RTCWEB WG
Internet-Draft
Intended status: Informational
Expires: October 6, 2016

E. Rescorla
RTFM, Inc.

April 04, 2016

Piggybacked DTLS Handshakes in SDP
draft-rescorla-dtls-in-sdp-00

Abstract

This document describes a mechanism for embedding DTLS handshake messages in SDP descriptions. This technique allows implementations to shave a full round-trip off of DTLS-SRTP session establishment, while retaining compatibility with ordinary DTLS-SRTP endpoints.

Status of This Memo

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

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

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

This Internet-Draft will expire on October 6, 2016.

Table of Contents

1. Introduction

DTLS-SRTP [RFC5763][RFC5763] uses a DTLS [RFC6347] handshake to establish keys which are then used to key SRTP [RFC3711]. The DTLS negotiation is tied to the offer/answer [RFC3264] transaction via an "a=fingerprint" attribute [RFC4572] in the SDP [RFC4566]. The common message flow is shown below for DTLS 1.2.

This figure and the rest of this document adopt the following assumptions about network behavior:

- ICE [RFC5245] is in use but that both endpoints implement endpoint-independent filtering [RFC5389] so that STUN checks succeed immediately.

- Signaling messages take the same time to be delivered as direct messages [this is generally false.]

Links to detailed diagrams with a more accurate vertical scale can be found below each diagram.

