

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 4, 2018

M. Kuehlewind
ETH Zurich
T. Pauly
C. Wood
Apple Inc.
July 03, 2017

Separating Crypto Negotiation and Communication
draft-kuehlewind-taps-crypto-sep-00

Abstract

Due to the latency involved in connection setup and security handshakes, there is an increasing deployment of cryptographic session resumption mechanisms. While cryptographic context and endpoint capabilities need to be known before encrypted application data can be sent, there is otherwise no technical constraint that the crypto handshake must be performed on the same transport connection. This document recommends a logical separation between the mechanism(s) used to negotiate capabilities and set up encryption context (handshake protocol), the application of encryption and authentication state to data (record protocol), and the associated transport connection(s).

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 January 4, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terminology	3
3. Protocol Interfaces	4
3.1. Handshake-Transport Interface	5
3.2. Handshake-Record Interface	6
3.3. Transport-Record Interface	6
4. Existing Mappings	6
5. Benefits of Separation	8
5.1. Reducing Connection Latency	9
5.2. Protocol Flexibility	9
5.3. Protocol Capability Negotiation	10
6. IANA Considerations	10
7. Security Considerations	10
8. Acknowledgments	10
9. Informative References	10
Authors' Addresses	11

1. Introduction

Secure transport protocols are generally composed of three pieces:

1. A transport protocol to control the transfer of data.
2. A record protocol to frame, encrypt and/or authenticate data
3. A handshake protocol to negotiate cryptographic secrets.

For ease of deployment and standardization, among other reasons, these constituents are often tightly coupled. For example, in TLS [RFC5246], the handshake protocol depends on the record protocol, and vice versa. However, more recent transport protocols such as QUIC [I-D.ietf-quic-tls] keep these pieces separate. QUIC uses TLS to negotiate secrets, and _exports_ those secrets to encrypt packets directly.

Separating these pieces is important, as new secure transport protocols increasingly rely on session resumption mechanisms where cryptographic context can be resumed to transmit application data with the first packet without delay for connection setup and negotiation. In the case where there is no cryptographic context available when an application expresses the need to transmit data to a certain endpoint, it must first run the handshake protocol on a transport connection before being able to transmit application data. If the handshake protocol can be separated from the other components, then it can use another transport connection to establish secrets without blocking the application's main transport connection. This also opens up the possibility to run the handshake protocol well in advance of the need to send application data, to avoid unnecessary delays. For example, a client system could maintain a database of endpoints it is likely to communicate with, and establish keying material with a handshake protocol at periodic intervals to ensure fresh keys for new transport connections.

[I-D.moskowitz-sse] proposes a similar approach. However while [I-D.moskowitz-sse] proposes a new protocol to negotiate and maintain long-term cryptographic sessions, this document relies on the use of existing protocols and only discusses requirements for the evolution of these protocols and exchange of information within one endpoint locally.

2. Terminology

- o Transport Protocol: A protocol that can transport messages between two endpoints. This may represent the service offered to applications to allow them to send and receive data before encryption; and also represent the protocol that can transmit handshake data and encrypted records.
- o Handshake Protocol: A protocol that can validate and authenticate endpoints, encrypt and authenticate its negotiation, and ultimately generate keying material.
- o Record Protocol: A protocol that can use keying material to transform messages. A record will generally add a frame around application data, and authenticate and/or encrypt the data.
- o Keying Material: One or more pre-shared keys that can be used to encrypt and authenticate data, generated by a handshake protocol and used by a record protocol.

