Workgroup: Network Working Group
Internet-Draft: draft-ietf-ipsecme-iptfs-19
Published: 4 September 2022
Intended Status: Standards Track
Expires: 8 March 2023
Authors: C. Hopps
LabN Consulting, L.L.C.

IP-TFS: Aggregation and Fragmentation Mode for ESP and its Use for IP Traffic Flow Security

Abstract

This document describes a mechanism for aggregation and fragmentation of IP packets when they are being encapsulated in ESP payloads. This new payload type can be used for various purposes such as decreasing encapsulation overhead for small IP packets; however, the focus in this document is to enhance IPsec traffic flow security (IP-TFS) by adding Traffic Flow Confidentiality (TFC) to encrypted IP encapsulated traffic. TFC 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 as non-constant send-rate usage.

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 8 March 2023.

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 directly obscuring the data with encryption [RFC4303], the patterns in the message traffic may expose information due to variations in its shape and timing ([RFC8546], [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.

This document defines an aggregation and fragmentation (AGGFRAG) mode for ESP, and its use for IP Traffic Flow Security (IP-TFS). This solution provides for full TFC without the aforementioned bandwidth limitation. This is accomplished by using 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. A non-constant send-rate is allowed, but the confidentiality properties of its use are outside the scope of this document.

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

Additionally, IP-TFS provides for operating fairly within congested networks [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.

The mechanisms, such as the AGGFRAG mode, defined in this document are generic with the intent of allowing for non-TFS uses, but such uses are outside the scope of this document.

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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

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

2. The AGGFRAG Tunnel

As mentioned in Section 1, AGGFRAG mode utilizes an IPsec [RFC4303] tunnel as its transport. For the purpose of IP-TFS, fixed-sized encapsulating packets are sent at a constant rate on the AGGFRAG tunnel.

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

The fixed packet size MAY either be specified manually or be determined through other methods such as the Packetization Layer MTU Discovery (PLMTUD) ([RFC4821], [RFC8899]) or Path MTU discovery (PMTUD) ([RFC1191], [RFC8201]). PMTUD is known to have issues so PLMTUD is considered the more robust option. For PLMTUD, congestion control payloads can be used as in-band probes (see Section 6.1.2 and [RFC8899]).

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

The egress (receiving) side of the AGGFRAG tunnel MUST allow for and expect the ingress (sending) side of the AGGFRAG 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 by introducing an AGGFRAG mode for ESP.

AGGFRAG mode aggregates as well as fragments the inner IP traffic flow into encapsulating IPsec tunnel packets. For IP-TFS, the IPsec encapsulating tunnel packets are a fixed size. Padding is only added to 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.

This is accomplished using a new Encapsulating Security Payload (ESP, [RFC4303]) Next Header field value AGGFRAG_PAYLOAD (Section 6.1).

Other non-IP-TFS uses of this AGGFRAG mode have been suggested, such as increased performance through packet aggregation, as well as handling MTU issues using fragmentation. These uses are not defined here, but are also not restricted by this document.

2.2. Payload Content

The AGGFRAG_PAYLOAD payload content defined in this document consists of a 4 or 24 octet header followed by either a partial datablock, a full datablock, or multiple partial or full datablocks. The following diagram illustrates this payload within the ESP packet. See Section 6.1 for the exact formats of the AGGFRAG_PAYLOAD payload.

