Stream: Internet Engineering Task Force (IETF)
RFC: 9850
Category: Informational
Published: December 2025
ISSN: 2070-1721
Authors: M. Thomson *Mozilla* Y. Rosomakho *Zscaler* H. Tschofenig *H-BRS*

RFC 9850
The SSLKEYLOGFILE Format for TLS

Abstract

This document describes a format that supports logging information about the secrets used in a TLS connection. Recording secrets to a file in SSLKEYLOGFILE format allows diagnostic and logging tools that use this file to decrypt messages exchanged by TLS endpoints. This format is intended for use in systems where TLS only protects test data.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction 2

1.1. Applicability Statement 3

1.2. Conventions 3

2. The SSLKEYLOGFILE Format 3

2.1. Secret Labels for TLS 1.3 4

2.2. Secret Labels for TLS 1.2 5

2.3. Secret Labels for ECH 5

3. Security Considerations 6

4. IANA Considerations 7

4.1. SSLKEYLOGFILE Media Type 7

4.2. TLS SSLKEYLOGFILE Labels Registry 8

5. References 9

5.1. Normative References 9

5.2. Informative References 9

Appendix A. Example 10

Acknowledgments 12

Authors' Addresses 12

1. Introduction

Debugging or analyzing protocols can be challenging when TLS [TLS13] is used to protect the content of communications. Inspecting the content of encrypted messages in diagnostic tools can enable more thorough analysis.

Over time, multiple TLS implementations have informally adopted a file format for logging the secret values generated by the TLS key schedule. In many implementations, the file that the secrets are logged to is specified in an environment variable named "SSLKEYLOGFILE", hence the name of SSLKEYLOGFILE format. Note the use of "SSL" as this convention originally predates the adoption of TLS as the name of the protocol.

This document describes the SSLKEYLOGFILE format. This format can be used for TLS 1.2 [TLS12] and TLS 1.3 [TLS13]. The format also supports earlier TLS versions, though use of earlier versions is strongly discouraged [RFC8996] [RFC9325]. This format can also be used with DTLS [DTLS13], QUIC [RFC9000] [RFC9001], and other protocols that use the TLS key schedule. Use of this format could complement other protocol-specific logging such as qlog [QLOG].

This document also defines labels that can be used to log information about exchanges that use Encrypted Client Hello (ECH) [ECH].

1.1. Applicability Statement

The artifact that this document describes -- if made available to entities other than endpoints -- completely undermines the core guarantees that TLS provides. This format is intended for use in systems where TLS only protects test data. While the access that this information provides to TLS connections can be useful for diagnosing problems while developing systems, this mechanism **MUST NOT** be used in a production system. For software that is compiled, use of conditional compilation is the best way to ensure that deployed binaries cannot be configured to enable key logging.

Section 3 addresses a number of additional concerns that arise from the use of key logging.

1.2. Conventions

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.

2. The SSLKEYLOGFILE Format

A file in SSLKEYLOGFILE format is a text file. This document specifies the character encoding as UTF-8 [RFC3629]. Though the format itself only includes ASCII characters [RFC0020], comments **MAY** contain other characters. Though Unicode is permitted in comments, the file **MUST NOT** contain a Unicode byte order mark (U+FEFF).

Lines are terminated using the line-ending convention of the platform on which the file is generated. Tools that process these files **MUST** accept CRLF (U+13 followed by U+10), CR (U+13), or LF (U+10) as a line terminator. Lines are ignored if they are empty or if the first character is an octothorpe character ('#', U+23). Other lines of the file each contain a single secret.

Implementations that record secrets to a file do so continuously as those secrets are generated.

Each secret is described using a single line composed of three values that are separated by a single space character (U+20). These values are:

label:
: The label identifies the type of secret that is being conveyed; see Section 2.1 for descriptions of the labels that are defined in this document.

client_random:
: The 32-byte value of the Random field from the ClientHello message that established the TLS connection. This value is encoded as 64 hexadecimal characters. In a log that can include secrets from multiple connections, this field can be used to identify a connection.

secret:
: The value of the identified secret for the identified connection. This value is encoded in hexadecimal, with a length that depends on the size of the secret.