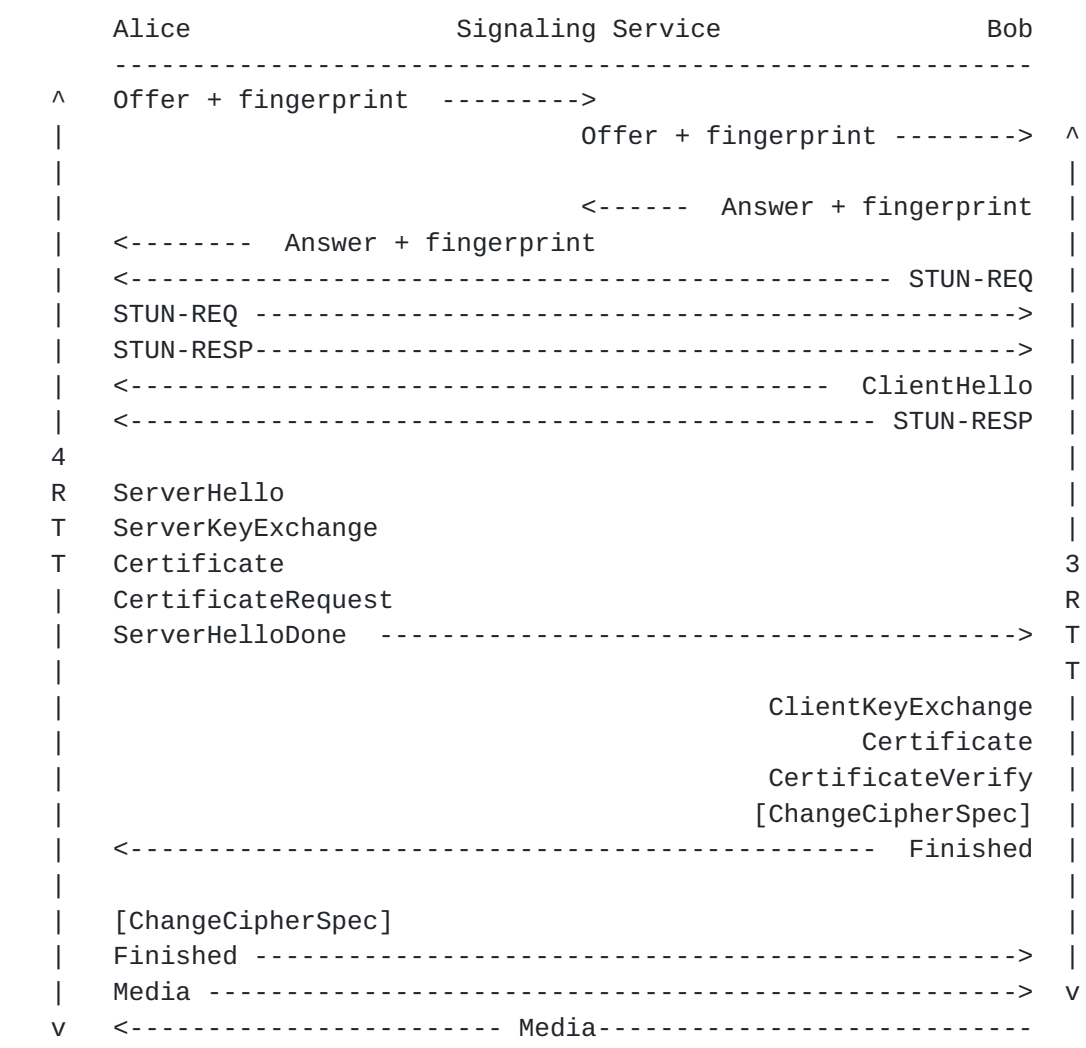

Figure 1: Standard DTLS-SRTP Negotiation

Better picture [1]

In this flow, the earliest that Alice can start sending media is after receiving Bob's Finished and the earliest Bob can start sending media is upon receiving Alice's Finished, and neither side can send any DTLS messages until they have had a successful STUN check. The result is that in the best case, Alice receives media four round trips after sending the offer and Bob receives media three round trips after receiving Alice's offer.

This document describes a technique for improving call setup time by piggybacking the first round of DTLS messages on the signaling messages. This reduces latency by a full round trip for both DTLS 1.2 and DTLS 1.3 handshakes, and for DTLS 1.3 [I-D.ietf-tls-tls13] allows the answerer to start sending media immediately upon receiving the offer, or, if ICE is used, upon ICE completion.

2. Protocol Overview

The basic concept, as shown in Figure 2, is for Alice to send her ClientHello in her Offer and Bob to send the server's first flight (ServerHello...ServerHelloDone for DTLS 1.2) in his Answer.