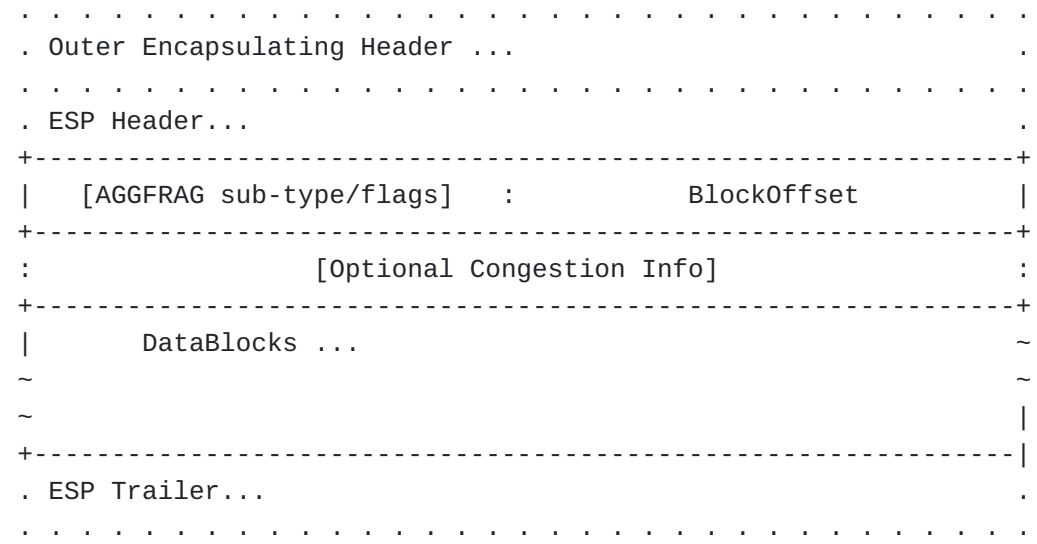

Figure 1: Layout of an AGGFRAG mode IPsec Packet

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 the data block that is still being re-assembled.

If the BlockOffset points past the end of the DataBlocks data then the next data block occurs in a subsequent encapsulating packet.

Having the BlockOffset always point at the next available data block allows for recovering the next inner packet in the presence of outer encapsulating packet loss.

An example AGGFRAG mode packet flow can be found in Appendix A.

2.2.1. Data Blocks

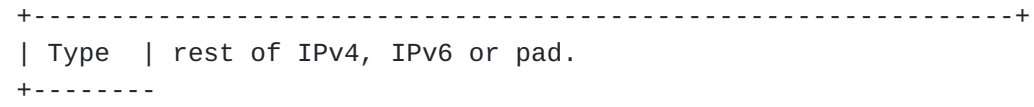

Figure 2: Layout of a DataBlock

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's Total Length or IPv6's Payload Length fields.

2.2.2. End Padding

Since a data block's type is identified in its first 4-bits, the only time padding is required is when there is no data to encapsulate. For this end padding a Pad Data Block is used.

2.2.3. Fragmentation, Sequence Numbers and All-Pad Payloads

In order for a receiver to reassemble fragmented inner packets, the sender MUST send the inner packet fragments back-to-back in the logical outer packet stream (i.e., using consecutive ESP sequence numbers). However, the sender is allowed to insert "all-pad" payloads (i.e., payloads with a BlockOffset of zero and a single pad DataBlock) in between the packets carrying the inner packet fragment payloads. This interleaving of all-pad payloads allows the sender to always send a tunnel packet, regardless of the encapsulation computational requirements.

When a receiver is reassembling an inner packet, and it receives an "all-pad" payload, it increments the expected sequence number that the next inner packet fragment is expected to arrive in.

Given the above, the receiver will need to handle out-of-order arrival of outer ESP packets prior to reassembly processing. ESP already provides for optionally detecting replay attacks. Detecting replay attacks normally utilizes a window method. A similar sequence number based sliding window can be used to correct re-ordering of the outer packet stream. Receiving a larger (newer) sequence number packet advances the window, and received older ESP packets whose sequence numbers the window has passed by are dropped. A good choice for the size of this window depends on the amount of misordering the user is experiencing; however, a value of 3 has been suggested as a default when no more informed choice exists.

As the amount of misordering that may be present is hard to predict, the window size SHOULD be configurable by the user. Implementations MAY also dynamically adjust the reordering window based on actual misordering seen in arriving packets.

Please note, when IP-TFS sends a continuous stream of packets, there is no requirement for an explicit lost packet timer; however, using a lost packet timer is RECOMMENDED. If an implementation does not use a lost packet timer and only considers an outer packet lost when the reorder window moves by it, the inner traffic can be delayed by up to the reorder window size times the per packet send rate. This delay could be significant for slower send rates or when larger reorder window sizes are in use. As the lost packet timer affects delay of inner packet delivery, an implementation or user could choose to set it proportionate to the tunnel rate.

While ESP guarantees an increasing sequence number with subsequently sent packets, it does not actually require the sequence numbers to be generated consecutively (e.g., sending only even numbered sequence numbers would be allowed as long as they are always increasing). Gaps in the sequence numbers will not work for this document so the sequence number stream MUST increase monotonically by 1 for each subsequent packet.

