IP Traffic Flow Security
draft-ietf-ipsecme-iptfs-00

Abstract

   This document describes a mechanism to enhance IPsec traffic flow
   security by adding traffic flow confidentiality to encrypted IP
   encapsulated traffic. Traffic flow confidentiality is provided by
   obscuring the size and frequency of IP traffic using a fixed-sized,
   constant-send-rate IPsec tunnel. The solution allows for congestion
   control as well.

Status of This Memo

   This Internet-Draft is submitted in full conformance with the
   provisions of BCP 78 and BCP 79.

   Internet-Drafts are working documents of the Internet Engineering
   Task Force (IETF). Note that other groups may also distribute
   working documents as Internet-Drafts. The list of current Internet-
   Drafts is at https://datatracker.ietf.org/drafts/current/.

   Internet-Drafts are draft documents valid for a maximum of six months
   and may be updated, replaced, or obsoleted by other documents at any
   time. It is inappropriate to use Internet-Drafts as reference
   material or to cite them other than as "work in progress."

   This Internet-Draft will expire on June 18, 2020.

Copyright Notice

Table of Contents

1. Introduction

   Traffic Analysis ([RFC4301], [AppCrypt]) is the act of extracting
   information about data being sent through a network. While one may
   directly obscure the data through the use of encryption [RFC4303],
   the traffic pattern itself exposes information due to variations in
   it's shape and timing ([I-D.iab-wire-image], [AppCrypt]). Hiding the
   size and frequency of traffic is referred to as Traffic Flow
   Confidentiality (TFC) per [RFC4303].

   [RFC4303] provides for TFC by allowing padding to be added to
   encrypted IP packets and allowing for transmission of all-pad packets
   (indicated using protocol 59). This method has the major limitation
   that it can significantly under-utilize the available bandwidth.

   The IP-TFS solution provides for full TFC without the aforementioned
   bandwidth limitation. To do this, we use a constant-send-rate IPsec
   [RFC4303] tunnel with fixed-sized encapsulating packets; however,
   these fixed-sized packets can contain partial, whole or multiple IP
   packets to maximize the bandwidth of the tunnel.

   For a comparison of the overhead of IP-TFS with the RFC4303
   prescribed TFC solution see Appendix C.

   Additionally, IP-TFS provides for dealing with network congestion
   [RFC2914]. This is important for when the IP-TFS user is not in full
   control of the domain through which the IP-TFS tunnel path flows.

1.1. Terminology & Concepts

   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
   [RFC2119] [RFC8174] when, and only when, they appear in all capitals,
   as shown here.

   This document assumes familiarity with IP security concepts described
   in [RFC4301].

2. The IP-TFS Tunnel

As mentioned in Section 1 IP-TFS utilizes an IPsec [RFC4303] tunnel
(SA) as it's transport. To provide for full TFC we send fixed-sized
encapsulating packets at a constant rate on the tunnel.

The primary input to the tunnel algorithm is the requested bandwidth
of the tunnel. Two values are then required to provide for this
bandwidth, the fixed size of the encapsulating packets, and rate at
which to send them.

The fixed packet size may either be specified manually or can be
determined through the use of Path MTU discovery [RFC1191] and
[RFC8201].

Given the encapsulating packet size and the requested tunnel
bandwidth, the corresponding packet send rate can be calculated. The
packet send rate is the requested bandwidth divided by the payload
size of the encapsulating packet.

The egress of the IP-TFS tunnel MUST allow for, and expect the
ingress (sending) side of the IP-TFS tunnel to vary the size and rate
of sent encapsulating packets, unless constrained by other policy.

2.1. Tunnel Content

As previously mentioned, one issue with the TFC padding solution in
[RFC4303] is the large amount of wasted bandwidth as only one IP
packet can be sent per encapsulating packet. In order to maximize
bandwidth IP-TFS breaks this one-to-one association.

With IP-TFS we aggregate as well as fragment the inner IP traffic
flow into fixed-sized encapsulating IPsec tunnel packets. We only
pad the tunnel packets if there is no data available to be sent at
the time of tunnel packet transmission, or if fragmentation has been
disabled by the receiver.

In order to do this we use a new Encapsulating Security Payload (ESP,
[RFC4303]) payload type which is the new IP protocol number
IPTFS_PROTOCOL (TBD1).

2.2. IPTFS_PROTOCOL Payload Content

The IPTFS_PROTOCOL ESP payload is comprised a 4 or 16 octet header
followed by either a partial, a full or multiple partial or full data
blocks. The following diagram illustrates the IPTFS_PROTOCOL ESP
payload within the ESP packet. See Section 6.1 for the exact formats
of the IPTFS_PROTOCOL payload.

```
  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
  . Outer Encapsulating Header ...                               .
  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
  . ESP Header...                                                .
   +---------------------------------------------------------------+
   |              ...            :            BlockOffset          |
   +---------------------------------------------------------------+
   :                  [Optional Congestion Info]                  :
   +---------------------------------------------------------------+
   |        DataBlocks ...                                         ~
   ~                                                               ~
   ~                                                               |
   +--------------------------------------------------------------|
   . ESP Trailer...                                               .
   . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
```
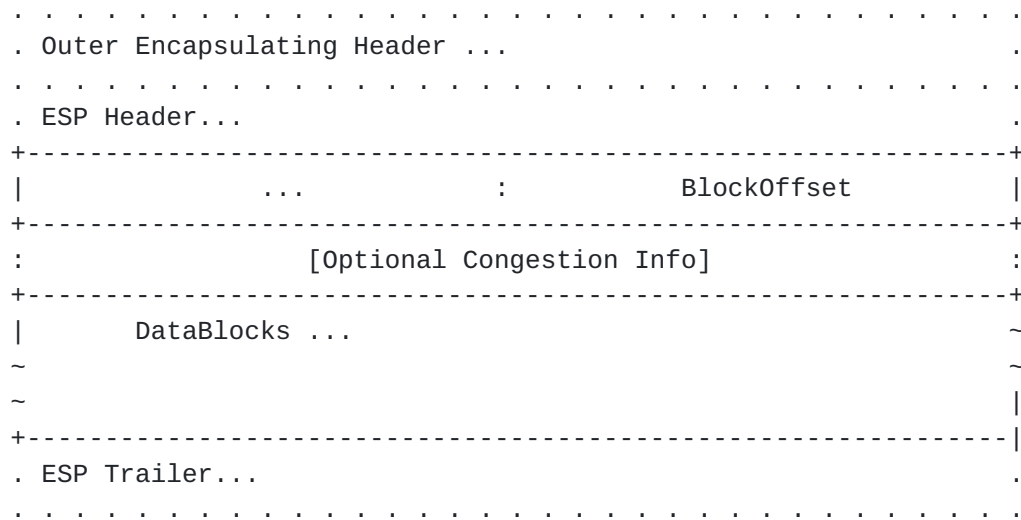
                 Figure 1: Layout of an IP-TFS IPsec Packet

