          Datagram Transport Layer Security (DTLS) over the Datagram
                   Congestion Control Protocol (DCCP)

Status of This Memo

   This document specifies an Internet standards track protocol for the
   Internet community, and requests discussion and suggestions for
   improvements. Please refer to the current edition of the "Internet
   Official Protocol Standards" (STD 1) for the standardization state
   and status of this protocol. Distribution of this memo is unlimited.

Abstract

   This document specifies the use of Datagram Transport Layer Security
   (DTLS) over the Datagram Congestion Control Protocol (DCCP). DTLS
   provides communications privacy for applications that use datagram
   transport protocols and allows client/server applications to
   communicate in a way that is designed to prevent eavesdropping and
   detect tampering or message forgery. DCCP is a transport protocol
   that provides a congestion-controlled unreliable datagram service.

Table of Contents

1. Introduction

   This document specifies how to carry application payloads with
   Datagram Transport Layer Security (DTLS), as specified in [RFC4347],
   in the Datagram Congestion Control Protocol (DCCP), as specified in
   [RFC4340].

   DTLS is an adaptation of Transport Layer Security (TLS, [RFC4346])
   that modifies TLS for use with the unreliable transport protocol UDP.
   TLS is a protocol that allows client/server applications to
   communicate in a way that is designed to prevent eavesdropping and
   detect tampering and message forgery. DTLS can be viewed as
   TLS-plus-adaptations-for-unreliability.

   DCCP provides an unreliable transport service, similar to UDP, but
   with adaptive congestion control, similar to TCP and Stream Control
   Transmission Protocol (SCTP). DCCP can be viewed equally well as
   either UDP-plus-congestion-control or TCP-minus-reliability
   (although, unlike TCP, DCCP offers multiple congestion control
   algorithms).

   The combination of DTLS and DCCP will offer transport security
   capabilities to applications using DCCP similar to those available
   for TCP, UDP, and SCTP.

2. Terminology

   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 [RFC2119].

3. DTLS over DCCP

   The approach here is very straightforward -- DTLS records are
   transmitted in the Application Data fields of DCCP-Data and
   DCCP-DataAck packets (in the rest of the document assume that
   "DCCP-Data packet" means "DCCP-Data or DCCP-DataAck packet").
   Multiple DTLS records MAY be sent in one DCCP-Data packet, as long as
   the resulting packet is within the Path Maximum Transfer Unit (PMTU)
   currently in force for normal data packets, if fragmentation is not
   allowed (the Don't Fragment (DF) bit is set for IPv4 or no
   fragmentation extension headers are being used for IPv6), or within
   the current DCCP maximum packet size if fragmentation is allowed (see
   Section 3.5 for more information on PMTU Discovery). A single DTLS
   record MUST be fully contained in a single DCCP-Data packet; it MUST
   NOT be split over multiple packets.

3.1. DCCP and DTLS Sequence Numbers

   Both DCCP and DTLS use sequence numbers in their packets/records.
   These sequence numbers serve somewhat, but not completely,
   overlapping functions. Consequently, there is no connection between
   the sequence number of a DCCP packet and the sequence number in a
   DTLS record contained in that packet, and there is no connection
   between sequence number-related features such as DCCP synchronization
   and DTLS anti-replay protection.

3.2. DCCP and DTLS Connection Handshakes

   Unlike UDP, DCCP is connection-oriented, and has a connection
   handshake procedure that precedes the transmission of DCCP-Data and
   DCCP-DataAck packets. DTLS is also connection-oriented, and has a
   handshake procedure of its own that must precede the transmission of
   actual application information. Using the rule of mapping DTLS
   records to DCCP-Data and DCCP-DataAck packets in Section 3, above,
   the two handshakes are forced to happen in series, with the DCCP
   handshake first, followed by the DTLS handshake. This is how TLS
   over TCP works.