When using the AGGFRAG_PAYLOAD in conjunction with replay detection, the window size for both MAY be reduced to the smaller of the two window sizes. This is because packets outside of the smaller window but inside the larger would still be dropped by the mechanism with the smaller window size. However, there is also no requirement to make these values the same. Indeed, in some cases, such as slow tunnels where a very small or zero reorder window size is appropriate, the user may still want a large replay detection window to log replayed packets. Additionally, large replay windows can be

implemented with very little overhead compared to large reorder windows.

Finally, as sequence numbers are reset when switching SAs (e.g., when re-keying a child SA), senders MUST NOT send initial fragments of an inner packet using one SA and subsequent fragments in a different SA.

A note on BlockOffset values, senders MUST encode the BlockOffset consistent with the immediately preceding non-all-pad payload packet. Specifically, if the immediately preceding non-all-pad payload packet ended with a Pad Data Block, this BlockOffset MUST be zero, as Pad Data Blocks are never fragmented. The BlockOffset MUST be consistent with the remaining size implied by the native length encoding of the fragmented inner packet.

2.2.3.1. Optional Extra Padding

When the tunnel bandwidth is not being fully utilized, a sender MAY pad-out the current encapsulating packet in order to deliver an inner packet un-fragmented in the following outer packet. The benefit would be to avoid inner packet fragmentation in the presence of a bursty offered load (non-bursty traffic will naturally not fragment). Senders MAY also choose to allow for a minimum fragment size to be configured (e.g., as a percentage of the AGGFRAG_PAYLOAD payload size) to avoid fragmentation at the cost of tunnel bandwidth. The cost with these methods is complexity and added delay of inner traffic. The main advantage to avoiding fragmentation is to minimize inner packet loss in the presence of outer packet loss. When this is worthwhile (e.g., how much loss and what type of loss is required, given different inner traffic shapes and utilization, for this to make sense), and what values to use for the allowable/added delay may be worth researching but is outside the scope of this document.

While use of padding to avoid fragmentation does not impact interoperability, used inappropriately it can reduce the effective throughput of a tunnel. Senders implementing either of the above approaches will need to take care to not reduce the effective capacity, and overall utility, of the tunnel through the overuse of padding.

2.2.4. Empty Payload

To support reporting of congestion control information (described later) using a non-AGGFRAG_PAYLOAD-enabled SA, it is allowed to send an AGGFRAG_PAYLOAD payload with no data blocks (i.e., the ESP payload length is equal to the AGGFRAG_PAYLOAD header length). This special payload is called an empty payload.

Currently this situation is only applicable in non-IKEv2 use cases.

2.2.5. 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], AGGFRAG mode may and often will be encapsulating more than one IP packet per ESP packet. To deal with this, these mappings are restricted further.

2.2.5.1. DF bit

AGGFRAG mode never maps the inner DF bit as it is unrelated to the AGGFRAG tunnel functionality; AGGFRAG mode never needs to IP fragment the inner packets and the inner packets will not affect the fragmentation of the outer encapsulation packets.

2.2.5.2. ECN value

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. The sender MAY still set the ECN value of inner packets based on the normal ECN specification [RFC3168], [RFC4301] and [RFC6040].

2.2.5.3. DS field

By default, the DS field SHOULD NOT be copied, although a sender MAY choose to allow for configuration to override this behavior. A sender SHOULD also allow the DS value to be set by configuration.

2.2.6. IPv4 Time-To-Live (TTL), IPv6 Hop Limit, and ICMP Messages

[RFC4301] specifies how to modify the inner packet IPv4 TTL [RFC0791] or IPv6 Hop Limit [RFC8200].

[RFC4301] also specifies how to apply policy to authenticated and unauthenticated ICMP error packets (e.g., Destination Unreachable) arriving at or being forwarded through the endpoint. In particular, whether to process, ignore or forward said packets. With one exception this document does not change the handling of these packets, they should be handled as specified in [RFC4301].

The one way in which an AGGFRAG tunnel differs in ICMP error packet mechanics is with PMTU. When fragmentation is enabled on the AGGFRAG tunnel, then no ICMP "too-big" errors need to be generated for

arriving ingress traffic as the arriving inner packets will be naturally fragmented by the AGGFRAG encapsultation.

Otherwise, when fragmentation has been disabled on the AGGFRAG tunnel, then the treatment of arriving inner traffic exactly maps to that of a non-AGGFRAG ESP tunnel. Explicitly, IPv4 with DF set and IPv6 packets which cannot fit in it's own outer packet payload will generate the appropriate ICMP "too-big" error as directed by [RFC4301], and IPv4 packets without DF set will be IP fragmented as directed by [RFC4301].

