Network Working Group
Internet-Draft
Intended status: Informational
Expires: May 21, 2020

T. Enghardt
TU Berlin
T. Pauly
Apple Inc.
C. Perkins
University of Glasgow
K. Rose
Akamai Technologies, Inc.
C. Wood, Ed.
Apple Inc.

November 18, 2019

A Survey of the Interaction Between Security Protocols and Transport Services

draft-ietf-taps-transport-security-10

Abstract

This document provides a survey of commonly used or notable network security protocols, with a focus on how they interact and integrate with applications and transport protocols. Its goal is to supplement efforts to define and catalog transport services by describing the interfaces required to add security protocols. This survey is not limited to protocols developed within the scope or context of the IETF, and those included represent a superset of features a Transport Services system may need to support.

Table of Contents

1. Introduction 3
1.1. Goals 4
1.2. Non-Goals 4
2. Terminology 4
3. Transport Security Protocol Descriptions 6
3.1. Application Payload Security Protocols 6
3.1.1. TLS 6
3.1.2. DTLS 6
3.2. Application-Specific Security Protocols 6
3.2.1. Secure RTP 6
3.2.2. ZRTP for Media Path Key Agreement 7
3.3. Transport-Layer Security Protocols 7
3.3.1. QUIC with TLS 7
3.3.2. Google QUIC 7
3.3.3. tcpcrypt 7
3.3.4. MinimalT 7
3.3.5. CurveCP 8
3.4. Packet Security Protocols 8
3.4.1. IKEv2 with ESP 8
3.4.2. WireGuard 8
3.4.3. OpenVPN 8
4. Transport Dependencies 9
4.1. Reliable Byte-Stream Transports 9
4.2. Unreliable Datagram Transports 9
4.2.1. Datagram Protocols with Defined Byte-Stream Mappings 10
4.3. Transport-Specific Dependencies 10
5. Application Interface 10
5.1. Pre-Connection Interfaces 11
5.2. Connection Interfaces 13
5.3. Post-Connection Interfaces 13
6. IANA Considerations 15

7. Security Considerations 15
8. Privacy Considerations 15
9. Acknowledgments 15
10. Informative References 15
Authors' Addresses 18

1. Introduction

Services and features provided by transport protocols have been cataloged in [RFC8095]. This document supplements that work by surveying commonly used and notable network security protocols, and identifying the services and features a Transport Services system (a system that provides a transport API) needs to provide in order to add transport security. It examines Transport Layer Security (TLS), Datagram Transport Layer Security (DTLS), QUIC + TLS, tcpcrypt, Internet Key Exchange with Encapsulating Security Protocol (IKEv2 + ESP), SRTP (with DTLS), WireGuard, CurveCP, and MinimalT. For each protocol, this document provides a brief description, the dependencies it has on the underlying transports, and the interfaces provided to applications.

Selected protocols represent a superset of functionality and features a Transport Services system may need to support, both internally and externally (via an API) for applications [I-D.ietf-taps-arch]. Ubiquitous IETF protocols such as (D)TLS, as well as non-standard protocols such as Google QUIC, are both included despite overlapping features. As such, this survey is not limited to protocols developed within the scope or context of the IETF. Outside of this candidate set, protocols that do not offer new features are omitted. For example, newer protocols such as WireGuard make unique design choices that have implications and limitations on application usage. In contrast, protocols such as ALTS [ALTS] are omitted since they do not provide interfaces deemed unique.

Authentication-only protocols such as TCP-AO [RFC5925] and IPsec AH [RFC4302] are excluded from this survey. TCP-AO adds authenticity protections to long-lived TCP connections, e.g., replay protection with per-packet Message Authentication Codes. (This protocol obsoletes TCP MD5 "signature" options specified in [RFC2385].) One prime use case of TCP-AO is for protecting BGP connections. Similarly, AH adds per-datagram authenticity and adds similar replay protection. Despite these improvements, neither protocol sees general use and both lack critical properties important for emergent transport security protocols: confidentiality, privacy protections, and agility. Such protocols are thus omitted from this survey.

