Workgroup: jhoyla
Internet-Draft:
draft-jhoyla-tls-extended-key-schedule-03
Published: 4 December 2020
Intended Status: Standards Track
Expires: 7 June 2021
Authors: J. Hoyland
Cloudflare Ltd.
C.A. Wood
Cloudflare

TLS 1.3 Extended Key Schedule

Abstract

TLS 1.3 is sometimes used in situations where it is necessary to inject extra key material into the handshake. This draft aims to describe methods for doing so securely. This key material must be injected in such a way that both parties agree on what is being injected and why, and further, in what order.

Note to Readers

Discussion of this document takes place on the TLS Working Group mailing list (tls@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/tls/.

Source for this draft and an issue tracker can be found at https://github.com/jhoyla/draft-jhoyla-tls-key-injection.

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 7 June 2021.

Copyright Notice

Table of Contents

1. Introduction
2. Conventions and Definitions
3. Key Schedule Extension
   3.1. Handshake Secret Injection
   3.2. Main Secret Injection
4. Key Schedule Injection Negotiation
5. Key Schedule Extension Structure
6. Security Considerations
7. IANA Considerations
8. References
   8.1. Normative References
   8.2. Informative References
Appendix A. Potential Use Cases
Acknowledgments
Authors' Addresses

1. Introduction

Introducing additional key material into the TLS handshake is a non-trivial process because both parties need to agree on the injection content and context. If the two parties do not agree then an attacker may exploit the mismatch in so-called channel synchronization attacks, such as those described by [SLOTH].

Injecting key material into the TLS handshake allows other protocols to be bound to the handshake. For example, it may provide additional protections to the ClientHello message, which in the standard TLS handshake only receives protections after the server's Finished message has been received. It may also permit the use of combined shared secrets, possibly from multiple key exchange algorithms, to be included in the key schedule. This pattern is common for Post Quantum key exchange algorithms, as discussed in [I-D.ietf-tls-hybrid-design]. In particular, [I-D.ietf-tls-hybrid-design] uses the concatenation pattern described in this draft, but does not add the requisite framing.

The goal of this document is to provide a standardised way for binding extra context into TLS 1.3 handshakes in a way that is easy to analyse from a security perspective, reducing the need for security analysis of extensions that affect the key schedule. It separates the concerns of whether an extension achieves its goals from the concerns of whether an extension reduces the security of a TLS handshake, either directly or through some unforseen interaction with another extension.

2. Conventions and Definitions

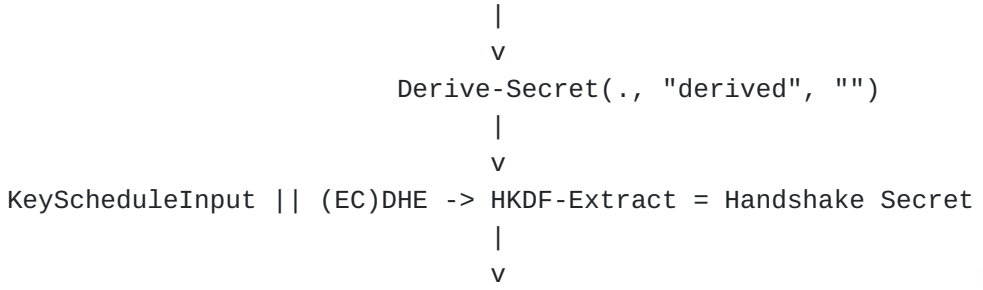
The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Key Schedule Extension

This section describes two places in which additional secrets can be injected into the TLS 1.3 key schedule.

3.1. Handshake Secret Injection

To inject extra key material into the Handshake Secret it is recommended to prefix it, inside an appropriate frame, to the (EC)DHE input, where || represents concatenation.



3.2. Main Secret Injection

To inject key material into the Main Secret it is recommended to prefix it, inside an appropriate frame, to the 0 input.

This structure mirrors the Handshake Injection point.

4. Key Schedule Injection Negotiation

Applications which make use of additional key schedule inputs MUST define a mechanism for negotiating the content and type of that input. This input MUST be framed in a KeyScheduleSecret struct, as defined in Section 5. Applications must take care that any negotiation that takes place unambiguously agrees a secret. It must be impossible, even under adversarial conditions, that a client and server agree on the transcript of the negotiation, but disagree on the secret that was negotiated.