For the hexadecimal values of `client_random` or `secret`, no convention exists for the case of characters "a" through "f" (or "A" through "F"). Files can be generated with either, so either form **MUST** be accepted when processing a file.

Diagnostic tools that accept files in this format might choose to ignore lines that do not conform to this format in the interest of ensuring that secrets can be obtained from corrupted files.

Logged secret values are not annotated with the cipher suite or other connection parameters. Therefore, a record of the TLS handshake might be needed to use the logged secrets.

2.1. Secret Labels for TLS 1.3

An implementation of TLS 1.3 produces a number of values as part of the key schedule (see Section 7.1 of [TLS13]). If ECH was successfully negotiated for a given connection, these labels **MUST** be followed by the Random from the Inner ClientHello. Otherwise, the Random from the Outer ClientHello **MUST** be used.

Each of the following labels correspond to the equivalent secret produced by the key schedule:

CLIENT_EARLY_TRAFFIC_SECRET:
: This secret is used to protect records sent by the client as early data, if early data is attempted by the client. Note that a server that rejects early data will not log this secret, though a client that attempts early data can do so unconditionally.

EARLY_EXPORTER_SECRET:
: This secret is used for early exporters. Like CLIENT_EARLY_TRAFFIC_SECRET, this is only generated when early data is attempted and might not be logged by a server if early data is rejected.

CLIENT_HANDSHAKE_TRAFFIC_SECRET:
: This secret is used to protect handshake records sent by the client.

SERVER_HANDSHAKE_TRAFFIC_SECRET:
: This secret is used to protect handshake records sent by the server.

CLIENT_TRAFFIC_SECRET_0:
: This secret is used to protect application_data records sent by the client immediately after the handshake completes. This secret is identified as `client_application_traffic_secret_0` in the TLS 1.3 key schedule.

SERVER_TRAFFIC_SECRET_0:
: This secret is used to protect application_data records sent by the server immediately after the handshake completes. This secret is identified as `server_application_traffic_secret_0` in the TLS 1.3 key schedule.

EXPORTER_SECRET:
: This secret is used in generating exporters (Section 7.5 of [TLS13]).

These labels all appear in uppercase in the key log, but they correspond to lowercase labels in the TLS key schedule (Section 7.1 of [TLS13]), except for the application data secrets as noted. For example, "EXPORTER_SECRET" in the log file corresponds to the secret named `exporter_secret`.

Note that the order that labels appear here corresponds to the order in which they are presented in [TLS13], but there is no guarantee that implementations will log secrets strictly in this order.

2.2. Secret Labels for TLS 1.2

Implementations of TLS 1.2 [TLS12] (and also earlier versions) use the label "CLIENT_RANDOM" to identify the "master" secret for the connection.

2.3. Secret Labels for ECH

With ECH [ECH], additional secrets are derived during the handshake to encrypt the Inner ClientHello message using Hybrid Public Key Encryption (HPKE) [HPKE]. A client can log the ECH labels described below if it offered ECH, regardless of server acceptance. The server can log the labels only if it successfully decrypted the ECH offered by the client, though it could choose to do so only when it accepts ECH.

These labels **MUST** always use the Random from the Outer ClientHello.

ECH_SECRET:
: This label corresponds to the Key Encapsulation Mechanism (KEM) shared secret used by HPKE (`shared_secret` in the algorithms in Section 5.1.1 of [HPKE]). The length of the secret is defined by the KEM negotiated for use with ECH.

ECH_CONFIG:
: The ECHConfig used to construct the ECH extension. The value is logged in hexadecimal representation.