1.1. Goals

This survey is intended to help identify the most common interface surfaces between security protocols and transport protocols, and between security protocols and applications.

One of the goals of Transport Services is to define a common interface for using transport protocols that allows software using transport protocols to easily adopt new protocols that provide similar feature-sets. The survey of the dependencies security protocols have upon transport protocols can guide implementations in determining which transport protocols are appropriate to be able to use beneath a given security protocol. For example, a security protocol that expects to run over a reliable stream of bytes, like TLS, restrict the set of transport protocols that can be used to those that offer a reliable stream of bytes.

Defining the common interfaces that security protocols provide to applications also allows interfaces to be designed in a way that common functionality can use the same APIs. For example, many security protocols that provide authentication let the application be involved in peer identity validation. Any interface to use a secure transport protocol stack thus needs to allow applications to perform this action during connection establishment.

1.2. Non-Goals

While this survey provides similar analysis to that which was performed for transport protocols in [RFC8095], it is important to distinguish that the use of security protocols requires more consideration.

It is not a goal to allow software implementations to automatically switch between different security protocols, even where their interfaces to transport and applications are equivalent. Even between versions, security protocols have subtly different guarantees and vulnerabilities. Thus, any implementation needs to only use the set of protocols and algorithms that are requested by applications or by a system policy.

2. Terminology

The following terms are used throughout this document to describe the roles and interactions of transport security protocols:

- Transport Feature: a specific end-to-end feature that the transport layer provides to an application. Examples include

confidentiality, reliable delivery, ordered delivery, message-versus-stream orientation, etc.

- Transport Service: a set of Transport Features, without an association to any given framing protocol, which provides functionality to an application.

- Transport Protocol: an implementation that provides one or more different transport services using a specific framing and header format on the wire. A Transport Protocol services an application.

- Application: an entity that uses a transport protocol for end-to-end delivery of data across the network. This may also be an upper layer protocol or tunnel encapsulation.

- Security Protocol: a defined network protocol that implements one or more security features, such as authentication, encryption, key generation, session resumption, and privacy. Security protocols may be used alongside transport protocols, and in combination with other security protocols when appropriate.

- Handshake Protocol: a protocol that enables peers to validate each other and to securely establish shared cryptographic context.

- Record: Framed protocol messages.

- Record Protocol: a security protocol that allows data to be divided into manageable blocks and protected using shared cryptographic context.

- Session: an ephemeral security association between applications.

- Connection: the shared state of two or more endpoints that persists across messages that are transmitted between these endpoints. A connection is a transient participant of a session, and a session generally lasts between connection instances.

- Peer: an endpoint application party to a session.

- Client: the peer responsible for initiating a session.

- Server: the peer responsible for responding to a session initiation.

3. Transport Security Protocol Descriptions

This section contains brief descriptions of the various security protocols currently used to protect data being sent over a network. The interfaces between these protocols and transports are described in Section 4; the interfaces between these protocols and applications are described in Section 5.

3.1. Application Payload Security Protocols

The following protocols provide security that protects application payloads sent over a transport. They do not specifically protect any headers used for transport-layer functionality.

3.1.1. TLS

TLS (Transport Layer Security) [RFC8446] is a common protocol used to establish a secure session between two endpoints. Communication over this session "prevents eavesdropping, tampering, and message forgery." TLS consists of a tightly coupled handshake and record protocol. The handshake protocol is used to authenticate peers, negotiate protocol options, such as cryptographic algorithms, and derive session-specific keying material. The record protocol is used to marshal (possibly encrypted) data from one peer to the other. This data may contain handshake messages or raw application data.

3.1.2. DTLS

DTLS (Datagram Transport Layer Security) [RFC6347] is based on TLS, but differs in that it is designed to run over unreliable datagram protocols like UDP instead of TCP. DTLS modifies the protocol to make sure it can still provide the same security guarantees as TLS even without reliability from the transport. DTLS was designed to be as similar to TLS as possible, so this document assumes that all properties from TLS are carried over except where specified.