3. Protocol Interfaces

In traditional models in which the protocols are not separated out into the three elements of handshake, record, and transport protocols, there are two basic approaches to the interactions:

1. The transport protocol provides data to the security protocol and gets back an encrypted version of the data to be sent (handshake and record protocols are combined)
2. The security protocol provides keying material to the transport protocol, and the transport protocol is responsible for encrypting data (transport and record protocols are combined)

By teasing apart all three portions as separate protocols, there end up being six interface points:

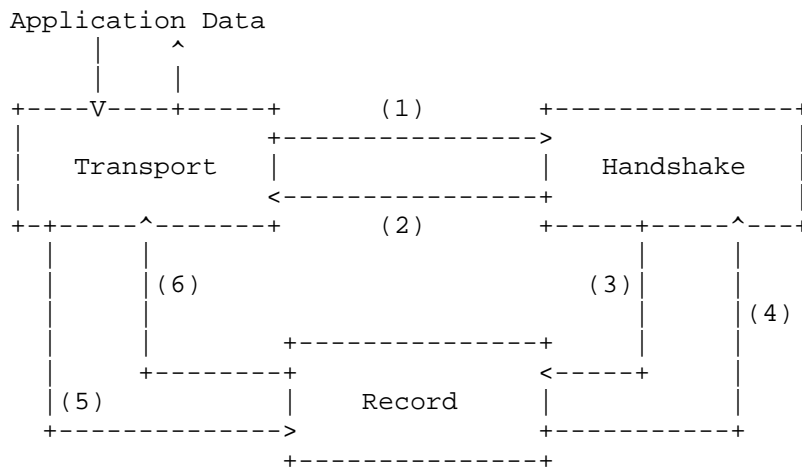


Figure 1: Secure Transport Protocol Components and Interactions

1. A transport protocol depends upon a handshake protocol to establish keying material to protect application data being sent through the transport. The main interface it relies upon is starting the handshake, or ensuring that the material is ready.
2. A handshake protocol depends upon a transport protocol in order to send and receive negotiation messages with the remote peer.
3. A handshake protocol sends its keying material and cryptographic context to the record protocol to use

4. A record protocol may signal state expiration events to a handshake protocol
5. A transport protocol uses a record protocol to send and receive application data
6. A record protocol uses a transport protocol to send and receive encrypted data

3.1. Handshake-Transport Interface

Note that for the purposes of this interface description, it is assumed that the application is primarily interacting with the transport protocol, and thus the handshake protocol interacts with the application primarily through the abstraction of the transport protocol.

- o Start negotiation: The interface MUST provide an indication to start the protocol handshake for key negotiation, and have a way to be notified when the handshake is complete.
- o Identity constraints: The interface MUST allow the application to constrain the identities that it will accept a connection to, such as the hostname it expects to be provided in certificate SAN.
- o Local identities: The interface MUST allow the local identity to be set via a raw private key or interface to one to perform cryptographic operations such as signing and decryption.
- o State changes: The interface SHOULD provide a way for the transport to be notified of important state changes during the protocol execution and session lifetime, e.g., when the handshake begins, ends, or when a key update occurs.
- o Validation: The interface MUST provide a way for the application to participate in the endpoint authentication and validation, which can either be specified as parameters to define how the peer's authentication can be validated, or when the protocol provides the authentication information for the application to inspect directly.
- o Caching domain and lifetime: The application SHOULD be able to specify the instances of the protocol that can share cached keys, as well as the lifetime of cached resources.
- o The protocol SHOULD allow applications to negotiate application protocols and related information.

- o The protocol SHOULD allow applications to specify negotiable cryptographic algorithm suites.
- o The protocol SHOULD expose the peer's identity information.

3.2. Handshake-Record Interface

- o Key export: The interface MUST provide a way to export keying material from a handshake protocol to a record protocol with well-defined cryptographic properties, e.g., "forward-secure" or "perfectly forward secure"
- o Key lifetime and rotation: The interface MUST provide a way for the handshake protocol to define key lifetime bounds in terms of `_time_` or `_bytes encrypted_` and, additionally, provide a way to forcefully update cryptographic session keys at will. The record protocol MUST be able to signal back to the handshake protocol that a lifetime has been reached and that rotation is required. These values SHOULD be configurable by the application.

