Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 1, 2019

M. Kuehlewind
ETH Zurich
T. Pauly
C. Wood
Apple Inc.

June 30, 2018

Separating Crypto Negotiation and Communication
draft-kuehlewind-taps-crypto-sep-03

Abstract

Secure transport protocols often consist of three logically distinct components: transport, control (handshake), and record protection. Typically, such a protocol contains a single module that is responsible for all three functions. However, in many cases, this coupling is unnecessary. For example, while cryptographic context and endpoint capabilities need to be known before encrypted application data can be sent on a specific transport connection, there is otherwise no technical constraint that a cryptographic handshake must be performed on said connection. This document recommends a logical separation between transport, control, and record components of secure transport protocols. We compare existing protocols such as Transport Layer Security, QUIC, and IKEv2+ESP in the context of this logical separation.

Table of Contents

1. Introduction

Secure transport protocols are generally composed of three pieces:

1. A transport protocol to handle the transfer of data.

2. A record protocol to frame, encrypt and/or authenticate data

3. A control protocol to perform cryptographic handshakes, negotiate shared secrets, and maintain state during the lifetime of cryptographic session including session resumption and key

refreshment. (In the context of TLS, the control protocol is called the handshake protocol.)

For ease of deployment and standardization, among other reasons, these constituents are often tightly coupled. For example, in TLS [RFC5246], the control 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. For example, QUIC uses TLS to negotiate secrets, and exports those secrets to encrypt packets independent of TLS.

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 control protocol on a transport connection before being able to transmit application data. If the control 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 control 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 control 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

- 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 control data and encrypted records.

- Control Protocol: A protocol that performs a cryptographic handshake and, in addition, can validate and authenticate endpoints, encrypt and authenticate its negotiation, and ultimately generate keying material.

- 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.

- Keying Material: A shared secret from which pre-shared keys can be derived and subsequently used to encrypt and authenticate data, generated by a control 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 control, 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 (control 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:

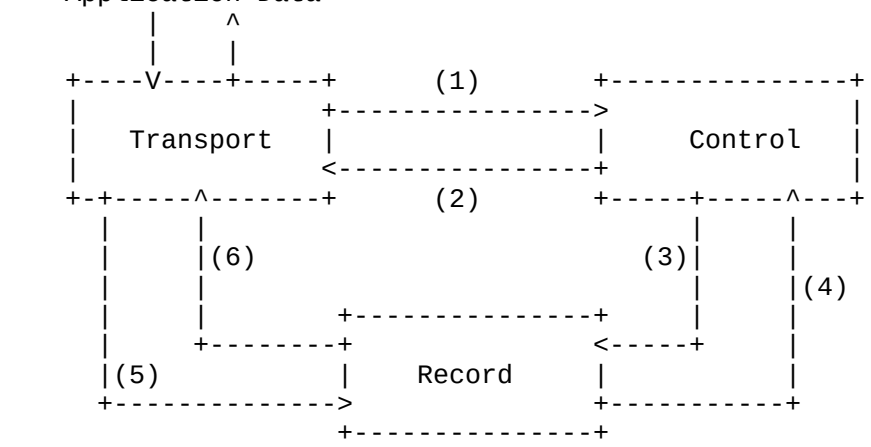


Figure 1: Secure Transport Protocol Components and Interactions

1. A transport protocol depends upon a control protocol to establish keying material to protect application data being sent through the transport. The main interface it relies upon is starting the

control channel, or handshake, or ensuring that the material is ready.

2. A control protocol depends upon a transport protocol in order to send and receive negotiation messages with the remote peer.

3. A control 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 control 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. Control-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 control protocol interacts with the application primarily through the abstraction of the transport protocol. Since security protocol interfaces often require pre-connection and active behavior on behalf of clients, we further categorize the following interfaces based on whether they are meant for passive configuration or active control.

3.1.1. Passive Configuration Interface

- 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.

- 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.

- 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.

- 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.

- Pre-shared keying material: The application SHOULD be able to specify pre-share keying material to use to bootstrap connections. The control protocol can pass this directly to the record protocol for use.

- The protocol SHOULD allow applications to negotiate application protocols and related information.

- The protocol SHOULD allow applications to specify negotiable cryptographic algorithm suites.