3.2. Application-Specific Security Protocols

The following protocols provide application-specific security by protecting application payloads used for specific use-cases. Unlike the protocols above, these are not intended for generic application use.

3.2.1. Secure RTP

Secure RTP (SRTP) is a profile for RTP that provides confidentiality, message authentication, and replay protection for RTP data packets and RTP control protocol (RTCP) packets [RFC3711].

3.2.2. ZRTP for Media Path Key Agreement

ZRTP [RFC6189] is an alternative key agreement protocol for SRTP. It uses standard SRTP to protect RTP data packets and RTCP packets, but provides alternative key agreement and identity management protocols. Key agreement is performed using a Diffie-Hellman key exchange that runs on the media path. This generates a shared secret that is then used to generate the master key and salt for SRTP.

3.3. Transport-Layer Security Protocols

The following security protocols provide protection for both application payloads and headers that are used for transport services.

3.3.1. QUIC with TLS

QUIC is a new standards-track transport protocol that runs over UDP, loosely based on Google's original proprietary gQUIC protocol [I-D.ietf-quic-transport] (See Section 3.3.2 for more details). The QUIC transport layer itself provides support for data confidentiality and integrity. This requires keys to be derived with a separate handshake protocol. A mapping for QUIC of TLS 1.3 [I-D.ietf-quic-tls] has been specified to provide this handshake.

3.3.2. Google QUIC

Google QUIC (gQUIC) is a UDP-based multiplexed streaming protocol designed and deployed by Google following experience from deploying SPDY, the proprietary predecessor to HTTP/2. gQUIC was originally known as "QUIC": this document uses gQUIC to unambiguously distinguish it from the standards-track IETF QUIC. The proprietary technical forebear of IETF QUIC, gQUIC was originally designed with tightly-integrated security and application data transport protocols.

3.3.3. tcpcrypt

Tcpcrypt [RFC8548] is a lightweight extension to the TCP protocol for opportunistic encryption. Applications may use tcpcrypt's unique session ID for further application-level authentication. Absent this authentication, tcpcrypt is vulnerable to active attacks.

3.3.4. MinimalT

MinimalT is a UDP-based transport security protocol designed to offer confidentiality, mutual authentication, DoS prevention, and connection mobility [MinimalT]. One major goal of the protocol is to leverage existing protocols to obtain server-side configuration

information used to more quickly bootstrap a connection. MinimalT uses a variant of TCP's congestion control algorithm.

3.3.5. CurveCP

CurveCP [CurveCP] is a UDP-based transport security protocol from Daniel J. Bernstein. Unlike other security protocols, it is based entirely upon highly efficient public key algorithms. This removes many pitfalls associated with nonce reuse and key synchronization. CurveCP provides its own reliability for application data as part of its protocol.

3.4. Packet Security Protocols

The following protocols provide protection for IP packets. These are generally used as tunnels, such as for Virtual Private Networks (VPNs). Often, applications will not interact directly with these protocols. However, applications that implement tunnels will interact directly with these protocols.

3.4.1. IKEv2 with ESP

IKEv2 [RFC7296] and ESP [RFC4303] together form the modern IPsec protocol suite that encrypts and authenticates IP packets, either for creating tunnels (tunnel-mode) or for direct transport connections (transport-mode). This suite of protocols separates out the key generation protocol (IKEv2) from the transport encryption protocol (ESP). Each protocol can be used independently, but this document considers them together, since that is the most common pattern.

3.4.2. WireGuard

WireGuard is an IP-layer protocol designed as an alternative to IPsec [WireGuard] for certain use cases. It uses UDP to encapsulate IP datagrams between peers. Unlike most transport security protocols, which rely on PKI for peer authentication, WireGuard authenticates peers using pre-shared public keys delivered out-of-band, each of which is bound to one or more IP addresses. Moreover, as a protocol suited for VPNs, WireGuard offers no extensibility, negotiation, or cryptographic agility.