3. Security Considerations

Access to the content of a file in SSLKEYLOGFILE format allows an attacker to break the confidentiality and integrity protection on any TLS connections that are included in the file. This includes both active connections and connections for which encrypted records were previously stored. Therefore, ensuring adequate access control on these files becomes very important.

Implementations that support logging this data need to ensure that logging can only be enabled by those who are authorized. Allowing logging to be initiated by any entity that is not otherwise authorized to observe or modify the content of connections for which secrets are logged could represent a privilege escalation attack. Implementations that enable logging also need to ensure that access to logged secrets is limited, using appropriate file permissions or equivalent access control mechanisms.

In order to support decryption, the secrets necessary to remove record protection are logged. However, as the keys that can be derived from these secrets are symmetric, an adversary with access to these secrets is also able to encrypt data for an active connection. This might allow for injection or modification of application data on a connection that would otherwise be protected by TLS.

As some protocols rely on TLS for generating encryption keys, the SSLKEYLOGFILE format includes keys that identify the secret used in TLS exporters or early exporters (Section 7.5 of [TLS13]). Knowledge of these secrets can enable more than inspection or modification of encrypted data, depending on how an application protocol uses exporters. For instance, exporters might be used for session bindings (e.g., [RFC8471]), authentication (e.g., [RFC9261]), or other derived secrets that are used in application context. An adversary that obtains these secrets might be able to use this information to attack these applications. A TLS implementation might either choose to omit these secrets in contexts where the information might be abused or require separate authorization to enable logging of exporter secrets.

Using an environment variable, such as `SSLKEYLOGFILE`, to enable logging implies that access to the launch context for the application is needed to authorize logging. On systems that support specially named files, logs might be directed to these names so that logging does not result in storage but enables consumption by other programs. In both cases, applications might require special authorization or might rely on system-level access control to limit access to these capabilities.

Forward secrecy guarantees provided in TLS 1.3 (see Section 1.2 and Appendix E.1 of [RFC8446]) and some modes of TLS 1.2 (such as those in Sections 2.2 and 2.4 of [RFC4492]) do not hold if key material is recorded. Access to key material allows an attacker to decrypt data exchanged in any previously logged TLS connections.

Logging the TLS 1.2 "master" secret provides the recipient of that secret far greater access to an active connection than TLS 1.3 secrets provide. In addition to reading and altering protected messages, the TLS 1.2 "master" secret confers the ability to resume the connection and impersonate either endpoint, insert records that result in renegotiation, and forge Finished messages. Implementations can avoid the risks associated with these capabilities by not logging this secret value.

Access to the ECH_SECRET record in SSLKEYLOGFILE allows the attacker to decrypt the ECH extension and thereby reveal the content of the Inner ClientHello message, including the payload of the Server Name Indication (SNI) extension.

Access to the HPKE-established shared secret used in ECH introduces a potential attack surface against the HPKE library since access to this keying material is normally not available otherwise.

4. IANA Considerations

This document registers a media type (Section 4.1) and creates a registry for labels (Section 4.2).

4.1. SSLKEYLOGFILE Media Type

The "`application/sslkeylogfile`" media type can be used to describe content in the SSLKEYLOGFILE format. IANA has added the following information to the "Media Types" registry at <https://www.iana.org/assignments/media-types>:

Type name: application

Subtype name: sslkeylogfile

Required parameters: N/A

Optional parameters: N/A

Encoding considerations: UTF-8 without BOM, or ASCII only

Security considerations: See Section 3.

Interoperability considerations: Line endings might differ from platform convention.

Published specification: RFC 9850

Applications that use this media type: Diagnostic and analysis tools that need to decrypt data that is otherwise protected by TLS.