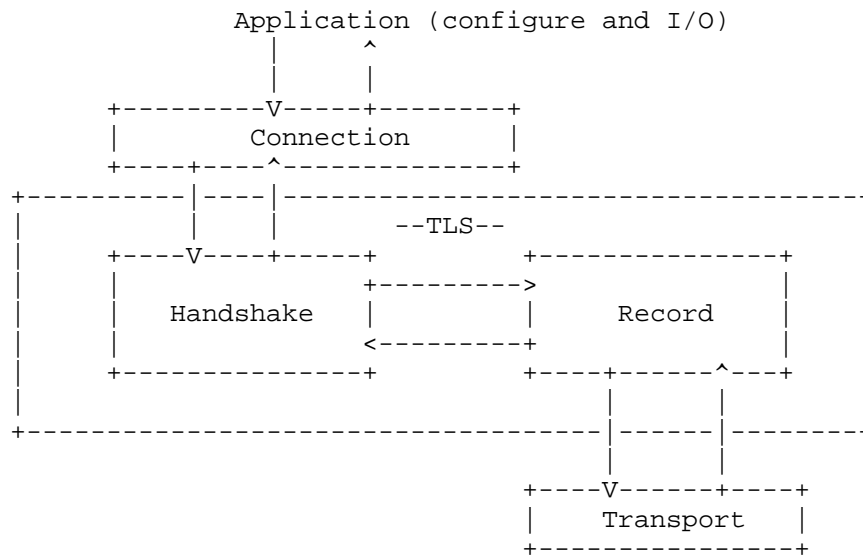
3.3. Transport-Record Interface

- o Transform data: The interface MUST provide a way to send raw application data from the transport protocol to a record protocol to transform it based on the keying material. This data is then sent out by the transport protocol. The same applies for inbound data, in which inbound transport data is transformed by the record protocol into raw application data.
- o Reliability: The transport MUST specify if messages are transmitted reliable and in order.
- o Maximum message size (optional): The transport may specify a maximum message size for the encrypted data if e.g. a datagram transport is used

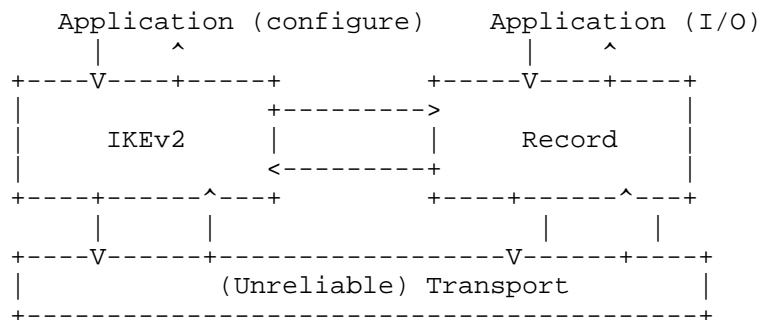
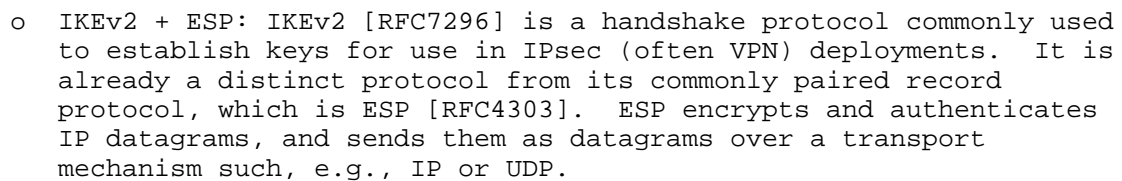
4. Existing Mappings

In this section we document existing mappings between common transport security protocols and the three components described in Section I.

- o TLS/DTLS: TLS [RFC5246] and DTLS [RFC6347] is a combination of a handshake and record protocol, with a dependency on some underlying transport.



- o QUIC + TLS: The emerging QUIC standard is decomposed into the three pieces outlined in Section I [I-D.ietf-quic-tls]. TLS is used as the handshake protocol running on a dedicated QUIC stream, a QUIC-specific record protocol encrypts and encapsulates stream frames, and the main QUIC component handles the transport of these frames.