Packets egressing the tunnel continue to be handled as specified in [RFC4301].

All other aspects of PMTU and the handling of ICMP "Too Big" messages (i.e., with regards to the outer AGGFRAG/ESP tunnel packet size) also remain unchanged from [RFC4301].

2.2.7. Effective MTU of the Tunnel

Unlike [RFC4301], there is normally no effective MTU (EMTU) on an AGGFRAG tunnel as all IP packet sizes are properly transmitted without requiring IP fragmentation prior to tunnel ingress. That said, a sender MAY allow for explicitly configuring an MTU for the tunnel.

If fragmentation has been disabled on the AGGFRAG tunnel, then the tunnel's EMTU and behaviors are the same as normal IPsec tunnels [RFC4301].

2.3. Exclusive SA Use

This document does not specify mixed use of an AGGFRAG_PAYLOAD-enabled SA. A sender MUST only send AGGFRAG_PAYLOAD payloads over an SA configured for AGGFRAG mode.

2.4. Modes of Operation

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

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

2.4.1. Non-Congestion-Controlled Mode

In the non-congestion-controlled mode, IP-TFS sends fixed-sized packets over an AGGFRAG tunnel 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 MUST only be used where the user has full administrative control over any path the tunnel will take, and MUST NOT be used if this is not the case. 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. The use of circuit breakers is also RECOMMENDED (Section 2.4.2.1).

Users that choose the non-congestion-controlled mode need to understand that this mode will send packets at a constant rate utilizing a constant fixed bandwidth and will not adjust based on congestion. Thus, if they do not guarantee the bandwidth required by the tunnel, the tunnel's operation, as well as the rest of their network, may be negatively impacted.

One expected use case for non-congestion-controlled mode is to guarantee the full tunnel bandwidth is available and preferred over other non-tunnel traffic. In fact, a typical site-to-site use case might have all of the user traffic utilizing the IP-TFS tunnel.

Non-congestion-controlled mode is also appropriate if ESP over TCP is in use [RFC8229]. However, the use of TCP is considered a highly non-preferred, and a fall-back only solution for IPsec. This is also one of the reasons that TCP was not chosen as the encapsulation for IP-TFS instead of AGGFRAG.

2.4.2. 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, transport overhead is minimized.

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. [RFC4342] provides an example of the [RFC5348] algorithm which matches the requirements of IP-TFS (i.e., designed for fixed-size packets and send rate varied based on congestion).

The required inputs for the TCP friendly rate control algorithm described in [RFC5348] are the receiver's loss event rate and the sender's 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 and 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. Not receiving this information over multiple consecutive RTT intervals should be considered a congestion event that causes the sender to adjust its 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 MAY choose to always include the congestion information in its AGGFRAG 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. An implementation choosing to always send the data MAY also choose to only update the LossEventRate and RTT header field values it sends every RTT though.

When choosing a congestion control algorithm (or a selection of algorithms), note 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 is worth noting that the variable send-rate of a congestion-controlled AGGFRAG 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.

2.4.2.1. Circuit Breakers

In additional to congestion control, implementations that support non-congestion control mode SHOULD implement circuit breakers [RFC8084] as a recovery method of last resort. When circuit breakers are enabled an implementation SHOULD also enable congestion control reports so that circuit breakers have information to act on.

The pseudowire congestion considerations [RFC7893] are equally applicable to the mechanisms defined in this document, notably the text on inellastic traffic.

One example of a simple slow-trip circuit breaker (CB) an implementation may provide would utilize 2 values, the amount of persistent loss rate required to trip the CB, and the required length of time this persistent loss rate must be seen to trip the CB. These 2 value are required configuration from the user. When the CB is tripped the tunnel traffic is disabled, and an appropriate log message or other management type alarm is triggered indicating operate intervention is required.

2.5. Summary of Receiver Processing

An AGGFRAG-enabled SA receiver has a few tasks to perform.

The receiver MAY process incoming AGGFRAG_PAYLOAD payloads as soon as they arrive as much as it can. I.e., if the incoming AGGFRAG_PAYLOAD packet contains complete inner packet(s), the receiver should extract and transmit them immediately. For partial packets, the receiver needs to keep the partial packets in the memory until they fall out from the reordering window, or until the missing parts of the packets are received, in which case it will reassemble and transmit them. If the AGGFRAG_PAYLOAD payload contains multiple packets they SHOULD be sent out in the order they are in the AGGFRAG_PAYLOAD (i.e., keep the original order they were received on the other end). The cost of using this method is that an amplification of out-of-order delivery of inner packets can occur due to inner packet aggregation.