2.1. DTLS 1.2

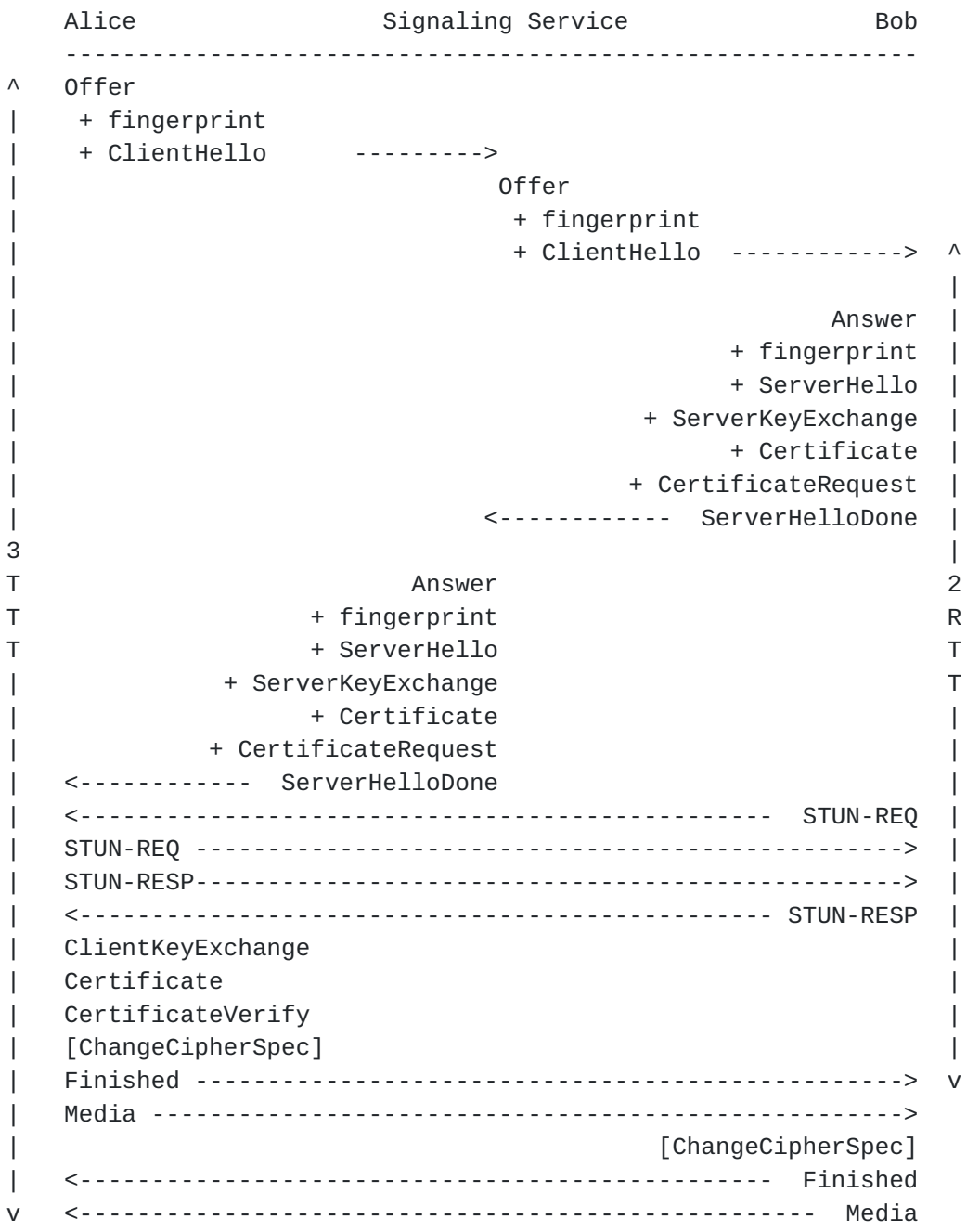

Figure 2: Piggybacked DTLS-SRTP Negotiation (TLS 1.2)

Better picture [2]

Note that in this flow, the active/passive (DTLS client/server) roles are reversed and Alice becomes the client. Because this is a basically symmetrical transaction, this is not an issue.

It should be immediately apparent that this exchange shaves off a full round trip from Bob's perspective (despite actually only shaving a half a round trip from the number of messages). The reason is that Bob does not need to wait for Alice's Finished to send but can piggyback his data on his Finished.

This change also shaves off a round trip from Alice's perspective because Alice can now safely perform TLS False Start [I-D.ietf-tls-falsestart] and send traffic prior to receiving Bob's Finished message. When only fingerprints are carried in the handshake, then extensions such as [RFC7301] indicators and DTLS-SRTP negotiation are not protected. However, in this case because those indicators are carried in the hello messages which are now tied to the signaling channel, they are authenticated via the same mechanisms that authenticate the fingerprint.

Note: One could argue that under some conditions Bob could do False Start in the ordinary handshake, but it's much harder to analyze and even then it leaves Alice one round trip slower than she would be with this optimization.

2.2. DTLS 1.3

Figure Figure 3 shows the impact of this optimization on DTLS 1.3.