   However, the DCCP handshake packets DCCP-Request and DCCP-Response
   have Application Data fields and can carry user data during the DCCP
   handshake, and this creates the opportunity to perform the handshakes
   partially in parallel. DTLS client implementations MAY choose to
   transmit one or more DTLS records (typically containing DTLS
   handshake messages or parts of them) in the DCCP-Request packet. A
   DTLS server implementation MAY choose to process these records as
   usual, and if it has one or more DTLS records to send as a response
   (typically containing DTLS handshake messages or parts of them), it
   MAY include those records in the DCCP-Response packet. DTLS servers
   MAY also choose to delay the response until the DCCP handshake
   completes and then send the DTLS response in a DCCP-Data packet.

   Note that even though the DCCP handshake is a reliable process (DCCP
   handshake messages are retransmitted as required if messages are
   lost), the transfer of Application Data in DCCP-Request and
   DCCP-Response packets is not necessarily reliable. For example, DCCP
   server implementations are free to discard Application Data received
   in DCCP-Request packets. And if DCCP-Request or DCCP-Response
   packets need to be retransmitted, the DCCP implementation may choose
   to not include the Application Data present in the initial message.

Since the DTLS handshake is also a reliable process, it will
interoperate across the data delivery unreliability of DCCP (after
all, one of the basic functions of DTLS is to work over unreliable
transport). If the DTLS records containing DTLS handshake messages
are lost, they will be retransmitted by DTLS.

This is regardless of whether the messages were sent in
DCCP-Response/Request packets or DCCP-Data packets. However, the
only way for DTLS to retransmit DTLS records that were originally
transmitted in DCCP-Request/Response packets (and they or the
responses were lost somehow) is to wait for the DCCP handshake to
complete and then resend the records in DCCP-Data packets. This is
due to the characteristic of DCCP that the next opportunity to send
data after sending data in a DCCP-Request is only after the
connection handshake completes.

DCCP and DTLS use similar strategies for retransmitting handshake
messages. If there is no response to the original request
(DCCP-Request or any DTLS handshake message where a response is
expected) within normally 1 second, the message is retransmitted.
The timer is then doubled and the process repeated until a response
is received, or a maximum time is exceeded.

Therefore, if DTLS records are sent in a DCCP-Request packet, and the
DCCP-Request or DCCP-Response message is lost, the DCCP and DTLS
handshakes could be timing out on similar schedules. The
DCCP-Request packets will be retransmitted on timeout, but the DTLS
records cannot be retransmitted until the DCCP handshake completes
(there is no possibility of adding new Application Data to a
DCCP-Request retransmission). In order to avoid multiple DTLS
retransmissions queuing up before the first retransmission can be
sent, DTLS over DCCP MUST wait until the completion of the DCCP
handshake before restarting its DTLS handshake retransmission timer.

3.3. Effects of DCCP Congestion Control

Given the large potential sizes of the DTLS handshake messages, it is
possible that DCCP congestion control could throttle the transmission
of the DTLS handshake to the point that the transfer cannot complete
before the DTLS timeout and retransmission procedures take effect.
Adding retransmitted messages to a congested situation might only
make matters worse and delay connection establishment.

Note that a DTLS over UDP application transmitting handshake data
into this same network situation will not necessarily receive better
throughput, and might actually see worse effective throughput.

Without the pacing of slow-start and congestion control, a UDP
application might be making congestion worse and lowering the
effective throughput it receives.

As stated in [RFC4347], "mishandling of the [retransmission] timer
can lead to serious congestion problems". This remains as true for
DTLS over DCCP as it is for DTLS over UDP.

DTLS over DCCP implementations SHOULD take steps to avoid
retransmitting a request that has been queued but not yet actually
transmitted by DCCP, when the underlying DCCP implementation can
provide this information. For example, DTLS could delay starting the
retransmission timer until DCCP indicates the message has been
transferred from DCCP to the IP layer.

In addition to the retransmission issues, if the throughput needs of
the actual application data differ from the needs of the DTLS
handshake, it is possible that the handshake transference could leave
the DCCP congestion control in a state that is not immediately
suitable for the application data that will follow. For example,
DCCP Congestion Control Identifier (CCID) 2 ([RFC4341]) congestion
control uses an Additive Increase Multiplicative Decrease (AIMD)
algorithm similar to TCP congestion control. If it is used, then it
is possible that transference of a large handshake could cause a
multiplicative decrease that would not have happened with the
application data. The application might then be throttled while
waiting for additive increase to return throughput to acceptable
levels.