   The "BlockOffset" value is either zero or some offset into or past
   the end of the "DataBlocks" data.

   If the "BlockOffset" value is zero it means that the "DataBlocks"
   data begins with a new data block.

   Conversely, if the "BlockOffset" value is non-zero it points to the
   start of the new data block, and the initial "DataBlocks" data
   belongs to a previous data block that is still being re-assembled.

   The "BlockOffset" can point past the end of the "DataBlocks" data
   which indicates that the next data block occurs in a subsequent
   encapsulating packet.

   Having the "BlockOffset" always point at the next available data
   block allows for quick recovery with minimal inner packet loss in the
   presence of outer encapsulating packet loss.

   An example IP-TFS packet flow can be found in Appendix A.

2.2.1. Data Blocks

```
   +---------------------------------------------------------------+
   | Type  | rest of IPv4, IPv6 or pad.
   +--------
```

                   Figure 2: Layout of IP-TFS data block

   A data block is defined by a 4-bit type code followed by the data
   block data. The type values have been carefully chosen to coincide

   with the IPv4/IPv6 version field values so that no per-data block
   type overhead is required to encapsulate an IP packet. Likewise, the
   length of the data block is extracted from the encapsulated IPv4 or
   IPv6 packet's length field.

2.2.2. No Implicit Padding Required

   It's worth noting that there is never a need for an implicit pad at
   the end of an encapsulating packet. Even when the start of a data
   block occurs near the end of a encapsulating packet such that there
   is no room for the length field of the encapsulated header to be
   included in the current encapsulating packet, the fact that the
   length comes at a known location and is guaranteed to be present is
   enough to fetch the length field from the subsequent encapsulating
   packet payload. Only when there is no data to encapsulate is padding
   required, and then an explicit "Pad Data Block" would be used to
   identify the padding.

2.2.3. Empty Payload

   In order to support reporting of congestion control information
   (described later) on a non-IP-TFS enabled SA, IP-TFS allows for the
   sending of an IP-TFS payload with no data blocks (i.e., the ESP
   payload length is equal to the IP-TFS header length). This special
   payload is called an empty payload.

2.2.4. IP Header Value Mapping

   [RFC4301] provides some direction on when and how to map various
   values from an inner IP header to the outer encapsulating header,
   namely the Don't-Fragment (DF) bit ([RFC0791] and [RFC8200]), the
   Differentiated Services (DS) field [RFC2474] and the Explicit
   Congestion Notification (ECN) field [RFC3168]. Unlike [RFC4301] with
   IP-TFS we may and often will be encapsulating more than 1 IP packet
   per ESP packet. To deal with this we further restrict these
   mappings. In particular we never map the inner DF bit as it is
   unrelated to the IP-TFS tunnel functionality; we never IP fragment
   the inner packets and the inner packets will not affect the
   fragmentation of the outer encapsulation packets. Likewise, the ECN
   value need not be mapped as any congestion related to the constant-
   send-rate IP-TFS tunnel is unrelated (by design!) to the inner
   traffic flow. Finally, by default the DS field SHOULD NOT be copied
   although an implementation MAY choose to allow for configuration to
   override this behavior. An implementation SHOULD also allow the DS
   value to be set by configuration.

2.3. Exclusive SA Use

It is not the intention of this specification to allow for mixed use
of an IP-TFS enabled SA. In other words, an SA that has IP-TFS
enabled is exclusively for IP-TFS use and MUST NOT have non-IP-TFS
payloads such as IP (IP protocol 4), TCP transport (IP protocol 6),
or ESP pad packets (protocol 59) intermixed with non-empty IP-TFS (IP
protocol TBD1) payloads. While it's possible to envision making the
algorithm work in the presence of sequence number skips in the IP-TFS
payload stream, the added complexity is not deemed worthwhile. Other
IPsec uses can configure and use their own SAs.

2.4. Initiating IP-TFS Operation On The SA.

While a user will normally configure their IPsec tunnel (SA) to
operate using IP-TFS to start, we also allow IP-TFS operation to be
enabled post-SA creation and use. This late-enabling may be useful
for debugging or other purposes. To support this late-enabled
operation the receiver switches to IP-TFS operation on receipt of the
first ESP payload with the IPTFS_PROTOCOL indicated as the payload
type which also contains a data block (i.e., a non-empty IP-TFS
payload). The the receipt of an empty IPTFS_PROTOCOL payload (i.e.,
one without any data blocks) is used to communicate congestion
control information from the receiver back to the sender on a non-IP-
TFS enabled SA, and MUST NOT cause IP-TFS to be enabled on that SA.