Fragment identifier considerations: N/A

Additional information:

Deprecated alias names for this type: N/A
Magic number(s): N/A
File extension(s): N/A
Macintosh file type code(s): N/A

Person & email address to contact for further information:
TLS WG (tls@ietf.org)

Intended usage: COMMON

Restrictions on usage: N/A

Author: IETF TLS Working Group

Change controller: IETF

4.2. TLS SSLKEYLOGFILE Labels Registry

IANA has created a new registry named "TLS SSLKEYLOGFILE Labels" within the existing "Transport Layer Security (TLS) Parameters" registry group. This new registry reserves labels used for SSLKEYLOGFILE entries. The initial contents of this registry are as follows:

Value	Description	Reference
CLIENT_RANDOM	Master secret in TLS 1.2 and earlier	RFC 9850
CLIENT_EARLY_TRAFFIC_SECRET	Secret for client early data records	RFC 9850
EARLY_EXPORTER_SECRET	Early exporters secret	RFC 9850
CLIENT_HANDSHAKE_TRAFFIC_SECRET	Secret protecting client handshake	RFC 9850
SERVER_HANDSHAKE_TRAFFIC_SECRET	Secret protecting server handshake	RFC 9850
CLIENT_TRAFFIC_SECRET_0	Secret protecting client records post handshake	RFC 9850
SERVER_TRAFFIC_SECRET_0	Secret protecting server records post handshake	RFC 9850
EXPORTER_SECRET	Exporter secret after handshake	RFC 9850
ECH_SECRET	HPKE KEM shared secret used in the ECH	RFC 9850
ECH_CONFIG	ECHConfig used for construction of the ECH	RFC 9850

Table 1

New assignments in the "TLS SSLKEYLOGFILE Labels" registry will be administered by IANA through the Specification Required procedure [RFC8126]. The role of the designated expert is described in Section 17 of [RFC8447]. The designated expert [RFC8126] ensures that the specification is publicly available. In the Reference column, it is sufficient to cite an Internet-Draft (that is posted but not published as an RFC) or a document from another standards body, an industry consortium, or any other location. The designated expert may provide more in-depth reviews, but their approval should not be taken as an endorsement of the SSLKEYLOGFILE label.

5. References

5.1. Normative References

[ECH] Rescorla, E., Oku, K., Sullivan, N., and C. A. Wood, "TLS Encrypted Client Hello", RFC 9849, DOI 10.17487/RFC9849, December 2025, <https://www.rfc-editor.org/info/rfc9849>.

[HPKE] Barnes, R., Bhargavan, K., Lipp, B., and C. Wood, "Hybrid Public Key Encryption", RFC 9180, DOI 10.17487/RFC9180, February 2022, <https://www.rfc-editor.org/info/rfc9180>.

[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>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <https://www.rfc-editor.org/info/rfc3629>.

[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>.

[TLS12] 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>.

[TLS13] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 9846, DOI 10.17487/RFC9846, December 2025, <https://www.rfc-editor.org/info/rfc9846>.

5.2. Informative References

[DTLS13] Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", RFC 9147, DOI 10.17487/RFC9147, April 2022, <https://www.rfc-editor.org/info/rfc9147>.

[QLOG] Marx, R., Ed., Niccolini, L., Ed., Seemann, M., Ed., and L. Pardue, Ed., "qlog: Structured Logging for Network Protocols", Work in Progress, Internet-Draft, draft-ietf-quic-qlog-main-schema-13, 20 October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-quic-qlog-main-schema-13>.

[RFC0020] Cerf, V., "ASCII format for network interchange", STD 80, RFC 20, DOI 10.17487/RFC0020, October 1969, <https://www.rfc-editor.org/info/rfc20>.

[RFC4492] Blake-Wilson, S., Bolyard, N., Gupta, V., Hawk, C., and B. Moeller, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)", RFC 4492, DOI 10.17487/RFC4492, May 2006, <https://www.rfc-editor.org/info/rfc4492>.

[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>.

[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>.

[RFC8447] Salowey, J. and S. Turner, "IANA Registry Updates for TLS and DTLS", RFC 8447, DOI 10.17487/RFC8447, August 2018, <https://www.rfc-editor.org/info/rfc8447>.