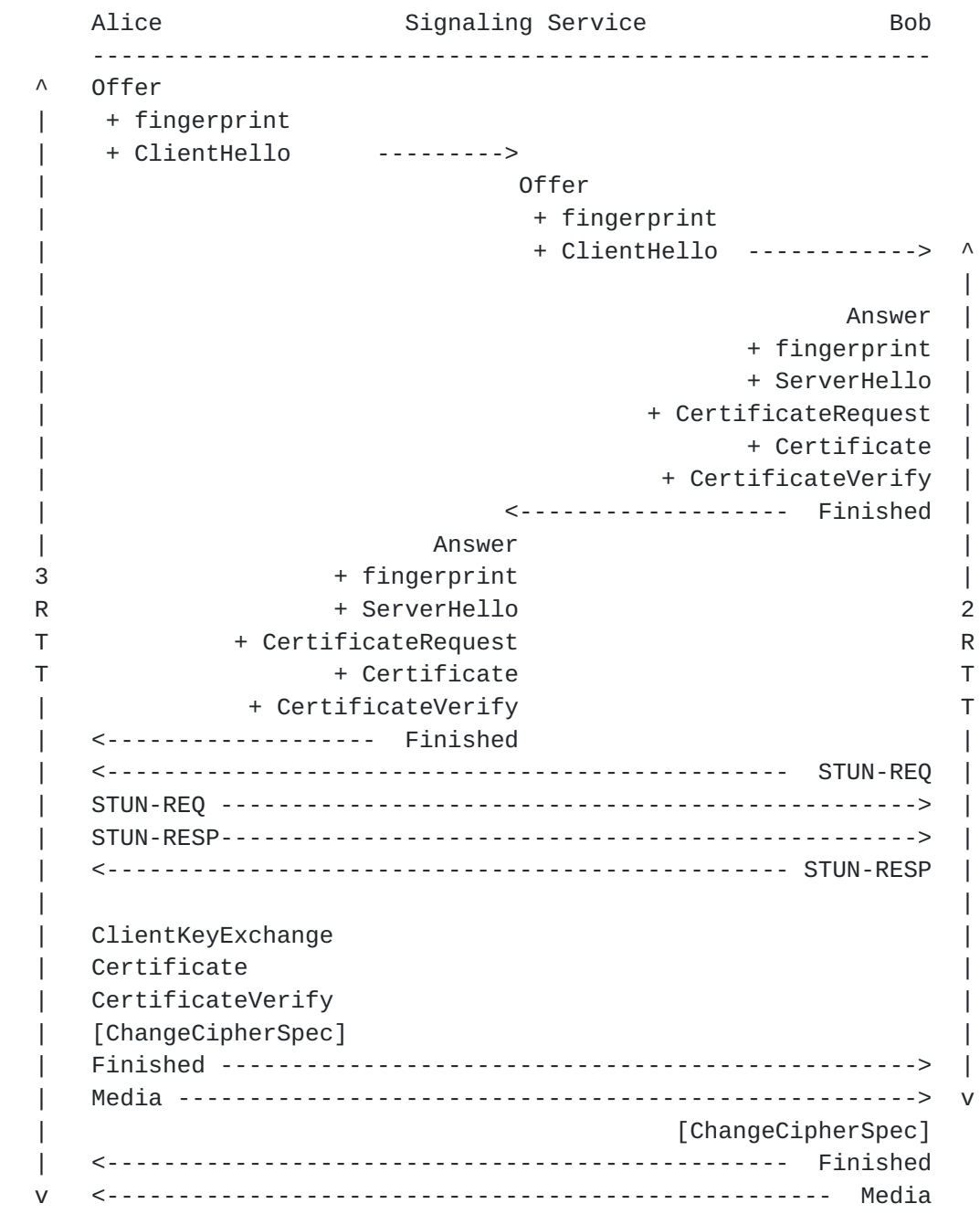

Figure 3: Piggybacked DTLS-SRTP Negotiation (TLS 1.3)

Better picture [3]

Alice cannot send any sooner than with DTLS 1.2 because sending at the point when she receives Bob's first message is already optimal. It may be possible for Bob to shave off yet another round trip, however. As described in Appendix A.

3. Attribute Definition

This document defines a new media-level SDP attribute, "a=dtls-message". This message is used to contain DTLS messages. The syntax of this attribute is:

```
attribute               =/   dtls-message-attribute

dtls-message-attribute  =    "dtls-message" ":" role SP value

role                    =    "client" / "server"

value                   =    1*(ALPHA / DIGIT / "+" / "/" / "=" )
                             ; base64 encoded message
```

An offeror which wishes to use the optimization defined in this document shall send his ClientHello in the "a=dtls-message" attribute of its initial offer with the role "client" and MUST use "a=setup:actpass". This allows the peer to either:

- Reject the optimization, in which case it ignores the attribute.
- Accept the optimization, in which case it MUST use "a=setup:passive" and send its first flight (starting with ServerHello) and using the role "server" in its response. These messages are simply serialized end-to-end as they would be on the wire. It MAY also choose to send its first flight separately in the media channel; DTLS implementations already handle retransmits properly.

The offerer MUST be able to detect whether an incoming DTLS message is a ClientHello or a ServerHello and adapt accordingly.

In subsequent negotiations, implementations MUST maintain these roles.

4. Interactions

This optimization has a number of interactions with existing pieces of protocol machinery.

4.1. ICE

When ICE is in use, there is a race condition between the answerer's ICE checks (at which point it will be able to send the first flight on the media channel) and the answerer's Answer, which contains the first flight. For this reason, we allow implementations to send the first flight on both channels. However, as a practical matter it is

reasonably likely that when ICE is in use the Answer will arrive first, for two reasons:

- The answerer consumes a full RTT doing a STUN check to verify the path to the offerer (even in the best case where the first STUN check succeeds). Thus, even if the path through the signaling server is twice as expensive as the direct path, there is a reasonable chance that the answer will arrive first.