Instead of the method described in the previous paragraph, the receiver MAY reorder out-of-order AGGFRAG_PAYLOAD payloads received into in-sequence-order AGGFRAG_PAYLOAD payloads (Section 2.2.3), and only after it has an in-order AGGFRAG_PAYLOAD payload stream would the receiver transmits the inner packets. Using this method will ensure the inner packets are sent in order. The cost of this method is that a lost packet will cause a delay of up to the lost packet timer interval (or the full reorder window if no lost packet timer is used). Additionally, there can be extra burstiness in the output stream. This burstiness can happen when a lost packet is dropped from the re-order window, and the remaining outer packets in the re-order window are immediately processed and sent out back to back.

Additionally, if congestion control is enabled, the receiver sends congestion control data (Section 6.1.2) back to the sender as described in Section 2.4.2 and Section 3.

Finally, a note on receiving incorrect BlockOffset values. To account for misbehaving senders, a receiver SHOULD gracefully handle the case where the BlockOffset of consecutive packets, and/or the inner packet they share, do not agree. It MAY drop the inner packet, or one or both of the outer packets.

3. Congestion Information

In order to support the congestion-controlled mode, the sender needs to know the loss event rate and to approximate the RTT [RFC5348]. In order to obtain these values, the receiver sends congestion control information on its SA back to the sender. Thus, 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-AGGFRAG_PAYLOAD enabled SA then an AGGFRAG_PAYLOAD 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 for the sender to estimate its RTT value, the sender places a timestamp value in the TVal header field. On first receipt of this TVal, the receiver records the new TVal value along with the time it arrived locally. Subsequent receipt of the same TVal MUST NOT update the recorded time.

When the receiver sends its congestion control header it places this latest recorded TVal in the TEcho header field, along with 2 delay values, Echo Delay and Transmit Delay. The Echo Delay value is the time delta from the recorded arrival time of TVal and the current clock in microseconds. The second value, Transmit Delay, is the receiver's current transmission delay on the tunnel (i.e., the average time between sending packets on its half of the AGGFRAG tunnel).

When the sender receives back its TVal in the TEcho header field it calculates 2 RTT estimates. The first is the actual delay found by subtracting the TEcho value from its current clock and then subtracting Echo Delay as well. The second RTT estimate is found by adding the received Transmit Delay header value to the sender's own transmission delay (i.e., the average time between sending packets on its half of the AGGFRAG tunnel). The larger of these 2 RTT estimates SHOULD be used as the RTT value.

The two RTT estimates are required to handle different combinations of faster or slower tunnel packet paths with faster or slower fixed

tunnel rates. Choosing the larger of the two values guarantees that the RTT is never considered faster than the aggregate transmission delay based on the IP-TFS send rate (the second estimate), as well as never being considered faster than the actual RTT along the tunnel packet path (the first estimate).

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 addition to normal packet loss information, AGGFRAG mode 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 (receiving) side 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 AGGFRAG_PAYLOAD payload header containing a LossEventRate value derived from a CE value being considered.

[RFC3168] and [RFC4301], updated by [RFC6040] defines behaviors for marking the outer ECN field value based on the ECN field of the inner packet. As AGGFRAG mode may have multiple inner packets present in a single outer packet, and there is no obvious correct way to map these multiple values to the single outer packet ECN field value, the tunnel ingress endpoint SHOULD operate in the "compatibility" mode rather than the "default" mode from RFC6040. In particular this means that the ingress (sending) endpoint of the tunnel always sets the newly constructed outer encapsulating packet header ECN field to Not-ECT [RFC6040].

4. Configuration of AGGFRAG Tunnels for IP-TFS

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

YANG and MIB documents have been defined for IP-TFS in [I-D.ietf-ipsecme-yang-iptfs] and [I-D.ietf-ipsecme-mib-iptfs].

4.1. Bandwidth

Bandwidth is a local configuration option. For non-congestion-controlled mode, the bandwidth SHOULD be configured. For congestion-controlled mode, the bandwidth can be configured or the congestion control algorithm discovers and uses 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 MAY be configured manually or can be automatically determined using other methods such as PLMTUD ([RFC4821], [RFC8899]) or PMTUD ([RFC1191], [RFC8201]). As PMTUD is known to have issues, PLMTUD is considered the more robust option. 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. USE_AGGFRAG Notification Message

As mentioned previously AGGFRAG tunnels utilize ESP payloads of type AGGFRAG_PAYLOAD.