5. Benefits of Separation

5.1. Reducing Connection Latency

One of the clearest benefits of separating the handshake protocol from the record protocol is that the handshake can be performed out-of-band from the application's data transfer. This should essentially reduce the number of RTTs required before being able to send data by the full length of the handshake (which is commonly 1 or 2 RTTs in the best cases for TLS 1.2 and IKEv2, potentially more if cookie challenges or extended authentication are required).

To avoid long-lived transport connections that wouldn't be actively used, and thus would be vulnerable to timeouts on NATs or firewalls, an obvious approach to separating the handshake and record protocols is to use different transport connections for the early handshake and the data transfer. However, this approach of using separate connections will not always save RTTs if the handshake and data transfer are back-to-back. Each connection may require its own transport protocol handshake, and if the data transfer must wait for two transport protocols to establish and the cryptographic handshake to be finished before sending, then it may experience higher latency. Implementations SHOULD avoid this by either allowing the handshake and record protocols to share a single transport connection or open two connections in parallel when the handshake protocol has not pre-fetched keys. Latency benefits, however, can even be achieved when ensuring that this scenario does not occur by always having the handshake protocol refresh the keys whenever old ones are near expiry.

5.2. Protocol Flexibility

Separation of the handshake, record, and transport protocols also allows for more flexible composition of protocols with one another. If a deployment uses a handshake protocol like TLS, which requires a stream-based transport protocol like TCP, separation of protocols will allow it to use the resulting keys for record protocols that run on datagram transport protocols like UDP.

This flexibility may be useful for implementations that are optimizing for packet size by choosing minimal/lightweight record protocols, while being able to use commonly supported handshake protocols like TLS. One example here is the approach of a VPN tunnel that uses ESP or Diet-ESP [I-D.mglt-ipsecme-diet-esp] to encrypt datagrams, but uses TLS for establishing keys.

5.3. Protocol Capability Negotiation

Enabling the use of a different transport protocol for the actual data transmission than for the cryptographic handshakes opens also the possibility to negotiate protocol capabilities for the data transmission. For TLS, usually TCP is the appropriate transport protocol to use, as it is also widely supported by endpoints. Allowing an endpoint to indicate the support of other, new transport protocols within the TCP connection that is used for the handshake, provides a dynamic transition path to enable easy deployment of new protocols.

6. IANA Considerations

This document has on request to IANA.

7. Security Considerations

(editor's note: this section will be added later. However, this document discusses the use of cryptographic context for transport connections and as such it has security relevant consideration within the whole document.)

8. Acknowledgments

This work is partially supported by the European Commission under Horizon 2020 grant agreement no. 688421 Measurement and Architecture for a Middleboxed Internet (MAMI), and by the Swiss State Secretariat for Education, Research, and Innovation under contract no. 15.0268. This support does not imply endorsement.

9. Informative References

[I-D.ietf-quic-tls]

Thomson, M. and S. Turner, "Using Transport Layer Security (TLS) to Secure QUIC", draft-ietf-quic-tls-04 (work in progress), June 2017.

[I-D.mglt-ipsecme-diet-esp]

Migault, D., Guggemos, T., and C. Bormann, "ESP Header Compression and Diet-ESP", draft-mglt-ipsecme-diet-esp-04 (work in progress), June 2017.

[I-D.moskowitz-sse]

Moskowitz, R., Faynberg, I., Lu, H., Hares, S., and P. Giacomin, "Session Security Envelope", draft-moskowitz-sse-05 (work in progress), June 2017.

- [RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <<http://www.rfc-editor.org/info/rfc4303>>.
- [RFC5246] 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>>.
- [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>>.
- [RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <<http://www.rfc-editor.org/info/rfc7296>>.
- [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>>.

Authors' Addresses

Mirja Kuehlewind
ETH Zurich
Gloriastrasse 35
8092 Zurich
Switzerland

Email: mirja.kuehlewind@tik.ee.ethz.ch

Tommy Pauly
Apple Inc.
1 Infinite Loop
Cupertino, California 95014
United States of America

Email: tpauly@apple.com

Christopher A. Wood
Apple Inc.
1 Infinite Loop
Cupertino, California 95014
United States of America

Email: cawood@apple.com