[RFC8471] Popov, A., Ed., Nystroem, M., Balfanz, D., and J. Hodges, "The Token Binding Protocol Version 1.0", RFC 8471, DOI 10.17487/RFC8471, October 2018, <https://www.rfc-editor.org/info/rfc8471>.

[RFC8792] Watsen, K., Auerswald, E., Farrel, A., and Q. Wu, "Handling Long Lines in Content of Internet-Drafts and RFCs", RFC 8792, DOI 10.17487/RFC8792, June 2020, <https://www.rfc-editor.org/info/rfc8792>.

[RFC8996] Moriarty, K. and S. Farrell, "Deprecating TLS 1.0 and TLS 1.1", BCP 195, RFC 8996, DOI 10.17487/RFC8996, March 2021, <https://www.rfc-editor.org/info/rfc8996>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/info/rfc9000>.

[RFC9001] Thomson, M., Ed. and S. Turner, Ed., "Using TLS to Secure QUIC", RFC 9001, DOI 10.17487/RFC9001, May 2021, <https://www.rfc-editor.org/info/rfc9001>.

[RFC9261] Sullivan, N., "Exported Authenticators in TLS", RFC 9261, DOI 10.17487/RFC9261, July 2022, <https://www.rfc-editor.org/info/rfc9261>.

[RFC9325] Sheffer, Y., Saint-Andre, P., and T. Fossati, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 9325, DOI 10.17487/RFC9325, November 2022, <https://www.rfc-editor.org/info/rfc9325>.

Appendix A. Example

The following is a sample of a file in SSLKEYLOGFILE format, including secrets from two TLS 1.3 connections.

```
# NOTE: '\' line wrapping per RFC 8792

CLIENT_HANDSHAKE_TRAFFIC_SECRET \
  cf34899b3dcb8c9fe7160ceaf95d354a294793b67a2e49cb9cca4d69b43593a0 \
  be4a28d81ce41242ff31c6d8a6615852178f2cd75eaca2ee8768f9ed51282b38
SERVER_HANDSHAKE_TRAFFIC_SECRET \
  cf34899b3dcb8c9fe7160ceaf95d354a294793b67a2e49cb9cca4d69b43593a0 \
  258179721fa704e2f1ee16688b4b0419967ddea5624cd5ad0863288dc5ead35f
CLIENT_HANDSHAKE_TRAFFIC_SECRET \
  b2eb93b8ddab8c228993567947bca1e133736980c22754687874e3896f7d6d0a \
  59ec0981b211a743f22d5a46a1fc77a2b230e16ef0de6d4e418abfe90eff10bf
SERVER_HANDSHAKE_TRAFFIC_SECRET \
  b2eb93b8ddab8c228993567947bca1e133736980c22754687874e3896f7d6d0a \
  a37fe4d3b6c9a6a372396b1562f6f8a40c1c3f85f1aa9b02d5ed46c4a1301365
CLIENT_TRAFFIC_SECRET_0 \
  cf34899b3dcb8c9fe7160ceaf95d354a294793b67a2e49cb9cca4d69b43593a0 \
  e9ca165bcb762fab8086068929d26c532e90ef2e2daa762d8b52346951a34c02
SERVER_TRAFFIC_SECRET_0 \
  cf34899b3dcb8c9fe7160ceaf95d354a294793b67a2e49cb9cca4d69b43593a0 \
  4f93c61ac1393008d4c820f3723db3c67494f06574b65fcc21c9eef22f90071a
EXPORTER_SECRET \
  cf34899b3dcb8c9fe7160ceaf95d354a294793b67a2e49cb9cca4d69b43593a0 \
  011c900833468f837f7c55d836b2719beebd39b1648fdeda58772f48d94a1ffa
CLIENT_TRAFFIC_SECRET_0 \
  b2eb93b8ddab8c228993567947bca1e133736980c22754687874e3896f7d6d0a \
  e9160bca1a531d871f5ecf51943d8cfb88833adeccf97701546b5fb93e030d79
SERVER_TRAFFIC_SECRET_0 \
  b2eb93b8ddab8c228993567947bca1e133736980c22754687874e3896f7d6d0a \
  fb1120b91e48d402fac20faa33880e77bace82c85d6688df0aa99bf5084430e4
EXPORTER_SECRET \
  b2eb93b8ddab8c228993567947bca1e133736980c22754687874e3896f7d6d0a \
  db1f4fa1a6942fb125d4cc47e02938b6f8030c6956bb81b9e3269f1cf855a8f8
```