2.5. Modes of Operation

Just as with normal IPsec/ESP tunnels, IP-TFS tunnels are
unidirectional. Bidirectional IP-TFS functionality is achieved by
setting up 2 IP-TFS tunnels, one in either direction.

An IP-TFS tunnel can operate in 2 modes, a non-congestion controlled
mode and congestion controlled mode.

2.5.1. Non-Congestion Controlled Mode

In the non-congestion controlled mode IP-TFS sends fixed-sized
packets at a constant rate. The packet send rate is constant and is
not automatically adjusted regardless of any network congestion
(e.g., packet loss).

For similar reasons as given in [RFC7510] the non-congestion
controlled mode should only be used where the user has full
administrative control over the path the tunnel will take. This is
required so the user can guarantee the bandwidth and also be sure as
to not be negatively affecting network congestion [RFC2914]. In this
case packet loss should be reported to the administrator (e.g., via

syslog, YANG notification, SNMP traps, etc) so that any failures due
to a lack of bandwidth can be corrected.

2.5.2. Congestion Controlled Mode

With the congestion controlled mode, IP-TFS adapts to network
congestion by lowering the packet send rate to accommodate the
congestion, as well as raising the rate when congestion subsides.
Since overhead is per packet, by allowing for maximal fixed-size
packets and varying the send rate we minimize transport overhead.

The output of the congestion control algorithm will adjust the rate
at which the ingress sends packets. While this document does not
require a specific congestion control algorithm, best current
practice RECOMMENDS that the algorithm conform to [RFC5348].
Congestion control principles are documented in [RFC2914] as well.
An example of an implementation of the [RFC5348] algorithm which
matches the requirements of IP-TFS (i.e., designed for fixed-size
packet and send rate varied based on congestion) is documented in
[RFC4342].

The required inputs for the TCP friendly rate control algorithm
described in [RFC5348] are the receivers loss event rate and the
senders estimated round-trip time (RTT). These values are provided
by IP-TFS using the congestion information header fields described in
Section 3. In particular these values are sufficient to implement
the algorithm described in [RFC5348].

At a minimum, the congestion information must be sent, from the
receiver as well as from the sender, at least once per RTT. Prior to
establishing an RTT the information SHOULD be sent constantly from
the sender and the receiver so that an RTT estimate can be
established. The lack of receiving this information over multiple
consecutive RTT intervals should be considered a congestion event
that causes the sender to adjust it's sending rate lower. For
example, [RFC4342] calls this the "no feedback timeout" and it is
equal to 4 RTT intervals. When a "no feedback timeout" has occurred
[RFC4342] halves the sending rate.

An implementation could choose to always include the congestion
information in it's IP-TFS payload header if sending on an IP-TFS
enabled SA. Since IP-TFS normally will operate with a large packet
size, the congestion information should represent a small portion of
the available tunnel bandwidth.

When an implementation is choosing a congestion control algorithm (or
a selection of algorithms) one should remember that IP-TFS is not

providing for reliable delivery of IP traffic, and so per packet ACKs
are not required and are not provided.

It's worth noting that the variable send-rate of a congestion
controlled IP-TFS tunnel, is not private; however, this send-rate is
being driven by network congestion, and as long as the encapsulated
(inner) traffic flow shape and timing are not directly affecting the
(outer) network congestion, the variations in the tunnel rate will
not weaken the provided inner traffic flow confidentiality.

2.5.2.1. Circuit Breakers

In additional to congestion control, implementations MAY choose to
define and implement circuit breakers [RFC8084] as a recovery method
of last resort. Enabling circuit breakers is also a reason a user
may wish to enable congestion information reports even when using the
non-congestion controlled mode of operation. The definition of
circuit breakers are outside the scope of this document.

3. Congestion Information

In order to support the congestion control mode, the sender needs to
know the loss event rate and also be able to approximate the RTT
([RFC5348]). In order to obtain these values the receiver sends
congestion control information on it's SA back to the sender. Thus,
in order to support congestion control the receiver must have a
paired SA back to the sender (this is always the case when the tunnel
was created using IKEv2). If the SA back to the sender is a non-IP-
TFS enabled SA then an IPTFS_PROTOCOL empty payload (i.e., header
only) is used to convey the information.

In order to calculate a loss event rate compatible with [RFC5348],
the receiver needs to have a round-trip time estimate. Thus the
sender communicates this estimate in the "RTT" header field. On
startup this value will be zero as no RTT estimate is yet known.

In order to allow the sender to calculate the "RTT" value, the
receiver communicates the last sequence number it has seen to the
sender in the "LastSeqNum" header field. In addition to the
"LastSeqNum" value, the receiver sends an estimate of the amount of
time between receiving the "LastSeqNum" packet and transmitting the
"LastSeqNum" value back to the sender in the congestion information.
It places this time estimate in the "Delay" header field along with
the "LastSeqNum".

The receiver also calculates, and communicates in the "LossEventRate"
header field, the loss event rate for use by the sender. This is
slightly different from [RFC4342] which periodically sends all the

loss interval data back to the sender so that it can do the
calculation. See Appendix B for a suggested way to calculate the
loss event rate value. Initially this value will be zero (indicating
no loss) until enough data has been collected by the receiver to
update it.

3.1. ECN Support

In additional to normal packet loss information IP-TFS supports use
of the ECN bits in the encapsulating IP header [RFC3168] for
identifying congestion. If ECN use is enabled and a packet arrives
at the egress endpoint with the Congestion Experienced (CE) value
set, then the receiver considers that packet as being dropped,
although it does not drop it. The receiver MUST set the E bit in any
IPTFS_PROTOCOL payload header containing a "LossEventRate" value
derived from a CE value being considered.