- If the offerer is behind a NAT without endpoint-independent filtering, the answerer's ICE checks will be discarded until the offerer sends its own ICE checks, which it can only do upon receiving the answer.

In this case, although a comparison of Figure 1 and Figure 2 would show the ClientHello (in ordinary DTLS) and the ServerHello (when piggybacked) as arriving at the same time, in fact the ServerHello may arrive up to a full RTT first, but the offerer can SEND its second flight immediately upon its STUN check succeeding, which happens first, thus increasing the advantage of this technique.

4.2. Forking

This technique does not interact very well with forking. Because each ClientHello is only usable for one server, the system must somehow ensure that only one of the forks takes up the piggybacked offers. The easiest approach is for any intermediary which does a fork to strip out the "a=dtls-message" attribute. An alternative would be to add another attribute which could be stripped out (this might interact better with RTCWEB Identity). Note that [RFC4474] protects against any SDP modifications, but I think at this point it's clear that that's not practical.

4.3. RTCWEB Identity

RTCWEB Identity assertions need to cover these DTLS messages.

5. Examples

[we need examples.]

6. Security Considerations

The security implications of this technique are described throughout this document.

7. IANA Considerations

This specification defines the "dtls-message" SDP attribute per the procedures of Section 8.2.4 of [RFC4566]. The required information for the registration is included here:

Contact Name: Eric Rescorla (ekr@rftm.com)

Attribute Name: dtls-message

Long Form: dtls-message

Type of Attribute: session-level

Charset Considerations: This attribute is not subject to the charset attribute.

Purpose: This attribute carries piggybacked DTLS message.

Appropriate Values: This document

8. References

8.1. Normative References

[I-D.ietf-tls-falsestart]
Langley, A., Modadugu, N., and B. Moeller, "Transport Layer Security (TLS) False Start", draft-ietf-tls-falsestart-01 (work in progress), November 2015.

[I-D.ietf-tls-tls13]
Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", draft-ietf-tls-tls13-12 (work in progress), March 2016.

[RFC3264] 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>.

[RFC3711] 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, <http://www.rfc-editor.org/info/rfc3711>.

[RFC4566] 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>.

[RFC4572] 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>.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, DOI 10.17487/RFC5245, April 2010, <http://www.rfc-editor.org/info/rfc5245>.

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, DOI 10.17487/RFC5389, October 2008, <http://www.rfc-editor.org/info/rfc5389>.

[RFC5763] 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, <http://www.rfc-editor.org/info/rfc5763>.

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

[RFC7301] Friedl, S., Popov, A., Langley, A., and E. Stephan, "Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension", RFC 7301, DOI 10.17487/RFC7301, July 2014, <http://www.rfc-editor.org/info/rfc7301>.

8.2. Informative References

[RFC4474] Peterson, J. and C. Jennings, "Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)", RFC 4474, DOI 10.17487/RFC4474, August 2006, <http://www.rfc-editor.org/info/rfc4474>.

8.3. URIs

[1] https://raw.githubusercontent.com/ekr/dtls-in-sdp/master/normal-12.png

[2] https://raw.githubusercontent.com/ekr/dtls-in-sdp/master/piggybacked-12.png

[3] https://raw.githubusercontent.com/ekr/dtls-in-sdp/master/piggybacked-13.png

[4] https://raw.githubusercontent.com/ekr/dtls-in-sdp/master/piggybacked-13-falsestart.png

Appendix A. Speculative: Server False-Start

WARNING: THE FOLLOWING SECTION HAS NOT RECEIVED ANY REAL SECURITY REVIEW AND MAY BE A REALLY BAD IDEA.

It has been observed that as if Alice uses a fresh DH ephemeral, then Bob knows (because he can trust the signaling service) that Alice's DH ephemeral corresponds to Alice and can therefore encrypt under the joint DH shared secret without waiting for Alice's CertificateVerify, as shown in Figure 4.