5. Key Schedule Extension Structure

In some cases, protocols may require more than one secret to be injected at a particular stage in the key schedule. Thus, we require a generic and extensible way of doing so. To accomplish this, we use a structure - KeyScheduleInput - that encodes well-ordered sequences of secret material to inject into the key schedule. KeyScheduleInput is defined as follows:

```
struct {
    KeyScheduleSecretType type;
    opaque secret_data<0..2^16-1>;
} KeyScheduleSecret;

enum {
    (65535)
} KeyScheduleSecretType;

struct {
    KeyScheduleSecret secrets<0..2^16-1>;
} KeyScheduleInput;
```

Each secret included in a KeyScheduleInput structure has a type and corresponding secret data. Each secret MUST have a unique KeyScheduleSecretType. When encoding KeyScheduleInput as the key schedule Input value, the KeyScheduleSecret values MUST be in ascending sorted order. This ensures that endpoints always encode the same KeyScheduleInput value when using the same secret keying material.

6. Security Considerations

[BINDEL] provides a proof that the concatenation approach in Section 3 is secure as long as either the concatenated secret is secure or the existing KDF input is secure.

[[OPEN ISSUE: Is this guarantee sufficient? Do we also need to guarantee that a malicious prefix can't weaken the resulting PRF output?]]

7. IANA Considerations

This document requests the creation of a new IANA registry: TLS KeyScheduleInput Types. This registry should be under the existing Transport Layer Security (TLS) Parameters heading. It should be administered under a Specification Required policy [RFC8126].

[[OPEN ISSUE: specify initial registry values]]

Value	Description	DTLS-OK	Reference
TBD	TBD	TBD	TBD

Table 1

8. References

8.1. Normative References

[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>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[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>.

8.2. Informative References

[BINDEL] Bindel, N., Brendel, J., Fischlin, M., Goncalves, B., and D. Stebila, "Hybrid Key Encapsulation Mechanisms and Authenticated Key Exchange", Post-Quantum Cryptography pp. 206-226, DOI 10.1007/978-3-030-25510-7_12, 2019, <https://doi.org/10.1007/978-3-030-25510-7_12>.

[I-D.friel-tls-eap-dpp] Friel, O. and D. Harkins, "Bootstrapped TLS Authentication", Work in Progress, Internet-Draft, draft-

friel-tls-eap-dpp-01, 13 July 2020, <http://www.ietf.org/internet-drafts/draft-friel-tls-eap-dpp-01.txt>.

[I-D.ietf-tls-hybrid-design]
Steblia, D., Fluhrer, S., and S. Gueron, "Hybrid key exchange in TLS 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-hybrid-design-01, 15 October 2020, <http://www.ietf.org/internet-drafts/draft-ietf-tls-hybrid-design-01.txt>.

[I-D.ietf-tls-semistatic-dh]
Rescorla, E., Sullivan, N., and C. Wood, "Semi-Static Diffie-Hellman Key Establishment for TLS 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-semistatic-dh-01, 7 March 2020, <http://www.ietf.org/internet-drafts/draft-ietf-tls-semistatic-dh-01.txt>.

[SLOTH] Bhargavan, K. and G. Leurent, "Transcript Collision Attacks: Breaking Authentication in TLS, IKE, and SSH", Proceedings 2016 Network and Distributed System Security Symposium, DOI 10.14722/ndss.2016.23418, 2016, <https://doi.org/10.14722/ndss.2016.23418>.

Appendix A. Potential Use Cases

The draft provides a mechanism for importing additional information into the TLS key schedule. Future applications and specifications can use this mechanism to layer TLS on to other protocols, as opposed to layering other protocols over TLS. For example, as discussed in Section 1, this can be used for hybrid key exchange, which, in effect, is layering TLS over a secondary AKE. Although the key exchanges are interleaved, the post-quantum AKE completes first, as demonstrated by its output key being used as an input for computing TLS's master secret.

This can also be used in more direct ways, such as bootstrapping EAP-TLS as in [I-D.friel-tls-eap-dpp]. This draft also allows for more direct implementations of things such as semi-static DH [I-D.ietf-tls-semistatic-dh]. The aim of this draft is to be sufficiently flexible that it can be used as the basis for layering TLS on top of any protocol that outputs a secure channel binding, where secure is defined by the goals of the overall layered protocol. This draft does not provide security itself, it simply provides a standard format for layering.

Acknowledgments

We thank Karthik Bhargavan for his comments.

Authors' Addresses

Jonathan Hoyland
Cloudflare Ltd.

Email: jonathan.hoyland@gmail.com

Christopher A. Wood
Cloudflare

Email: caw@heapingbits.net