As noted in [RFC3168] the ECN bits are not protected by IPsec and
thus may constitute a covert channel. For this reason ECN use SHOULD
NOT be enabled by default.

4. Configuration

IP-TFS is meant to be deployable with a minimal amount of
configuration. All IP-TFS specific configuration should be able to
be specified at the unidirectional tunnel ingress (sending) side. It
is intended that non-IKEv2 operation is supported, at least, with
local static configuration.

4.1. Bandwidth

Bandwidth is a local configuration option. For non-congestion
controlled mode the bandwidth SHOULD be configured. For congestion
controlled mode one can configure the bandwidth or have no
configuration and let congestion control discover the maximum
bandwidth available. No standardized configuration method is
required.

4.2. Fixed Packet Size

The fixed packet size to be used for the tunnel encapsulation packets
can be configured manually or can be automatically determined using
Path MTU discovery (see [RFC1191] and [RFC8201]). No standardized
configuration method is required.

4.3. Congestion Control

   Congestion control is a local configuration option. No standardized
   configuration method is required.

5. IKEv2

5.1. TFS Type Transform Type

   When IP-TFS is used with IKEv2 a new "TFS Type" Transform Type (TBD2)
   is used to negotiate (as defined in [RFC7296]) the possible operation
   of IP-TFS on a child SA pair. This document defines 3 "TFS Type"
   Transform IDs for the new "TFS Type" Transform Type: None (0),
   TFS_IPTFS_CC (1) for congestion-controlled IP-TFS mode or
   TFS_IPTFS_NOCC (2) for non-congestion controlled IP-TFS mode. The
   selection of a proposal with a "TFS Type" Transform ID TFS_IPTFS_CC
   or TFS_IPTFS_NOCC does not mandate the use of IP-TFS, rather it
   indicates a willingness or intent to use IP-TFS on the SA pair. In
   addition, a new Notify Message Status Type IPTFS_REQUIREMENTS (TBD3)
   MAY be used by the initiator as well as the responder to further
   refine any operational requirements.

   Additional "TFS Type" Transform IDs may be defined in the future, and
   so readers are referred to [IKEV2IANA] for the most up to date list.

5.2. IPTFS_REQUIREMENTS Status Notification

   As mentioned in the previous section, a new Notify Message Status
   Type IPTFS_REQUIREMENTS (TBD3) MAY be sent by the initiator and/or
   the responder to further refine what will be supported. This
   notification is sent during IKE_AUTH and new CREATE_CHILD_SA
   exchanges; however, it MUST NOT be sent, and MUST be ignored, during
   a CREATE_CHILD_SA rekeying exchange as the requirements are not
   allowed to change during rekeying.

   The IPTFS_REQUIREMENTS notification contains a 1 octet payload of
   flags that specify any extra requirements from the sender of the
   message. The flag values (currently a single flag) are defined
   below. If the IPTFS_REQUIREMENTS notification is not sent then it
   implies that all the flag bits are clear.

```
    +-+-+-+-+-+-+-+-+
    |0|0|0|0|0|0|0|D|
    +-+-+-+-+-+-+-+-+
```

   0:
      MUST be zero on send and MUST be ignored on receive.

D:

   Don't Fragment bit, if set indicates the sender of the notify
   message does not support receiving packet fragments (i.e., inner
   packets MUST be sent using a single "Data Block"). This value
   only applies to what the sender is capable of receiving; the
   sender MAY still send packet fragments unless similarly restricted
   by the receiver in it's IPTFS_REQUIREMENTS notification.

6. Packet and Data Formats

6.1. ESP IP-TFS Payload

An ESP IP-TFS payload is identified by the IP protocol number
IPTFS_PROTOCOL (TBD1). This payload begins with a fixed 4 or 16
octet header followed by a variable amount of "DataBlocks" data. The
exact payload format and fields are defined in the following
sections.

6.1.1. Non-Congestion Control IPTFS_PROTOCOL Payload Format

The non-congestion control IPTFS_PROTOCOL payload is comprised of a 4
octet header followed by a variable amount of "DataBlocks" data as
shown below.

```
                        1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |V|C|        Reserved         |            BlockOffset           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |       DataBlocks ...
   +-+-+-+-+-+-+-+-+-+-+-
```

V:

   A 1 bit version field that MUST be set to zero. If received as
   one the packet MUST be dropped.

C:

   A 1 bit value that MUST be set to 0 to indicate no congestion
   control information is present.

Reserved:

   A 14 bit field set to 0 and ignored on receipt.

BlockOffset:

   A 16 bit unsigned integer counting the number of octets of
   "DataBlocks" data before the start of a new data block.
   "BlockOffset" can count past the end of the "DataBlocks" data in
   which case all the "DataBlocks" data belongs to the previous data

block being re-assembled. If the "BlockOffset" extends into
subsequent packets it continues to only count subsequent
"DataBlocks" data (i.e., it does not count subsequent packets
non-"DataBlocks" octets).

DataBlocks:
    Variable number of octets that begins with the start of a data
    block, or the continuation of a previous data block, followed by
    zero or more additional data blocks.