Applications where this might be a problem should consider using DCCP
CCID 3 ([RFC4342]). CCID 3 implements TCP-Friendly Rate Control
(TFRC, [RFC3448])). TFRC varies the allowed throughput more slowly
than AIMD and might avoid the discontinuities possible with CCID 2.

3.4. Relationships between DTLS Sessions/Connections and DCCP
      Connections

DTLS uses the concepts of sessions and connections. A DTLS
connection is used by upper-layer endpoints to exchange data over a
transport protocol. DTLS sessions contain cached state information
that is used to reduce the number of roundtrips and computation
required to create multiple DTLS connections between the same
endpoints.

In DTLS over DCCP, a DTLS connection is carried by a DCCP connection.
Often the DCCP connection establishment is immediately followed by
DTLS connection establishment (either creating a new DTLS session
with full handshake, or resuming an existing DTLS session), and the
DTLS connection termination is immediately followed by DCCP
connection termination, but this is not the only possibility.

The life of a DTLS over DCCP connection is completely contained
within the life of the underlying DCCP connection; a DTLS connection
cannot continue if its underlying DCCP connection terminates.
However, multiple DTLS connections can be resumed from the same DTLS
session, each running over its own DCCP connection. The session
resumption features of DTLS are widely used, and this situation is
likely to occur in many use cases. It is also possible to resume a
DTLS session with a new DTLS connection running over a different
transport.

Note that it is possible for an application to start a DCCP
connection by transferring unprotected packets, and then switch to
DTLS after some time. This is likely to be useful for applications
that would like to negotiate using DTLS or not and has implications
for the choice of DCCP Service Code. See Section 3.6 for more
information.

Many DTLS Application Programming Interfaces (APIs) do not prevent an
application from sending a mix of encrypted and clear packets over
the same transport connection. Applications MUST NOT send
unprotected data on a DCCP connection while it is also carrying a
DTLS connection, since this presents a vulnerability to packet
insertion attacks.

Many DTLS APIs also allow an application to start multiple DTLS
connections over one transport connection in series, with the
termination of one DTLS connection followed by the start of another.
Processing a DTLS handshake is relatively CPU intensive. An
application that uses this strategy is open to an attacker that
repeatedly starts and immediately stops sessions. Therefore,
applications that use this strategy SHOULD limit the potential burden
on the system by some means. For example, the application could
enforce a minimum time of 1 second between session initiations.

3.5. PMTU Discovery

Each DTLS record must fit within a single DCCP-Data packet. DCCP
packets are normally transmitted with the DF (Don't Fragment) bit set
for IPv4 (or without fragmentation extension headers for IPv6).
Because of this, DCCP performs Path Maximum Transmission Unit (PMTU)
Discovery.

DTLS also normally uses the DF bit and performs PMTU Discovery on its
own, using an algorithm that is strongly similar to the one used by
DCCP. A DTLS over DCCP implementation MAY use the DCCP-managed value
for PMTU and not perform PMTU Discovery on its own. However,
implementations that choose to use the DCCP-managed PMTU value SHOULD
continue to follow the procedures of Section 4.1.1.1 of [RFC4347]
with regard to fragmenting handshake messages during handshake
retransmissions. Alternatively, a DTLS over DCCP implementation MAY
choose to use its own PMTU Discovery calculations, as specified in
[RFC4347], but MUST NOT use a value greater than the value determined
by DCCP.

DTLS implementations normally allow applications to reset the PMTU
estimate back to the initial state. When that happens, DTLS over
DCCP implementations SHOULD also reset the DCCP PMTU estimation.

DTLS implementations also sometimes allow applications to control the
use of the DF bit (when running over IPv4) or the use of
fragmentation extension headers (when running over IPv6). DTLS over
DCCP implementations SHOULD control the use of the DF bit or
fragmentation extension headers by DCCP in concert with the
application's indications, when the DCCP implementation supports
this. Note that DCCP implementations are not required to support
sending fragmentable packets.