3.1.2. Active Control and Introspection Interface

- 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.

- 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.

- The protocol SHOULD expose the peer's identity information during and after connection establishment.

3.2. Control-Record Interface

- Key export: The interface MUST provide a way to export keying material from a control protocol to a record protocol with well-defined cryptographic properties, e.g., "forward-secure."

- Key lifetime and rotation: The interface MUST provide a way for the control 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 control 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

- 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.

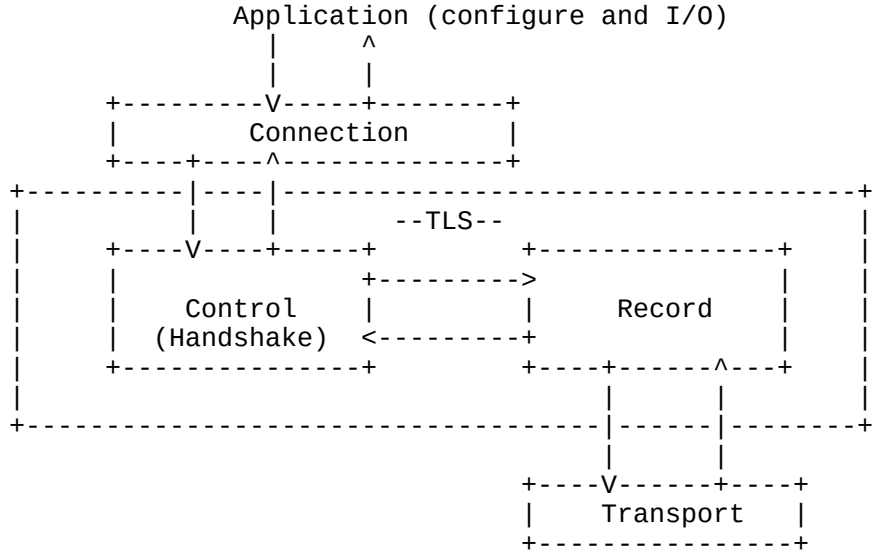
- Reliability: The transport MUST specify if messages are transmitted reliable and in order.

- Maximum message size (optional): The transport may specify a maximum message size for the encrypted data if e.g. a datagram transport is used

