Abstract

The TLS protocol is widely deployed and used over the Internet. Client and server nodes compute a set of keys called the key-block, according to a pseudo random function (PRF). This draft proposes a keying infrastructure based on the TLS protocol. It suggests defining an additional Key Distribution Function (KDF) in order to deliver a set of cryptographic keys. In a peer to peer mode keys are directly produced as inputs of the KDF functions. For centralized architectures they are delivered through containers, secured with keys derived from the KDF function.
Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119.

Table of Contents

Copyright Notice................................................... 1
Abstract........................................................................ 1
Conventions used in this document................................. 2
1 Introduction......................................................... 3
2 Basic Exchanges..................................................... 4
   2.1 Full handshake............................................... 4
   2.2 Abbreviated Handshake.................................... 4
   2.3 AVP Container............................................. 5
   2.4 Data encryption and integrity............................ 6
3 Key Derivation Functions (KDF)................................. 6
   3.1 TLS PRF function.......................................... 6
   3.2 HMAC KDF................................................ 7
   3.3 NIST compatible KDF..................................... 7
      3.3.1 PRF functions .................................... 7
      3.3.2 KDF functions .................................... 8
4 Use cases.................................................................... 8
   4.1 Hierarchical Model for Content/Service Protection...... 8
5 IANA Considerations................................................ 9
6 Security Considerations.......................................... 9
   7.1 Normative References...................................... 9
   7.2 Informative References................................... 9
Authors’ Addresses.................................................. 10
Copyright Statement................................................ 10
Acknowledgment..................................................... 10
1 Introduction

The TLS protocol [TLS 1.0] [TLS 1.1] [TLS 1.2] is widely deployed and used over the Internet. Client and server nodes compute a set of keys called the key-block, according to a pseudo random function (PRF) and two random values client-random and server-random, respectively produced by these entities.

There is an increasing need in the Internet to set up efficient keys distribution infrastructures, able to generate and to deliver cryptographic material required by multiple applications, as illustrated by section 6.

This draft proposes a keying infrastructure based on the TLS protocol.

Full or abbreviated handshakes are performed according to the TLS specification.

We suggest defining additional Key Distribution Functions (KDFi) in order to deliver a set of cryptographic keys according to the relation,

\[ \text{Keys = KDFi(master-secret, client-random, server-random, other-information)} \]

Particular KDF MAY be negotiated by the client according to specific TLS extensions [TLS-EXT] or dedicated cipher suites.

In a peer to peer mode keys are directly produced as inputs of the KDF functions, this mode of operation is for example quite suitable for distributed architecture, in which networks nodes are typically equipped with certificates and RSA keys.

On the other hand, centralized architectures deliver keys from a unique server to remotely managed nodes. In this later case cryptographic keys MUST be delivered through application data, according to cipher suites negotiated during the TLS handshake, but working with keys computed with the KDF function.
2 Basic Exchanges

2.1 Full Handshake

The full handshake mode is a four ways handshake that performs one-way or mutual authentication between client and server entities. A master secret is computed by both parties, and each of them proves its knowledge of such value by Finished messages, which are protected according to cryptographic algorithms negotiated between client and server, and identified by the CipherSuite parameter.

\[
\begin{align*}
\text{Client} & \quad \text{Server} \\
(A) \quad \text{ClientHello} & \quad --------> \quad \text{ServerHello} \\
& \quad \text{Certificate*} \\
& \quad \text{ServerKeyExchange*} \\
& \quad \text{CertificateRequest*} \\
& \quad <-------- (B) \quad \text{ServerHelloDone} \\
(C) \quad \text{Certificate*} \quad \text{CertificateVerify*} \\
& \quad \text{ClientKeyExchange} \quad \text{[ChangeCipherSpec]} \\
& \quad \text{Finished} \quad --------> \quad \text{[ChangeCipherSpec]} \\
& \quad <-------- (D) \quad \text{Finished} \\
& \quad <-------- (E) \quad *AVP Container
\end{align*}
\]

Figure 1. Full handshake

In the peer to peer mode, keys are computed according to the selected KDF procedure and its associated information:

\[
\text{Keys} = \text{KDF(master-secret, client-random, server-random, other-information)}
\]

In centralized architecture, an AVP attribute conveys a set of keys, protected according to cryptographic algorithms pointed by the CipherSuite parameter, but dealing with keys produced by the KDF function.

2.2 Abbreviated Handshake

The abbreviated handshake mode is a three ways handshake that performs a mutual authentication between client and server entities, which share the knowledge of a previous master secret. Each of them proves this assumption by Finished messages, which are protected.
according to cryptographic algorithms negotiated between client and server, and identified by the CipherSuite parameter.

Client                      Server

(A) ClientHello   -------->  ServerHello
                        [ChangeCipherSpec]
                        <--------  (B)   Finished

(C) [ChangeCipherSpec]
    Finished   -------->  (D) *AVPs Container

*Optional

Figure 2. Abbreviated handshake

In the peer to peer mode, keys are computing according to the relation:

Keys = KDF(master-secret, client-random, server-random, other-information)

In centralized architecture, an AVP attributes conveys a set of keys, protected according to cryptographic algorithms pointed by the CipherSuite parameter, but using keys produced by the KDF function.

2.3 AVP Container

The AVP format is imported from the [TTLS] document.