6.1.2. Congestion Control IPTFS_PROTOCOL Payload Format

The congestion control IPTFS_PROTOCOL payload is comprised of a 16
octet header followed by a variable amount of "DataBlocks" data as
shown below.
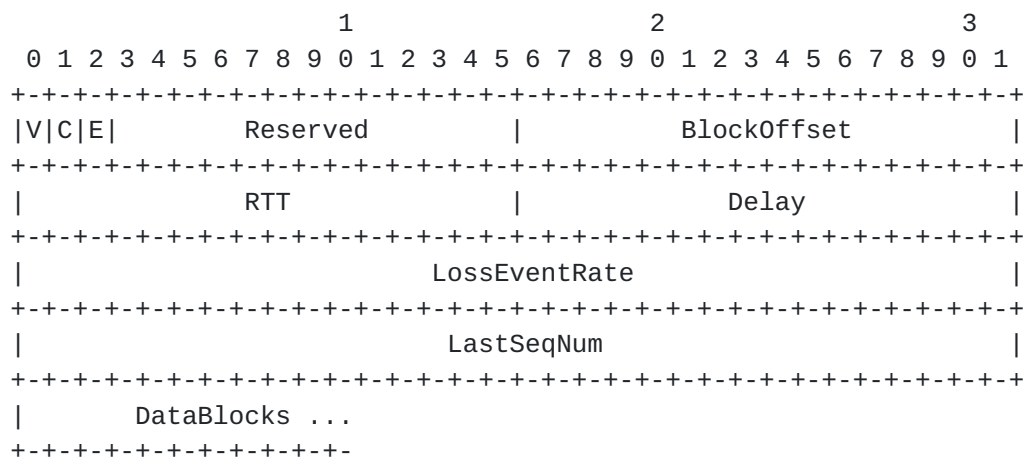
```
                    1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |V|C|E|      Reserved         |           BlockOffset           |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |             RTT             |             Delay               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                          LossEventRate                        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                           LastSeqNum                          |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |      DataBlocks ...
 +-+-+-+-+-+-+-+-+-+-
```

V:
    A 1 bit version field that MUST be set to zero. If received as
    one the packet MUST be dropped.

C:
    A 1 bit value that MUST be set to 1 which indicates the presence
    of the congestion information header fields "RTT", "Delay",
    "LossEventRate" and "LastSeqNum".

E:
    A 1 bit value if set indicates that Congestion Experienced (CE)
    ECN bits were received and used in deriving the reported
    "LossEventRate".

Reserved:
    A 13 bit field set to 0 and ignored on receipt.

BlockOffset:

The same value as the non-congestion controlled payload format
value.

RTT:
   A 16 bit value specifying the sender's current round-trip time
   estimate in milliseconds. The value MAY be zero prior to the
   sender having calculated a round-trip time estimate. The value
   SHOULD be set to zero on non-IP-TFS enabled SAs.

Delay:
   A 16 bit value specifying the delay in milliseconds incurred
   between the receiver receiving the "LastSeqNum" packet and the
   sending of this acknowledgement of it.

LossEventRate:
   A 32 bit value specifying the inverse of the current loss event
   rate as calculated by the receiver. A value of zero indicates no
   loss. Otherwise the loss event rate is "1/LossEventRate".

LastSeqNum:
   A 32 bit value containing the lower 32 bits of the largest
   sequence number last received. This is the latest in the sequence
   not necessarily the most recent (in the case of re-ordering of
   packets it may be less recent). When determining largest and 64
   bit extended sequence numbers are in use, the upper 32 bits should
   be used during the comparison.

DataBlocks:
   Variable number of octets that begins with the start of a data
   block, or the continuation of a previous data block, followed by
   zero or more additional data blocks. For the special case of
   sending congestion control information on an non-IP-TFS enabled SA
   this value MUST be empty (i.e., be zero octets long).

6.1.3. Data Blocks

```
                        1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | Type  | IPv4, IPv6 or pad...
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

Type:
   A 4 bit field where 0x0 identifies a pad data block, 0x4 indicates
   an IPv4 data block, and 0x6 indicates an IPv6 data block.

6.1.3.1. IPv4 Data Block

```
                     1                   2                   3
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |  0x4  |  IHL  |  TypeOfService |           TotalLength         |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  | Rest of the inner packet ...
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

These values are the actual values within the encapsulated IPv4
header. In other words, the start of this data block is the start of
the encapsulated IP packet.

Type:
   A 4 bit value of 0x4 indicating IPv4 (i.e., first nibble of the
   IPv4 packet).

TotalLength:
   The 16 bit unsigned integer length field of the IPv4 inner packet.

6.1.3.2. IPv6 Data Block

```
                     1                   2                   3
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |  0x6  | TrafficClass  |                FlowLabel              |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |           TotalLength         | Rest of the inner packet ...
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

These values are the actual values within the encapsulated IPv6
header. In other words, the start of this data block is the start of
the encapsulated IP packet.

Type:
   A 4 bit value of 0x6 indicating IPv6 (i.e., first nibble of the
   IPv6 packet).

TotalLength:
   The 16 bit unsigned integer length field of the inner IPv6 inner
   packet.

6.1.3.3. Pad Data Block

```
                              1                   2                   3
        0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |  0x0  | Padding ...
       +-+-+-+-+-+-+-+-+-
```

   Type:
      A 4 bit value of 0x0 indicating a padding data block.

   Padding:
      extends to end of the encapsulating packet.

7. IANA Considerations

7.1. IPTFS_PROTOCOL Type

   This document requests a protocol number IPTFS_PROTOCOL be allocated
   by IANA from "Assigned Internet Protocol Numbers" registry for
   identifying the IP-TFS ESP payload format.

   Type:
      TBD1

   Description:
      IP-TFS ESP payload format.

   Reference:
      This document

7.2. IKEv2 Transform Type TFS Type

   This document requests an IKEv2 Transform Type "TFS Type" be
   allocated by IANA from the "Transform Type Values" registry.

   Type:
      TBD2

   Description:
      TFS Type

   Used In:
      (optional in ESP)

   Reference:
      This document

7.3. TFS Type Transform IDs Registry

   This document requests a "Transform Type TBD3 - TFS Type Transform
   IDs" registry be created. The registration procedure is Expert
   Review. The initial values are as follows:

```
   Number  Name             Reference
   ----------------------------------------
        0  NONE             This document
        1  TFS_IPTFS_CC     This document
        2  TFS_IPTFS_NOCC   This document
   3-65535  Reserved         This document
```

7.4. IPTFS_REQUIREMENTS Notify Message Status Type

   This document requests a status type IPTFS_REQUIREMENTS be allocated
   from the "IKEv2 Notify Message Types - Status Types" registry.

   Value:
      TBD3

   Name:
      IPTFS_REQUIREMENTS

   Reference:
      This document

8. Security Considerations

   This document describes a mechanism to add Traffic Flow
   Confidentiality to IP traffic. Use of this mechanism is expected to
   increase the security of the traffic being transported. Other than
   the additional security afforded by using this mechanism, IP-TFS
   utilizes the security protocols [RFC4303] and [RFC7296] and so their
   security considerations apply to IP-TFS as well.

   As noted previously in Section 2.5.2, for TFC to be fully maintained
   the encapsulated traffic flow should not be affecting network
   congestion in a predictable way, and if it would be then non-
   congestion controlled mode use should be considered instead.

9. References

9.1. Normative References

   [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>.

   [RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)",
              RFC 4303, DOI 10.17487/RFC4303, December 2005,
              <https://www.rfc-editor.org/info/rfc4303>.

   [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>.

   [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>.

9.2. Informative References

   [AppCrypt]
              Schneier, B., "Applied Cryptography: Protocols,
              Algorithms, and Source Code in C", 11 2017.

   [I-D.iab-wire-image]
              Trammell, B. and M. Kuehlewind, "The Wire Image of a
              Network Protocol", draft-iab-wire-image-01 (work in
              progress), November 2018.

   [IKEV2IANA]
              IANA, "Internet Key Exchange Version 2 (IKEv2)
              Parameters",
              <http://www.iana.org/assignments/ikev2-parameters/>.

   [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791,
              DOI 10.17487/RFC0791, September 1981,
              <https://www.rfc-editor.org/info/rfc791>.

   [RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191,
              DOI 10.17487/RFC1191, November 1990,
              <https://www.rfc-editor.org/info/rfc1191>.

   [RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black,
              "Definition of the Differentiated Services Field (DS
              Field) in the IPv4 and IPv6 Headers", RFC 2474,
              DOI 10.17487/RFC2474, December 1998,
              <https://www.rfc-editor.org/info/rfc2474>.

   [RFC2914] Floyd, S., "Congestion Control Principles", BCP 41,
              RFC 2914, DOI 10.17487/RFC2914, September 2000,
              <https://www.rfc-editor.org/info/rfc2914>.

   [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition
              of Explicit Congestion Notification (ECN) to IP",
              RFC 3168, DOI 10.17487/RFC3168, September 2001,
              <https://www.rfc-editor.org/info/rfc3168>.

   [RFC4301] Kent, S. and K. Seo, "Security Architecture for the
              Internet Protocol", RFC 4301, DOI 10.17487/RFC4301,
              December 2005, <https://www.rfc-editor.org/info/rfc4301>.

   [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,
              DOI 10.17487/RFC4342, March 2006,
              <https://www.rfc-editor.org/info/rfc4342>.

   [RFC5348] Floyd, S., Handley, M., Padhye, J., and J. Widmer, "TCP
              Friendly Rate Control (TFRC): Protocol Specification",
              RFC 5348, DOI 10.17487/RFC5348, September 2008,
              <https://www.rfc-editor.org/info/rfc5348>.

   [RFC7510] Xu, X., Sheth, N., Yong, L., Callon, R., and D. Black,
              "Encapsulating MPLS in UDP", RFC 7510,
              DOI 10.17487/RFC7510, April 2015,
              <https://www.rfc-editor.org/info/rfc7510>.

   [RFC8084] Fairhurst, G., "Network Transport Circuit Breakers",
              BCP 208, RFC 8084, DOI 10.17487/RFC8084, March 2017,
              <https://www.rfc-editor.org/info/rfc8084>.

   [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6
              (IPv6) Specification", STD 86, RFC 8200,
              DOI 10.17487/RFC8200, July 2017,
              <https://www.rfc-editor.org/info/rfc8200>.

   [RFC8201] McCann, J., Deering, S., Mogul, J., and R. Hinden, Ed.,
              "Path MTU Discovery for IP version 6", STD 87, RFC 8201,
              DOI 10.17487/RFC8201, July 2017,
              <https://www.rfc-editor.org/info/rfc8201>.

Appendix A. Example Of An Encapsulated IP Packet Flow

   Below we show an example inner IP packet flow within the
   encapsulating tunnel packet stream. Notice how encapsulated IP

packets can start and end anywhere, and more than one or less than 1
may occur in a single encapsulating packet.