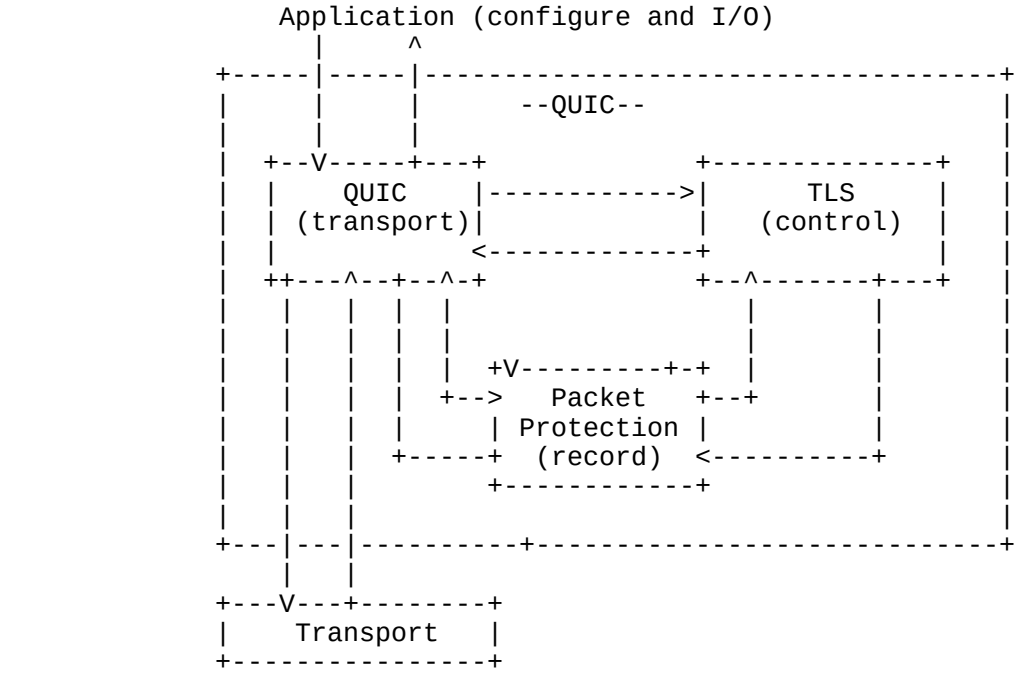
4. Existing Mappings

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

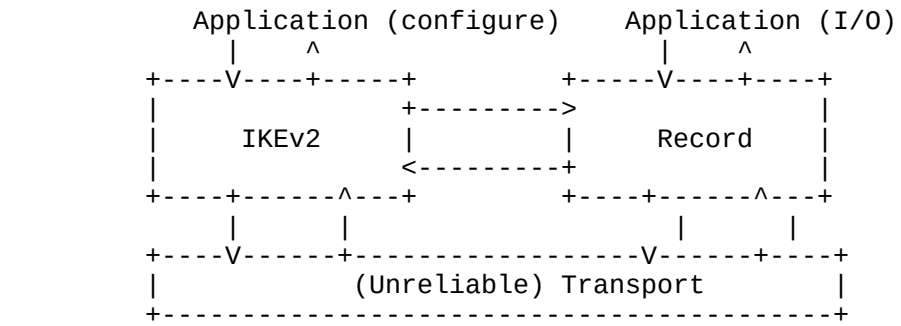
- TLS/DTLS: TLS [RFC5246] and DTLS [RFC6347] is a combination of a control (handshake) and record protocol, with a dependency on some underlying transport.



- 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 control 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.

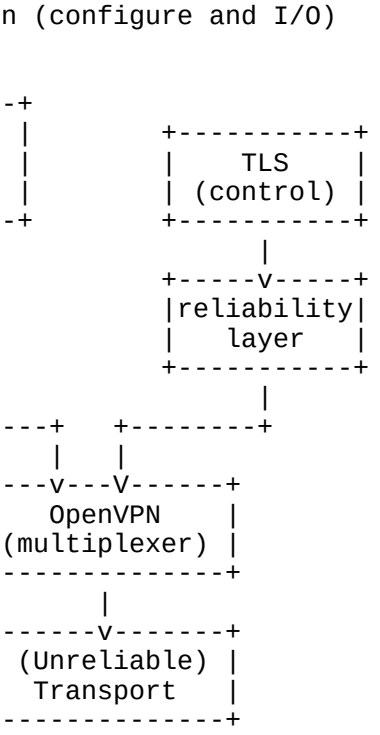


o IKEv2 + ESP: IKEv2 [RFC7296] is a control protocol commonly used to establish keys for use in IPsec (often VPN) deployments. It is already a distinct protocol from its commonly paired record protocol, which is ESP [RFC4303]. ESP encrypts and authenticates IP datagrams, and sends them as datagrams over a transport mechanism such, e.g., IP or UDP.

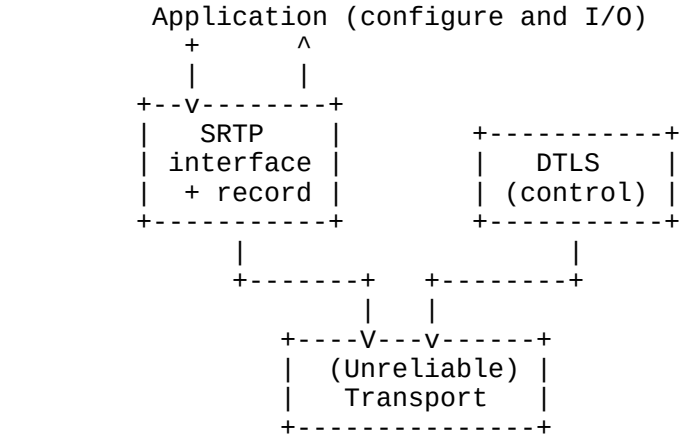


o OpenVPN [OpenVPN]: OpenVPN consists of two separate stacks - one for TLS, which is used for key exchange and derivation, and the other as an interface to tunnel IP packets over UDP. A common multiplexing layer is used to send TLS and OpenVPN framed packets over an unreliable transport layer. OpenVPN adds a reliability

layer to TLS to ensure packets are sent and processed in order. Running over TCP naturally provides this reliability. After the TLS connection finishes, OpenVPN extracts encryption and authentication keys from TLS, via the PRF, and uses them to encrypt and authenticate IP packets. Packets are framed using a simple length-type-value envelope, wherein the type specifies the contents of the packet, e.g., channel control (TLS ciphertext) bytes.