3.4.3. OpenVPN

OpenVPN [OpenVPN] is a commonly used protocol designed as an alternative to IPsec. A major goal of this protocol is to provide a VPN that is simple to configure and works over a variety of transports. OpenVPN encapsulates either IP packets or Ethernet frames within a secure tunnel and can run over UDP or TCP.

4. Transport Dependencies

Across the different security protocols listed above, the primary dependency on transport protocols is the presentation of data: either an unbounded stream of bytes, or framed messages. Within protocols that rely on the transport for message framing, most are built to run over transports that inherently provide framing, like UDP, but some also define how their messages can be framed over byte-stream transports.

4.1. Reliable Byte-Stream Transports

The following protocols all depend upon running on a transport protocol that provides a reliable, in-order stream of bytes. This is typically TCP.

Application Payload Security Protocols:

- TLS

Transport-Layer Security Protocols:

- tcpcrypt

Packet Security Protocols:

- OpenVPN

4.2. Unreliable Datagram Transports

The following protocols all depend on the transport protocol to provide message framing to encapsulate their data. These protocols are built to run using UDP, and thus do not have any requirement for reliability. Running these protocols over a protocol that does provide reliability will not break functionality, but may lead to multiple layers of reliability if the security protocol is encapsulating other transport protocol traffic.

Application Payload Security Protocols:

- DTLS
- SRTP
- ZRTP

Transport-Layer Security Protocols:

- QUIC
- MinimalT
- CurveCP

Packet Security Protocols:

- IKEv2 and ESP
- WireGuard

4.2.1. Datagram Protocols with Defined Byte-Stream Mappings

Of the protocols listed above that depend on the transport for message framing, some do have well-defined mappings for sending their messages over byte-stream transports like TCP.

Application Payload Security Protocols:

- SRTP [RFC7201]

Packet Security Protocols:

- IKEv2 and ESP [RFC8229]

4.3. Transport-Specific Dependencies

One protocol surveyed, tcpcrypt, has an direct dependency on a feature in the transport that is needed for its functionality. Specific, tcpcrypt is designed to run on top of TCP, and uses the TCP Encryption Negotiation Option (ENO) [RFC8547] to negotiate its protocol support.

QUIC, CurveCP, and MinimalT provide both transport functionality and security functionality. They have a dependencies on running over a framed protocol like UDP, but they add their own layers of reliability and other transport services. Thus, an application that uses one of these protocols cannot decouple the security from transport functionality.

5. Application Interface

This section describes the interface surface exposed by the security protocols described above. Note that not all protocols support each interface. We partition these interfaces into pre-connection (configuration), connection, and post-connection interfaces,

following conventions in [I-D.ietf-taps-interface] and [I-D.ietf-taps-arch].

5.1. Pre-Connection Interfaces

Configuration interfaces are used to configure the security protocols before a handshake begins or the keys are negotiated.

- Identities and Private Keys: The application can provide its identities (certificates) and private keys, or mechanisms to access these, to the security protocol to use during handshakes.
  - TLS
  - DTLS
  - SRTP
  - QUIC
  - MinimalT
  - CurveCP
  - IKEv2
  - WireGuard
- Supported Algorithms (Key Exchange, Signatures, and Ciphersuites): The application can choose the algorithms that are supported for key exchange, signatures, and ciphersuites.
  - TLS
  - DTLS
  - SRTP
  - QUIC
  - tcpcrypt
  - MinimalT
  - IKEv2

- Extensions (Application-Layer Protocol Negotiation): The application enables or configures extensions that are to be negotiated by the security protocol, such as ALPN [RFC7301].
  - TLS
  - DTLS
  - QUIC
- Session Cache Management: The application provides the ability to save and retrieve session state (such as tickets, keying material, and server parameters) that may be used to resume the security session.
  - TLS
  - DTLS
  - QUIC
  - MinimalT
- Authentication Delegation: The application provides access to a separate module that will provide authentication, using EAP for example.
  - SRTP
  - IKEv2
- Pre-Shared Key Import: Either the handshake protocol or the application directly can supply pre-shared keys for the record protocol use for encryption/decryption and authentication. If the application can supply keys directly, this is considered explicit import; if the handshake protocol traditionally provides the keys directly, it is considered direct import; if the keys can only be shared by the handshake, they are considered non-importable.
  - Explicit import: QUIC, ESP
  - Direct import: TLS, DTLS, tcpcrypt, MinimalT, WireGuard
  - Non-importable: CurveCP

