jhoyla
Internet-Draft
Intended status: Standards Track
Expires: May 7, 2020

J. Hoyland
Cloudflare Ltd.
C. Wood
Apple, Inc.

November 04, 2019

TLS 1.3 Extended Key Schedule
draft-jhoyla-tls-extended-key-schedule-00

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/ [1].

Source for this draft and an issue tracker can be found at https://github.com/jhoyla/draft-jhoyla-tls-key-injection [2].

Status of This Memo

Table of Contents

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.

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.stebila-tls-hybrid-design].

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

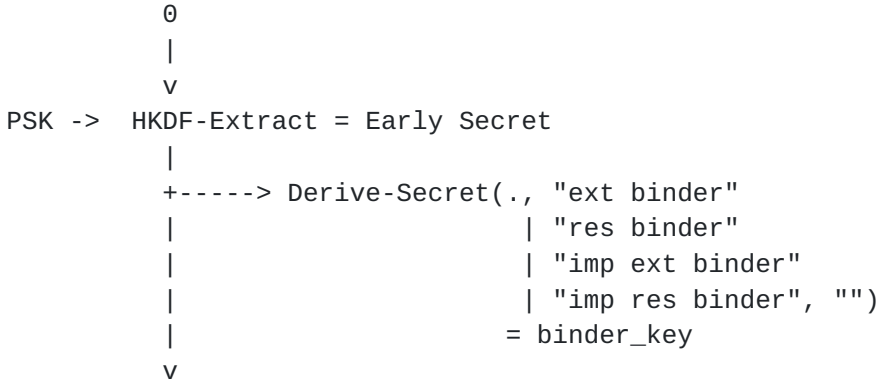
This section describes two ways in which additional secrets can be injected into the TLS 1.3 key schedule.

3.1. Early Secret Injection

TLS provides exporter keys that allow for other protocols to provide data authenticated by the TLS channel. This can be used to bind a protocol to a specific TLS handshake, giving joint authentication guarantees. In a similar way, one may wish to introduce externally authenticated and pre-shared data to the early secret derivation. This can be used to bind TLS to an external protocol.

To achieve this, pre-shared keys modify the binder key computation. This is needed since it ensures that both parties agree on both the authenticated data and the context in which it was used.

The binder key computation change is as follows:



Use of the "imp ext binder" label implies that both parties agree that there is some context that has been agreed, and that they are using an external PSK. Use of the "imp res binder" label implies that both parties agree that there is some context that has been agreed, and that they are using an resumption PSK. This assumes the

PSK has some mechanism by which additional context is included. [I-D.ietf-tls-external-psk-importer] describes one way by which such context may be included.

```
  struct {
    opaque external_identity<1...2^16-1>;
    opaque context<0...2^16>;
  } PSKIDWithAdditionalData;
```

external_identity is the "PSK_ID" that would have been used if the additional data were not agreed upon.

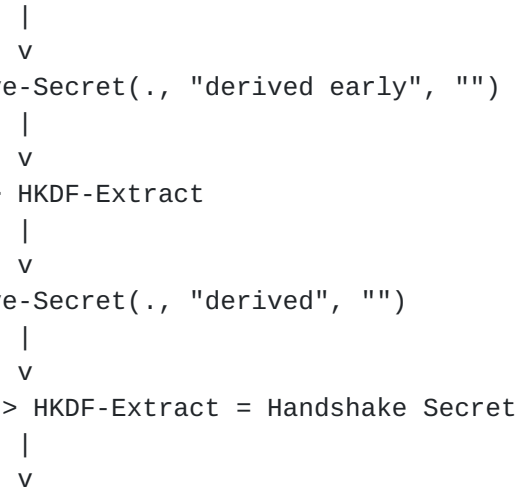
context is an opaque value that is bound to the agreed upon additional data.

Those using the "imp ext binder" or "imp res binder" label MUST include a "context" field, to allow the additional data.

Note that this structure is recursive. If this mechanism is used multiple times then the "external_identity" field will contain previous contexts in sequential order. If the client does not know in advance which pieces of additional data the server will be willing to agree on, it can provide multiple binders with different subsets of the additional data. The server can then select a binder with which it is willing to proceed. The binders MUST be verified in an all-or-nothing manner, and only one binder SHOULD be checked. A server MUST NOT accept a binder for which it only agrees upon some of the data.

3.2. Handshake Secret Injection

To inject key material into the Handshake Secret it is recommended to use an extra derive secret.



As shown in the figure above, the key schedule has an extra derive secret and HKDF-Extract step. This extra step isolates the Input material from the rest of the handshake secret, such that even maliciously chosen values cannot weaken the security of the key schedule overall.

The additional Derive-Secret with the "derived early" label enforces the separation of the key schedule from vanilla TLS handshakes, because HKDFs can be assumed to ensure that keys derived with different labels are independent.

4. 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.

5. Security Considerations

[[OPEN ISSUE: This draft has not seen any security analysis.]]

6. IANA Considerations

[[TODO: define secret registry structure]]

7. References

7.1. Normative References

[I-D.ietf-tls-external-psk-importer]
Benjamin, D. and C. Wood, "Importing External PSKs for TLS", draft-ietf-tls-external-psk-importer-01 (work in progress), October 2019.

[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>.

7.2. Informative References

[I-D.stebila-tls-hybrid-design]
Steblia, D., Fluhrer, S., and S. Gueron, "Design issues for hybrid key exchange in TLS 1.3", draft-stebila-tls-hybrid-design-01 (work in progress), July 2019.

7.3. URIs

[1] https://mailarchive.ietf.org/arch/browse/tls/

[2] https://github.com/jhoyla/draft-jhoyla-tls-key-injection

Acknowledgments

We thank Karthik Bhargavan for his comments.

Authors' Addresses

Jonathan Hoyland
Cloudflare Ltd.

Email: jonathan.hoyland@gmail.com

Christopher A. Wood
Apple, Inc.

Email: cawood@apple.com