o DTLS-SRTP: DTLS [RFC5764] is commonly used as a way to perform mutual authentication and key agreement for SRTP [RFC5763]. (Here, certificates marshal public keys between endpoints. Thus, self- signed certificates may be used if peers do not mutually trust one another, as is common on the Internet.) When DTLS is used, certificate fingerprints are transmitted out-of-band using SIP. Peers typically verify that DTLS-offered certificates match that which are offered over SIP. This prevents active attacks on RTP, but not on the signaling (SIP or WebRTC) channel.



5. Benefits of Separation

5.1. Reducing Connection Latency

One of the clearest benefits of separating the control protocol from the record protocol is that the cryptographic 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 control 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 cryptographic 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 control and record protocols to share a single transport connection or open two connections in parallel when the control 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 control protocol refresh the keys whenever old ones are near expiry.

5.2. Protocol Flexibility

Separation of the control, record, and transport protocols also allows for more flexible composition of protocols with one another. If a deployment uses a control 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 control 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. This design is similar to that used by OpenVPN [OpenVPN], as described above.

5.3. Protocol Capability and Upgrade 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 cryptographic handshake, provides a dynamic transition path to enable easy deployment of new protocols. Another example is providing an upgrade path from TCP+TLS to QUIC. If TLS could negotiate the use of other transport layers, such as QUIC, applications could perform an abbreviated upgrade from TCP+TLS connections to QUIC, i.e., without doing a full QUIC handshake.

6. Transport Service Architecture Integration

The Transport Services Architecture ([I-D.ietf-taps-arch]) describes a system that can provide transport security functionality behind a common interface. Such systems and their APIs provide applications with the ability to establish connections for sending and receiving data. The lifetime of a connection is comprised of a pre-establishment configuration stage, established (connected) stage, and terminated stage. Pre-establishment properties configured include: Local and Remote Endpoint, protocol selection properties, and specific protocol options. Applications configure security protocols during pre-establishment using the passive interfaces described in Section Section 3.1. Active control interfaces are exercised during connection establishment, i.e., from pre-establishment to established states. Applications can query connection metadata or state

information, e.g., peer identity information, during and after connection establishment.

7. IANA Considerations

This document has on request to IANA.

8. 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.)

9. 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. Thanks to Brian Trammell for reviewing this draft.

10. Informative References

[I-D.ietf-quic-tls]
Thomson, M. and S. Turner, "Using Transport Layer Security (TLS) to Secure QUIC", draft-ietf-quic-tls-13 (work in progress), June 2018.

[I-D.ietf-taps-arch]
Pauly, T., Trammell, B., Brunstrom, A., Fairhurst, G., Perkins, C., Tiesel, P., and C. Wood, "An Architecture for Transport Services", draft-ietf-taps-arch-00 (work in progress), April 2018.

[I-D.mglt-ipsecme-diet-esp]
Migault, D., Guggemos, T., Bormann, C., and D. Schinazi, "ESP Header Compression and Diet-ESP", draft-mglt-ipsecme-diet-esp-06 (work in progress), May 2018.

[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.

[OpenVPN] "OpenVPN Security Overview", n.d., <https://openvpn.net/index.php/open-source/documentation/security-overview.html>.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <https://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, <https://www.rfc-editor.org/info/rfc5246>.

[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, <https://www.rfc-editor.org/info/rfc5763>.

[RFC5764] McGrew, D. and E. Rescorla, "Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)", RFC 5764, DOI 10.17487/RFC5764, May 2010, <https://www.rfc-editor.org/info/rfc5764>.

[RFC6347] 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>.

[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, <https://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, <https://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.
One Apple Park Way
Cupertino, California 95014
United States of America

Email: tpauly@apple.com

Christopher A. Wood
Apple Inc.
One Apple Park Way
Cupertino, California 95014
United States of America

Email: cawood@apple.com