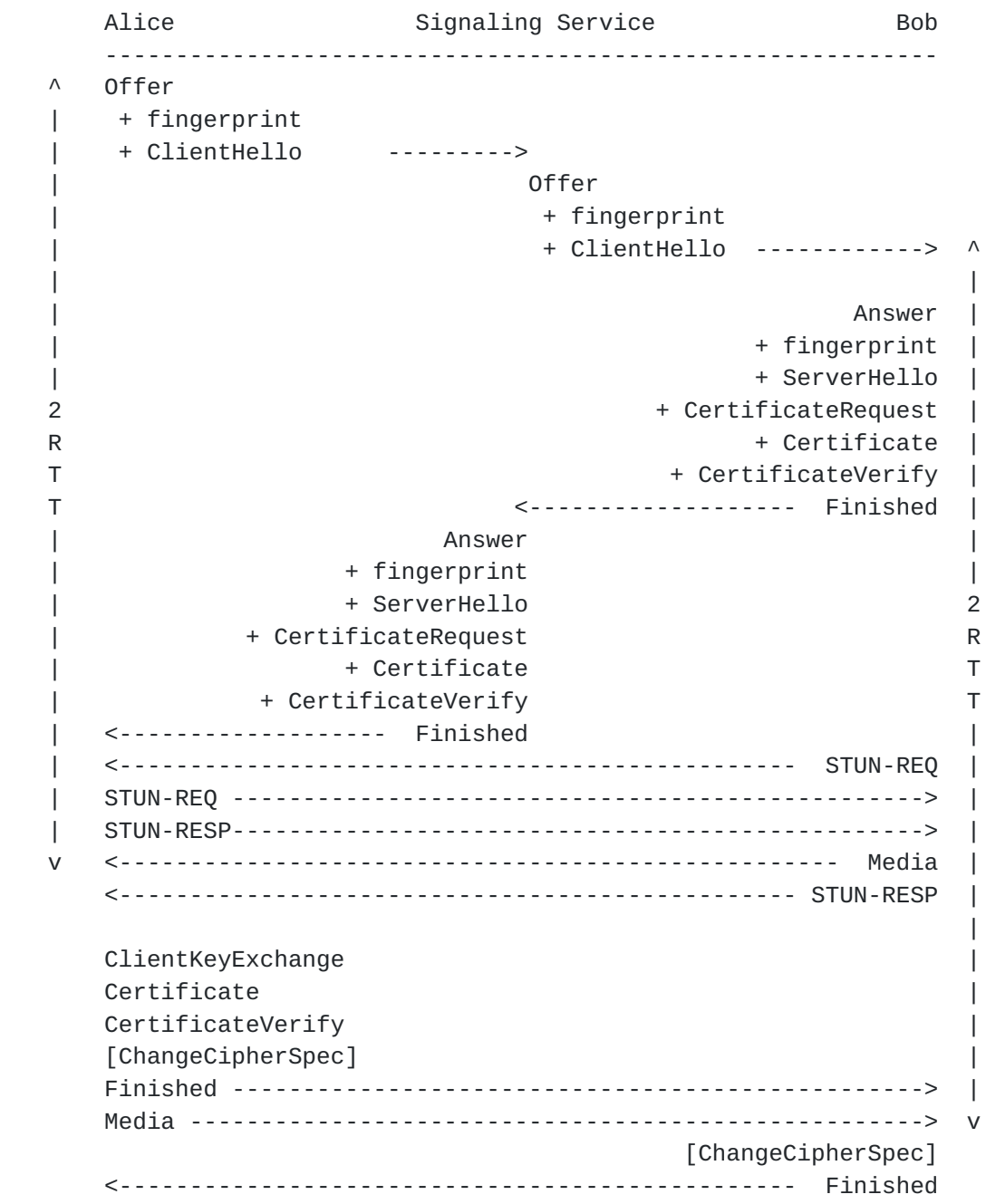

Figure 4: Piggybacked DTLS-SRTP Negotiation (TLS 1.3 with false start)

Better picture [4]

This has demonstrably inferior security properties if Alice is using a long-term key (for key continuity or fingerprint validation), because Bob has not yet verified that Alice controls that key and does not even know if Alice is using a fresh DH ephemeral, if implementations decide to adopt this optimization, they must do

something hacky like Send data immediately but generate an error if the handshake, including a signature, does not complete within some reasonable period (a small number of measured round trips) [Just one reason why this is a questionable technique.].

Appendix B. Acknowledgements

Thanks to Cullen Jennings, Martin Thomson, and Justin Uberti for helpful suggestions.

Author's Address

Eric Rescorla
RTFM, Inc.

Email: ekr@rtfm.com