5.2. Connection Interfaces

- Identity Validation: During a handshake, the security protocol will conduct identity validation of the peer. This can call into the application to offload validation.
  - TLS
  - DTLS
  - SRTP
  - QUIC
  - MinimalT
  - CurveCP
  - IKEv2
  - WireGuard
  - OpenVPN
- Source Address Validation: The handshake protocol may delegate validation of the remote peer that has sent data to the transport protocol or application. This involves sending a cookie exchange to avoid DoS attacks. Protocols: QUIC + TLS, DTLS, WireGuard
  - DTLS
  - QUIC
  - WireGuard

5.3. Post-Connection Interfaces

- Connection Termination: The security protocol may be instructed to tear down its connection and session information. This is needed by some protocols to prevent application data truncation attacks.
  - TLS
  - DTLS
  - QUIC
  - tcpcrypt

    * MinimalT

    * IKEv2

- Key Update: The handshake protocol may be instructed to update its keying material, either by the application directly or by the record protocol sending a key expiration event.

    * TLS

    * DTLS

    * QUIC

    * tcpcrypt

    * MinimalT

    * IKEv2

- Pre-Shared Key Export: The handshake protocol will generate one or more keys to be used for record encryption/decryption and authentication. These may be explicitly exportable to the application, traditionally limited to direct export to the record protocol, or inherently non-exportable because the keys must be used directly in conjunction with the record protocol.

    * Explicit export: TLS (for QUIC), DTLS (for SRTP), tcpcrypt, IKEv2

    * Direct export: TLS, DTLS, MinimalT

    * Non-exportable: CurveCP

- Key Expiration: The record protocol can signal that its keys are expiring due to reaching a time-based deadline, or a use-based deadline (number of bytes that have been encrypted with the key). This interaction is often limited to signaling between the record layer and the handshake layer.

    * ESP

- Mobility Events: The record protocol can be signaled that it is being migrated to another transport or interface due to connection mobility, which may reset address and state validation and induce state changes such as use of a new Connection Identifier (CID).

    * QUIC

* MinimalT
* CurveCP
* ESP
* WireGuard

6. IANA Considerations

This document has no request to IANA.

7. Security Considerations

This document summarizes existing transport security protocols and their interfaces. It does not propose changes to or recommend usage of reference protocols. Moreover, no claims of security and privacy properties beyond those guaranteed by the protocols discussed are made. For example, metadata leakage via timing side channels and traffic analysis may compromise any protocol discussed in this survey. Applications using Security Interfaces should take such limitations into consideration when using a particular protocol implementation.

8. Privacy Considerations

Analysis of how features improve or degrade privacy is intentionally omitted from this survey. All security protocols surveyed generally improve privacy by reducing information leakage via encryption. However, varying amounts of metadata remain in the clear across each protocol. For example, client and server certificates are sent in cleartext in TLS 1.2 [RFC5246], whereas they are encrypted in TLS 1.3 [RFC8446]. A survey of privacy features, or lack thereof, for various security protocols could be addressed in a separate document.

9. Acknowledgments

The authors would like to thank Bob Bradley, Frederic Jacobs, Mirja Kuehlewind, Yannick Sierra, Brian Trammell, and Magnus Westerlund for their input and feedback on this draft.

10. Informative References

[ALTS] Ghali, C., Stubblefield, A., Knapp, E., Li, J., Schmidt, B., and J. Boeuf, "Application Layer Transport Security", <https://cloud.google.com/security/encryption-in-transit/application-layer-transport-security/>.

[CurveCP] Bernstein, D., "CurveCP -- Usable security for the Internet", <http://curvecp.org>.