```
  Offset: 0        Offset: 100    Offset: 2900    Offset: 1400
 [ ESP1  (1500) ][ ESP2  (1500) ][ ESP3  (1500) ][ ESP4  (1500) ]
 [--800--][--800--][60][-240-][--4000---------------------][pad]
```

Figure 3: Inner and Outer Packet Flow

The encapsulated IP packet flow (lengths include IP header and
payload) is as follows: an 800 octet packet, an 800 octet packet, a
60 octet packet, a 240 octet packet, a 4000 octet packet.

The "BlockOffset" values in the 4 IP-TFS payload headers for this
packet flow would thus be: 0, 100, 2900, 1400 respectively. The
first encapsulating packet ESP1 has a zero "BlockOffset" which points
at the IP data block immediately following the IP-TFS header. The
following packet ESP2s "BlockOffset" points inward 100 octets to the
start of the 60 octet data block. The third encapsulating packet
ESP3 contains the middle portion of the 4000 octet data block so the
offset points past its end and into the forth encapsulating packet.
The fourth packet ESP4s offset is 1400 pointing at the padding which
follows the completion of the continued 4000 octet packet.

[Appendix B](#). A Send and Loss Event Rate Calculation

The current best practice indicates that congestion control should be
done in a TCP friendly way. A TCP friendly congestion control
algorithm is described in [[RFC5348](#)]. For our use case (as with
[[RFC4342](#)]) we consider our (fixed) packet size the segment size for
the algorithm. The formula for the send rate is then as follows:

$$
X\_Pps = \frac{1}{R * (\sqrt{2*p/3} + 12*\sqrt{3*p/8}*p*(1+32*p^2))}
$$

Where "X_Pps" is the send rate in packets per second, "R" is the
round trip time estimate and "p" is the loss event rate (the inverse
of which is provided by the receiver).

The IP-TFS receiver, having the RTT estimate from the sender MAY use
the same method as described in [[RFC4342](#)] to collect the loss
intervals and calculate the loss event rate value using the weighted
average as indicated. The receiver communicates the inverse of this
value back to the sender in the IPTFS_PROTOCOL payload header field
"LossEventRate".

The IP-TFS sender now has both the "R" and "p" values and can
calculate the correct sending rate ("X_Pps"). If following [RFC5348]
the sender SHOULD also use the slow start mechanism described therein
when the IP-TFS SA is first established.

Appendix C. Comparisons of IP-TFS

C.1. Comparing Overhead

C.1.1. IP-TFS Overhead

The overhead of IP-TFS is 40 bytes per outer packet. Therefore the
octet overhead per inner packet is 40 divided by the number of outer
packets required (fractional allowed). The overhead as a percentage
of inner packet size is a constant based on the Outer MTU size.

```
OH = 40 / Outer Payload Size / Inner Packet Size
OH % of Inner Packet Size = 100 * OH / Inner Packet Size
OH % of Inner Packet Size = 4000 / Outer Payload Size
```

Type	IP-TFS	IP-TFS	IP-TFS
MTU	576	1500	9000
PSize	536	1460	8960
40	7.46%	2.74%	0.45%
576	7.46%	2.74%	0.45%
1500	7.46%	2.74%	0.45%
9000	7.46%	2.74%	0.45%

Figure 4: IP-TFS Overhead as Percentage of Inner Packet Size

C.1.2. ESP with Padding Overhead

The overhead per inner packet for constant-send-rate padded ESP
(i.e., traditional IPsec TFC) is 36 octets plus any padding, unless
fragmentation is required.

When fragmentation of the inner packet is required to fit in the
outer IPsec packet, overhead is the number of outer packets required
to carry the fragmented inner packet times both the inner IP overhead
(20) and the outer packet overhead (36) minus the initial inner IP
overhead plus any required tail padding in the last encapsulation
packet. The required tail padding is the number of required packets
times the difference of the Outer Payload Size and the IP Overhead
minus the Inner Payload Size. So:

```
    Inner Paylaod Size = IP Packet Size - IP Overhead
    Outer Payload Size = MTU - IPsec Overhead


                 Inner Payload Size
   NF0 = ----------------------------------
            Outer Payload Size - IP Overhead


   NF = CEILING(NF0)


   OH = NF * (IP Overhead + IPsec Overhead)
        - IP Overhead
        + NF * (Outer Payload Size - IP Overhead)
        - Inner Payload Size


   OH = NF * (IPsec Overhead + Outer Payload Size)
        - (IP Overhead + Inner Payload Size)


   OH = NF * (IPsec Overhead + Outer Payload Size)
        - Inner Packet Size
```

C.2. Overhead Comparison

The following tables collect the overhead values for some common L3
MTU sizes in order to compare them. The first table is the number of
octets of overhead for a given L3 MTU sized packet. The second table
is the percentage of overhead in the same MTU sized packet.

```
         Type  ESP+Pad  ESP+Pad  ESP+Pad  IP-TFS   IP-TFS   IP-TFS
        L3 MTU     576     1500     9000     576     1500     9000
         PSize     540     1464     8964     536     1460     8960
        --------------------------------------------------------
            40     500     1424     8924     3.0      1.1      0.2
           128     412     1336     8836     9.6      3.5      0.6
           256     284     1208     8708    19.1      7.0      1.1
           536       4      928     8428    40.0     14.7      2.4
           576     576      888     8388    43.0     15.8      2.6
          1460     268        4     7504   109.0     40.0      6.5
          1500     228     1500     7464   111.9     41.1      6.7
          8960    1408     1540        4   668.7    245.5     40.0
          9000    1368     1500     9000   671.6    246.6     40.2
```

                 Figure 5: Overhead comparison in octets

```
       Type  ESP+Pad  ESP+Pad   ESP+Pad  IP-TFS   IP-TFS   IP-TFS
        MTU      576     1500      9000     576     1500     9000
      PSize      540     1464      8964     536     1460     8960
      -------------------------------------------------------------
         40  1250.0%  3560.0%  22310.0%   7.46%    2.74%    0.45%
        128   321.9%  1043.8%   6903.1%   7.46%    2.74%    0.45%
        256   110.9%   471.9%   3401.6%   7.46%    2.74%    0.45%
        536     0.7%   173.1%   1572.4%   7.46%    2.74%    0.45%
        576   100.0%   154.2%   1456.2%   7.46%    2.74%    0.45%
       1460    18.4%     0.3%    514.0%   7.46%    2.74%    0.45%
       1500    15.2%   100.0%    497.6%   7.46%    2.74%    0.45%
       8960    15.7%    17.2%      0.0%   7.46%    2.74%    0.45%
       9000    15.2%    16.7%    100.0%   7.46%    2.74%    0.45%
```

              Figure 6: Overhead as Percentage of Inner Packet Size

C.3. Comparing Available Bandwidth

   Another way to compare the two solutions is to look at the amount of
   available bandwidth each solution provides. The following sections
   consider and compare the percentage of available bandwidth. For the
   sake of providing a well understood baseline we will also include
   normal (unencrypted) Ethernet as well as normal ESP values.

C.3.1. Ethernet

   In order to calculate the available bandwidth we first calculate the
   per packet overhead in bits. The total overhead of Ethernet is 14+4
   octets of header and CRC plus and additional 20 octets of framing
   (preamble, start, and inter-packet gap) for a total of 48 octets.
   Additionally the minimum payload is 46 octets.