Note that secrets from the two connections might be interleaved as shown here, because secrets could be logged as they are generated.

The following shows a log entry for a TLS 1.2 connection.

```
# NOTE: '\' line wrapping per RFC 8792

CLIENT_RANDOM \
  ad52329fcadd34ee3aa07092680287f09954823e26d7b5ae25c0d47714152a6a \
  97af4c8618cfdc0b2326e590114c2ec04b43b08b7e2c3f8124cc61a3b068ba966\
  9517e744e3117c3ce6c538a2d88dfdf
```

The following shows a log entry for a TLS 1.3 connection that successfully negotiated ECH.

```
# NOTE: '\' line wrapping per RFC 8792

ECH_SECRET \
  0ba587ee6b65ce21a726630efb881206a7cd995611095b5f4c244bb2b23f1ee1 \
  e8828ec09909cc9363179dc13b62498550c8637129345263011a1678370ca52a
ECH_CONFIG \
  0ba587ee6b65ce21a726630efb881206a7cd995611095b5f4c244bb2b23f1ee1 \
  fe0d003c5500200020d5260ae4cdda08bcbdc37bd0dc53c29aea5f0fdd2b2d594\
  e4235e99b134ac904000400010001000d636f7665722e6465666f2e69650000
CLIENT_HANDSHAKE_TRAFFIC_SECRET \
  8726180bb24718089a4c5c8c93e0ea1c6d6649d7dd3c978fc1413854a20e9647 \
  a195b63ec4270609692a204c08e63e74d9ae58e377d11a383bfe641a63c01140
SERVER_HANDSHAKE_TRAFFIC_SECRET \
  8726180bb24718089a4c5c8c93e0ea1c6d6649d7dd3c978fc1413854a20e9647 \
  022d1cb827a90f27dadde0c99110c2b7d0f362fdfe420a04818aa223e5f2c14c
CLIENT_TRAFFIC_SECRET_0 \
  8726180bb24718089a4c5c8c93e0ea1c6d6649d7dd3c978fc1413854a20e9647 \
  c2310f7db71109de88bab6f2f433fdc1704aecc0d57349cbf9113e5033178172
SERVER_TRAFFIC_SECRET_0 \
  8726180bb24718089a4c5c8c93e0ea1c6d6649d7dd3c978fc1413854a20e9647 \
  04ffc7c154f71ba5f530c7344b0496f60ce71b9b7c6b0e203ea574bfcdf14e27
EXPORTER_SECRET \
  8726180bb24718089a4c5c8c93e0ea1c6d6649d7dd3c978fc1413854a20e9647 \
  befb5db5ac6785b5dd4c6a8c4693c379ec0a1486b5fd035b25e50c3c95abc500
```

Acknowledgments

The SSLKEYLOGFILE format originated in the Network Security Services (NSS) project, but it has evolved over time as TLS has changed. Many people contributed to this evolution. The authors are only documenting the format as it is used while extending it to cover ECH.

Authors' Addresses

Martin Thomson
Mozilla
Email: mt@lowentropy.net

Yaroslav Rosomakho
Zscaler
Email: yrosomakho@zscaler.com

Hannes Tschofenig
University of Applied Sciences Bonn-Rhein-Sieg
Email: Hannes.Tschofenig@gmx.net