Note that the DCCP Maximum Packet Size (MPS in [RFC4340]) is bounded
by the current congestion control state (Congestion Control Maximum
Packet Size, CCMPS in [RFC4340]). Even when the DF bit is not set
and DCCP packets may then be fragmented, the MPS may be less than the
65,535 bytes normally used in UDP. It is also possible for the DCCP
CCMPS, and thus the MPS, to vary over time as congestion conditions
change. DTLS over DCCP implementations MUST NOT use a DTLS record
size that is greater than the DCCP MPS currently in force.

3.6. DCCP Service Codes

The DCCP connection handshake includes a field called Service Code
that is intended to describe "the application-level service to which
the client application wants to connect". Further, "Service Codes
are intended to provide information about which application protocol
a connection intends to use, thus aiding middleboxes and reducing
reliance on globally well-known ports" [RFC4340].

It is expected that many middleboxes will give different privileges
to applications running DTLS over DCCP versus just DCCP. Therefore,
applications that use DTLS over DCCP sometimes and just DCCP other
times SHOULD register and use different Service Codes for each mode

of operation. Applications that use both DCCP and DTLS over DCCP MAY
choose to listen for incoming connections on the same DCCP port and
distinguish the mode of the request by the offered Service Code.

Some applications may start out using DCCP without DTLS, and then
optionally switch to using DTLS over the same connection. Since
there is no way to change the Service Code for a connection after it
is established, these applications will use one Service Code.

3.7. New Versions of DTLS

As DTLS matures, revisions to and updates for [RFC4347] can be
expected. DTLS includes mechanisms for identifying the version in
use, and presumably future versions will either include backward
compatibility modes or at least not allow connections between
dissimilar versions. Since DTLS over DCCP simply encapsulates the
DTLS records transparently, these changes should not affect this
document and the methods of this document should apply to future
versions of DTLS.

Therefore, in the absence of a revision to this document, this
document is assumed to apply to all future versions of DTLS. This
document will only be revised if a revision to DTLS or DCCP
(including its related CCIDs) makes a revision to the encapsulation
necessary.

It is RECOMMENDED that an application migrating to a new version of
DTLS keep the same DCCP Service Code used for the old version and
allow DTLS to provide the version negotiation support. If a new
version of DTLS provides significant new capabilities to the
application that could change the behavior of middleboxes with regard
to the application, an application developer MAY register a new
Service Code.

4. Security Considerations

Security considerations for DTLS are specified in [RFC4347] and for
DCCP in [RFC4340]. The combination of DTLS and DCCP introduces no
new security considerations.

5. Acknowledgments

The author would like to thank Eric Rescorla for initial guidance on
adapting DTLS to DCCP, and Gorry Fairhurst, Pasi Eronen, Colin
Perkins, Lars Eggert, Magnus Westerlund, and Tom Petch for comments
on the document.

6. References

6.1. Normative References

   [RFC2119]   Bradner, S., "Key words for use in RFCs to Indicate
               Requirement Levels", BCP 14, RFC 2119, March 1997.

   [RFC4340]   Kohler, E., Handley, M., and S. Floyd, "Datagram
               Congestion Control Protocol (DCCP)", RFC 4340, March 2006.

   [RFC4346]   Dierks, T. and E. Rescorla, "The Transport Layer Security
               (TLS) Protocol Version 1.1", RFC 4346, April 2006.

   [RFC4347]   Rescorla, E. and N. Modadugu, "Datagram Transport Layer
               Security", RFC 4347, April 2006.

6.2. Informative References

   [RFC3448]   Handley, M., Floyd, S., Padhye, J., and J. Widmer, "TCP
               Friendly Rate Control (TFRC): Protocol Specification", RFC
               3448, January 2003.

   [RFC4341]   Floyd, S. and E. Kohler, "Profile for Datagram Congestion
               Control Protocol (DCCP) Congestion Control ID 2: TCP-like
               Congestion Control", RFC 4341, March 2006.

   [RFC4342]   Floyd, S., Kohler, E., and J. Padhye, "Profile for
               Datagram Congestion Control Protocol (DCCP) Congestion
               Control ID 3: TCP-Friendly Rate Control (TFRC)", RFC 4342,
               March 2006.

Author's Address

   Tom Phelan
   Sonus Networks
   7 Technology Park Dr.
   Westford, MA USA 01886
   Phone: 978-614-8456
   Email: tphelan@sonusnet.com