When using IKEv2, a new "USE_AGGFRAG" Notification Message enables the AGGFRAG_PAYLOAD payload on a child SA pair. The method used is similar to how USE_TRANSPORT_MODE is negotiated, as described in [RFC7296].

To request use of the AGGFRAG_PAYLOAD payload on the Child SA pair, the initiator includes the USE_AGGFRAG notification in an SA payload requesting a new Child SA (either during the initial IKE_AUTH or during CREATE_CHILD_SA exchanges). If the request is accepted then the response MUST also include a notification of type USE_AGGFRAG. If the responder declines the request the child SA will be established without AGGFRAG_PAYLOAD payload use enabled. If this is unacceptable to the initiator, the initiator MUST delete the child SA.

As the use of the AGGFRAG_PAYLOAD payload is currently only defined for non-transport mode tunnels, the USE_AGGFRAG notification MUST NOT be combined with USE_TRANSPORT notification.

The USE_AGGFRAG notification contains a 1 octet payload of flags that specify requirements from the sender of the notification. If

any requirement flags are not understood or cannot be supported by the receiver then the receiver SHOULD NOT enable use of AGGFRAG_PAYLOAD (either by not responding with the USE_AGGFRAG notification, or in the case of the initiator, by deleting the child SA if the now established non-AGGFRAG_PAYLOAD using SA is unacceptable).

The notification type and payload flag values are defined in Section 6.1.4.

6. Packet and Data Formats

The packet and data formats defined below are generic with the intent of allowing for non-IP-TFS uses, but such uses are outside the scope of this document.

6.1. AGGFRAG_PAYLOAD Payload

ESP Next Header value: 144

An AGGFRAG payload is identified by the ESP Next Header value AGGFRAG_PAYLOAD which has the value 144, which has been reserved in the IP protocol numbers space. The first octet of the payload indicates the format of the remaining payload data.

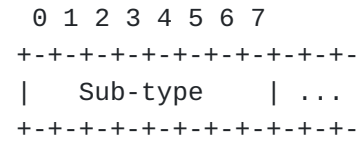

Figure 3: AGGFRAG_PAYLOAD payload format

Sub-type:
An 8-bit value indicating the payload format.

This document defines 2 payload sub-types. These payload formats are defined in the following sections.

6.1.1. Non-Congestion Control AGGFRAG_PAYLOAD Payload Format

The non-congestion control AGGFRAG_PAYLOAD payload consists of a 4-octet header followed by a variable amount of DataBlocks data as shown below.

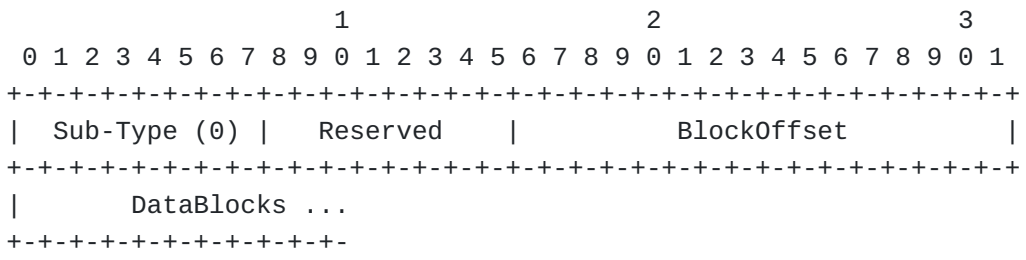

Figure 4: Non-congestion control payload format

Sub-type:
An octet indicating the payload format. For this non-congestion control format, the value is 0.

Reserved:
An octet set to 0 on generation 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. If the start of a new data block occurs in a subsequent payload the BlockOffset will point past the end of the DataBlocks data. In this case all the DataBlocks data belongs to the current data block being assembled. When the BlockOffset extends into subsequent payloads it continues to only count DataBlocks data (i.e., it does not count subsequent packets non-DataBlocks data such as header 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 AGGFRAG_PAYLOAD Payload Format

The congestion control AGGFRAG_PAYLOAD payload consists of a 24 octet header followed by a variable amount of DataBlocks data as shown below.