The AVP Code is four octets and, combined with the Vendor-ID field if present, identifies the attribute (i.e. the container structure) uniquely.
The 'V' (Vendor-Specific) bit indicates whether the optional Vendor-ID field is present.

The 'M' (Mandatory) bit indicates whether support of the AVP is required. If this bit is set to 0, this indicates that the AVP may be safely ignored if the receiving party does not understand or support it.

The 'r' (reserved) bits are unused and MUST be set to 0 by the sender and MUST be ignored by the receiver.

The AVP Length field is three octets, and indicates the length of this AVP including the AVP Code, AVP Length, AVP Flags, Vendor-ID (if present) and Data.

Data Length is two octets and indicates the size of data without padding bytes. According to selected cryptographic algorithms padding byte MAY be added in order to get a length compatible with ciphered blocs.

2.4 Data encryption and integrity

Cryptographic keys (Kc and Ki), used by encryption algorithms (Kc) and MAC procedures (Ki), are derived according to TLS specifications, by use KDF in place of the TLS PRF function.

A message M is concatenated to L null bytes, where L is the output length (in bytes) of the HMAC procedure, and optional padding bytes. If a cipher bloc algorithm is used, then the total length MUST be a multiple of the bloc size.

\[ M_1 = M \ || \ 0:L \ || \ Pad^* \]

A HMAC is computed over \( M_1 \)

\[ MAC = HMAC(Ki, M_1) \]

A M2 message is built, \( M_2 = M \ || \ MAC \ || \ Pad^* \)

This M2 message is encrypted in order to produce the data field

\[ Data = \{M2\}Kc \]

3 Key Derivation Functions (KDF)

3.1 TLS PRF function

In [TLS extractor] it is suggested to use the TLS PRF function, according to the relation:
KDF = PRF(master_secret, label,
            client-random || server-random ||
            context-value-length || context-value)

3.2 HMAC KDF

We suggest using a HMAC based instantiation for KDF, as introduced in [HMAC KDF].

The HMAC procedure is identical to the MAC algorithm negotiated during the handshake session (e.g. HMAC-MD5, HMAC-SHA1, HMAC-SHA256).

Notations.

- The first argument to a keyed function denotes the key, the value \( K \) is the key to PRF and \( x \) its input.
- The symbol || denotes concatenation.
- Given two numbers \( N \) and \( n \) the symbol \( N:n \) represents the value \( N \) written as \( n \)-bit integer.
- \( L \) is the length in bits, of this output value delivers by the HMAC procedure

Pseudo Random Key = PRK = HMAC(client-random || server-random,
master-secret)

Keying material, whose length in bits is \( D \), required \( D/L \) operations.

First is expressed as

\[
K(1) = \text{HMAC}(PRK, 0:L || KeyLabel || 0:32),
\]

Further operations (whose number is \( i \)) are computed according to

\[
K(i+1) = \text{HMAC}(PRK, K(i)|| KeyLabel || i:32),
\]

where KeyLabel is an ASCII string.

3.3 NIST compatible KDF

3.3.1 PRF functions

The [NIST KDF] standard proposes two candidates for PRF functions:

- \( \text{PRF} = \text{HMAC}(s,x) \), and
- \( \text{PRF} = \text{CMAC}(s,x) \)

Where \( s \) is a seed (a secret key in this draft context) and \( x \) an input data (a stream of bytes)
We suggest to compute $s$ according to [HMAC KDF]

$$s = PRK = HMAC(\text{client-random} || \text{server-random}, \text{master-secret})$$

3.3.2 KDF functions

Two methods are proposed by the [NIST KDF] standard:

- KDF in counter mode, and
- KDF in feedback mode

KDF functions are based on PRF using the PRK key and additional data such as

- The length ($L$) of the generated cryptographic material,
- The Label (a binary string) of the derived keying material,
- The Context (a binary string) of the derived keying material (for example identities of parties who want to compute/use this key)

$$KDF = PRF(\text{PRK}, f(L, \text{Label}, \text{context}))$$

4 Use cases

4.1 Hierarchical Model for Content/Service Protection

The ETSI 102 474 standard [TS 102.474] introduces a hierarchical model for content/service protection mechanisms, targeting digital IP datacast over video broadcasting. This scheme is a nice illustration of a centralized architecture, in which cryptographic keys are securely pushed from head-ends towards terminal devices.

Each terminal device holds a Right Encryption Key (REK).

This cryptographic material is used in conjunction with Keys Management protocol (KMM) in order to push from head-ends to terminals Services Rights Keys (SEK).

In turn the Key Stream Message (KSM) protocol, associated to the SEK key, transport Traffic Encryption Keys (TEK) from head-ends to terminals.

At last, clear content is encrypted with TEK, and is broadcasted from head-ends to terminals

As an illustration the TLS protocol MAY be used in order to collect the SEK keys. The REK cryptographic material can be instantiated either by a terminal certificate (authenticated by a Certificate-Verify message) or by a pre-shared key [RFC 4279]. A SEK key is securely pushed and transported in an AVP container.
5 IANA Considerations

6 Security Considerations

7 References

7.1 Normative References


7.2 Informative References


Authors’ Addresses

Pascal Urien
Telecom ParisTech
37/39 rue Dareau, 75014 Paris, France

Email: Pascal.Urien@enst.fr

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Copyright Statement

Copyright (C) The IETF Trust (2008). This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

Acknowledgment
Funding for the RFC Editor function is currently provided by the Internet Society.