[I-D.ietf-quic-tls]
Thomson, M. and S. Turner, "Using TLS to Secure QUIC", draft-ietf-quic-tls-23 (work in progress), September 2019.

[I-D.ietf-quic-transport]
Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", draft-ietf-quic-transport-23 (work in progress), September 2019.

[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-04 (work in progress), July 2019.

[I-D.ietf-taps-interface]
Trammell, B., Welzl, M., Enghardt, T., Fairhurst, G., Kuehlewind, M., Perkins, C., Tiesel, P., Wood, C., and T. Pauly, "An Abstract Application Layer Interface to Transport Services", draft-ietf-taps-interface-04 (work in progress), July 2019.

[MinimalT]
Petullo, W., Zhang, X., Solworth, J., Bernstein, D., and T. Lange, "MinimaLT -- Minimal-latency Networking Through Better Security", <http://dl.acm.org/citation.cfm?id=2516737>.

[OpenVPN] "OpenVPN cryptographic layer", <https://openvpn.net/community-resources/openvpn-cryptographic-layer/>.

[RFC2385] Heffernan, A., "Protection of BGP Sessions via the TCP MD5 Signature Option", RFC 2385, DOI 10.17487/RFC2385, August 1998, <https://www.rfc-editor.org/info/rfc2385>.

[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, <https://www.rfc-editor.org/info/rfc3711>.

[RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <https://www.rfc-editor.org/info/rfc4302>.

[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>.

[RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, DOI 10.17487/RFC5925, June 2010, <https://www.rfc-editor.org/info/rfc5925>.

[RFC6189] Zimmermann, P., Johnston, A., Ed., and J. Callas, "ZRTP: Media Path Key Agreement for Unicast Secure RTP", RFC 6189, DOI 10.17487/RFC6189, April 2011, <https://www.rfc-editor.org/info/rfc6189>.

[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>.

[RFC7201] Westerlund, M. and C. Perkins, "Options for Securing RTP Sessions", RFC 7201, DOI 10.17487/RFC7201, April 2014, <https://www.rfc-editor.org/info/rfc7201>.

[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>.

[RFC8095] Fairhurst, G., Ed., Trammell, B., Ed., and M. Kuehlewind, Ed., "Services Provided by IETF Transport Protocols and Congestion Control Mechanisms", RFC 8095, DOI 10.17487/RFC8095, March 2017, <https://www.rfc-editor.org/info/rfc8095>.

[RFC8229] Pauly, T., Touati, S., and R. Mantha, "TCP Encapsulation of IKE and IPsec Packets", RFC 8229, DOI 10.17487/RFC8229, August 2017, <https://www.rfc-editor.org/info/rfc8229>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8547] Bittau, A., Giffin, D., Handley, M., Mazieres, D., and E. Smith, "TCP-ENO: Encryption Negotiation Option", RFC 8547, DOI 10.17487/RFC8547, May 2019, <https://www.rfc-editor.org/info/rfc8547>.

[RFC8548] Bittau, A., Giffin, D., Handley, M., Mazieres, D., Slack, Q., and E. Smith, "Cryptographic Protection of TCP Streams (tcpcrypt)", RFC 8548, DOI 10.17487/RFC8548, May 2019, <https://www.rfc-editor.org/info/rfc8548>.

[WireGuard] Donenfeld, J., "WireGuard -- Next Generation Kernel Network Tunnel", <https://www.wireguard.com/papers/wireguard.pdf>.

Authors' Addresses

Theresa Enghardt
TU Berlin
Marchstr. 23
10587 Berlin
Germany

Email: theresa@inet.tu-berlin.de

Tommy Pauly
Apple Inc.
One Apple Park Way
Cupertino, California 95014
United States of America

Email: tpauly@apple.com

Colin Perkins
University of Glasgow
School of Computing Science
Glasgow G12 8QQ
United Kingdom

Email: csp@csperkins.org

Kyle Rose
Akamai Technologies, Inc.
150 Broadway
Cambridge, MA 02144
United States of America

Email: krose@krose.org

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

Email: cawood@apple.com