted" in TLS 1.3.

- The "external_session_id" extension defined in Section 4.3 has been assigned a code point of TBD; it is recommended and is marked as "Encrypted" in TLS 1.3.

8. References

8.1. Normative References

[ASCII] Cerf, V., "ASCII format for network interchange", STD 80, RFC 20, DOI 10.17487/RFC0020, October 1969, <https://www.rfc-editor.org/info/rfc20>.

[DTLS] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <https://www.rfc-editor.org/info/rfc6347>.

[DTLS-SDP] Holmberg, C. and R. Shpount, "Session Description Protocol (SDP) Offer/Answer Considerations for Datagram Transport Layer Security (DTLS) and Transport Layer Security (TLS)", draft-ietf-mmusic-dtls-sdp-32 (work in progress), October 2017.

[DTLS-SRTP] Fischl, J., Tschofenig, H., and E. Rescorla, "Framework for Establishing a Secure Real-time Transport Protocol (SRTP) Security Context Using Datagram Transport Layer Security (DTLS)", RFC 5763, DOI 10.17487/RFC5763, May 2010, <https://www.rfc-editor.org/info/rfc5763>.

[FINGERPRINT] Lennox, J. and C. Holmberg, "Connection-Oriented Media Transport over the Transport Layer Security (TLS) Protocol in the Session Description Protocol (SDP)", RFC 8122, DOI 10.17487/RFC8122, March 2017, <https://www.rfc-editor.org/info/rfc8122>.

[PASSPoRT] Wendt, C. and J. Peterson, "PASSporT: Personal Assertion Token", RFC 8225, DOI 10.17487/RFC8225, February 2018, <https://www.rfc-editor.org/info/rfc8225>.

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

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

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

[SHA] Dang, Q., "Secure Hash Standard", National Institute of Standards and Technology report, DOI 10.6028/nist.fips.180-4, July 2015.

[SIP-ID] Peterson, J., Jennings, C., Rescorla, E., and C. Wendt, "Authenticated Identity Management in the Session Initiation Protocol (SIP)", RFC 8224, DOI 10.17487/RFC8224, February 2018, <https://www.rfc-editor.org/info/rfc8224>.

[SRTP] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, DOI 10.17487/RFC3711, March 2004, <https://www.rfc-editor.org/info/rfc3711>.

[TLS13] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[UTF8] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <https://www.rfc-editor.org/info/rfc3629>.

[WEBRTC-SEC] Rescorla, E., "WebRTC Security Architecture", draft-ietf-rtcweb-security-arch-17 (work in progress), November 2018.

8.2. Informative References

[AGILITY] Housley, R., "Guidelines for Cryptographic Algorithm Agility and Selecting Mandatory-to-Implement Algorithms", BCP 201, RFC 7696, DOI 10.17487/RFC7696, November 2015, <https://www.rfc-editor.org/info/rfc7696>.

[BASE64] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/info/rfc4648>.

[EXPORTER] Rescorla, E., "Keying Material Exporters for Transport Layer Security (TLS)", RFC 5705, DOI 10.17487/RFC5705, March 2010, <https://www.rfc-editor.org/info/rfc5705>.

[ICE] Keranen, A., Holmberg, C., and J. Rosenberg, "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal", RFC 8445, DOI 10.17487/RFC8445, July 2018, <https://www.rfc-editor.org/info/rfc8445>.

[JSON] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

[RFC3725] Rosenberg, J., Peterson, J., Schulzrinne, H., and G. Camarillo, "Best Current Practices for Third Party Call Control (3pcc) in the Session Initiation Protocol (SIP)", BCP 85, RFC 3725, DOI 10.17487/RFC3725, April 2004, <https://www.rfc-editor.org/info/rfc3725>.

[RTP] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <https://www.rfc-editor.org/info/rfc3550>.

[SIGMA] Krawczyk, H., "SIGMA: The 'SIGn-and-MAc'approach to authenticated Diffie-Hellman and its use in the IKE protocols", Annual International Cryptology Conference, Springer, pp. 400-425 , 2003.

[UKS] Blake-Wilson, S. and A. Menezes, "Unknown Key-Share Attacks on the Station-to-Station (STS) Protocol", Lecture Notes in Computer Science 1560, Springer, pp. 154-170 , 1999.

[WEBRTC] Bergkvist, A., Burnett, D., Narayanan, A., Jennings, C., Aboba, B., Brandstetter, T., and J. Bruaroey, "WebRTC 1.0: Real-time Communication Between Browsers", W3C Editor's Draft , November 2018.

Appendix A. Acknowledgements

This problem would not have been discovered if it weren't for discussions with Sam Scott, Hugo Krawczyk, and Richard Barnes. A solution similar to the one presented here was first proposed by Karthik Bhargavan who provided valuable input on this document.

Thyla van der Merwe assisted with a formal model of the solution.
Adam Roach and Paul E. Jones provided significant review and input.

Authors' Addresses

Martin Thomson
Mozilla

Email: mt@lowentropy.net

Eric Rescorla
Mozilla

Email: ekr@rftm.com