```
       Size  E + P  E + P  E + P   IPTFS   IPTFS   IPTFS   Enet    ESP
        MTU    590   1514   9014     590    1514    9014    any    any
         OH     74     74     74      78      78      78     38     74
      -------------------------------------------------------------
         40    614   1538   9038      45      42      40     84    114
        128    614   1538   9038     146     134     129    166    202
        256    614   1538   9038     293     269     258    294    330
        536    614   1538   9038     614     564     540    574    610
        576   1228   1538   9038     659     606     581    614    650
       1460   1842   1538   9038    1672    1538    1472   1498   1534
       1500   1842   3076   9038    1718    1580    1513   1538   1574
       8960  11052  10766   9038   10263    9438    9038   8998   9034
       9000  11052  10766  18076   10309    9480    9078   9038   9074
```

                     Figure 7: L2 Octets Per Packet

Size	E + P	E + P	E + P	IPTFS	IPTFS	IPTFS	Enet	ESP
MTU	590	1514	9014	590	1514	9014	any	any
OH	74	74	74	78	78	78	38	74
40	2.0M	0.8M	0.1M	27.3M	29.7M	31.0M	14.9M	11.0M
128	2.0M	0.8M	0.1M	8.5M	9.3M	9.7M	7.5M	6.2M
256	2.0M	0.8M	0.1M	4.3M	4.6M	4.8M	4.3M	3.8M
536	2.0M	0.8M	0.1M	2.0M	2.2M	2.3M	2.2M	2.0M
576	1.0M	0.8M	0.1M	1.9M	2.1M	2.2M	2.0M	1.9M
1460	678K	812K	138K	747K	812K	848K	834K	814K
1500	678K	406K	138K	727K	791K	826K	812K	794K
8960	113K	116K	138K	121K	132K	138K	138K	138K
9000	113K	116K	69K	121K	131K	137K	138K	137K

Figure 8: Packets Per Second on 10G Ethernet

Size	E + P	E + P	E + P	IPTFS	IPTFS	IPTFS	Enet	ESP
	590	1514	9014	590	1514	9014	any	any
	74	74	74	78	78	78	38	74
40	6.51%	2.60%	0.44%	87.30%	94.93%	99.14%	47.62%	35.09%
128	20.85%	8.32%	1.42%	87.30%	94.93%	99.14%	77.11%	63.37%
256	41.69%	16.64%	2.83%	87.30%	94.93%	99.14%	87.07%	77.58%
536	87.30%	34.85%	5.93%	87.30%	94.93%	99.14%	93.38%	87.87%
576	46.91%	37.45%	6.37%	87.30%	94.93%	99.14%	93.81%	88.62%
1460	79.26%	94.93%	16.15%	87.30%	94.93%	99.14%	97.46%	95.18%
1500	81.43%	48.76%	16.60%	87.30%	94.93%	99.14%	97.53%	95.30%
8960	81.07%	83.22%	99.14%	87.30%	94.93%	99.14%	99.58%	99.18%
9000	81.43%	83.60%	49.79%	87.30%	94.93%	99.14%	99.58%	99.18%

Figure 9: Percentage of Bandwidth on 10G Ethernet

A sometimes unexpected result of using IP-TFS (or any packet
aggregating tunnel) is that, for small to medium sized packets, the
available bandwidth is actually greater than native Ethernet. This
is due to the reduction in Ethernet framing overhead. This increased
bandwidth is paid for with an increase in latency. This latency is
the time to send the unrelated octets in the outer tunnel frame. The
following table illustrates the latency for some common values on a
10G Ethernet link. The table also includes latency introduced by
padding if using ESP with padding.

```
                  ESP+Pad  ESP+Pad  IP-TFS   IP-TFS
                  1500     9000     1500     9000

                  -------------------------------------------
              40  1.14 us  7.14 us  1.17 us  7.17 us
             128  1.07 us  7.07 us  1.10 us  7.10 us
             256  0.97 us  6.97 us  1.00 us  7.00 us
             536  0.74 us  6.74 us  0.77 us  6.77 us
             576  0.71 us  6.71 us  0.74 us  6.74 us
            1460  0.00 us  6.00 us  0.04 us  6.04 us
            1500  1.20 us  5.97 us  0.00 us  6.00 us
```

Figure 10: Added Latency

Notice that the latency values are very similar between the two
solutions; however, whereas IP-TFS provides for constant high
bandwidth, in some cases even exceeding native Ethernet, ESP with
padding often greatly reduces available bandwidth.

Appendix D. Acknowledgements

We would like to thank Don Fedyk for help in reviewing this work.

Appendix E. Contributors

The following people made significant contributions to this document.

   Lou Berger
   LabN Consulting, L.L.C.

   Email: lberger@labn.net

Author's Address

   Christian Hopps
   LabN Consulting, L.L.C.

   Email: chopps@chopps.org