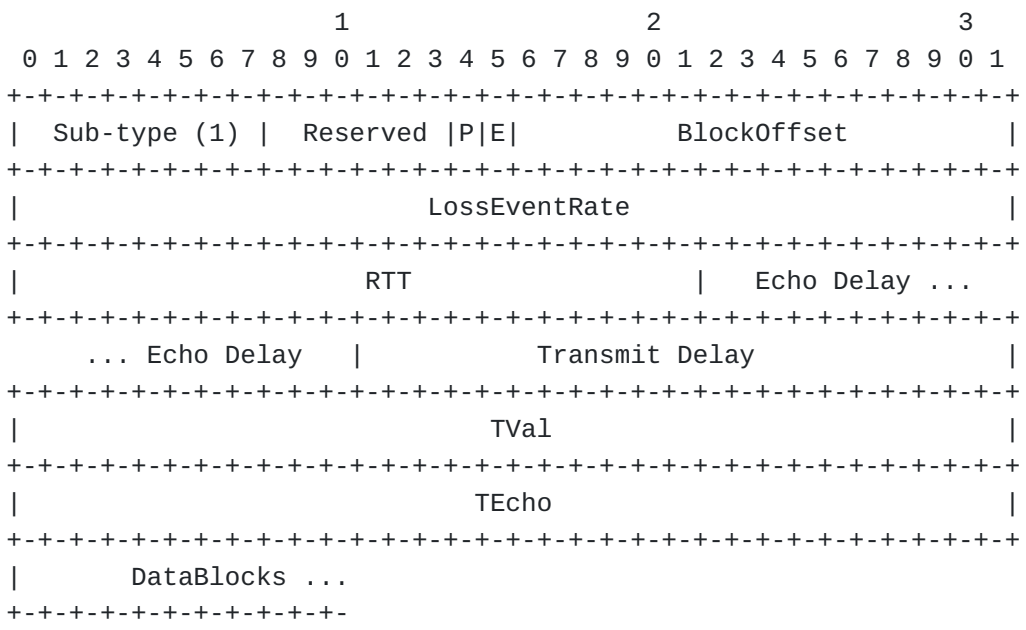

Figure 5: Congestion control payload format

Sub-type:
An octet indicating the payload format. For this congestion control format, the value is 1.

Reserved:
A 6-bit field set to 0 on generation and ignored on receipt.

P:
A 1-bit value that if set indicates that PLMTUD probing is in progress. This information can be used to avoid treating missing packets as loss events by the CC algorithm when running the PLMTUD probe algorithm.

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

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

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.

RTT:
A 22-bit value specifying the sender's current round-trip time estimate in microseconds. 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-AGGFRAG_PAYLOAD-enabled SAs. If the RTT is equal to or larger than 0x3FFFFF the value MUST be set to 0x3FFFFF.

Echo Delay:
A 21-bit value specifying the delay in microseconds incurred between the receiver first receiving the TVal value which it is sending back in TEcho. If the delay is equal to or larger than 0x1FFFFF the value MUST be set to 0x1FFFFF.

Transmit Delay:
A 21-bit value specifying the transmission delay in microseconds. This is the fixed (or average) delay on the receiver between it sending packets on the IPTFS tunnel. If the delay is equal to or larger than 0x1FFFFF the value MUST be set to 0x1FFFFF.

TVal:
An opaque 32-bit value that will be echoed back by the receiver in later packets in the TEcho field, along with an Echo Delay value of how long that echo took.

TEcho:
The opaque 32-bit value from a received packet's TVal field. The received TVal is placed in TEcho along with an Echo Delay value indicating how long it has been since receiving the TVal value.

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 a non-IP-TFS enabled SA this field MUST be empty (i.e., be zero octets long).

6.1.3. Data Blocks

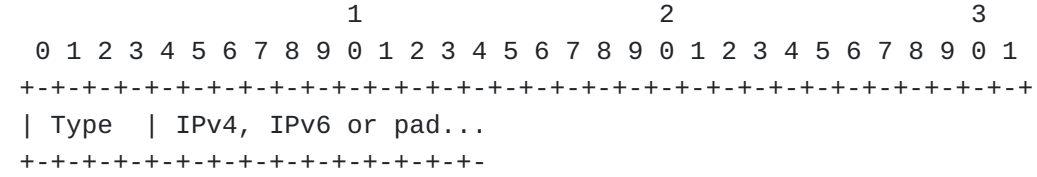

Figure 6: Data Block format

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

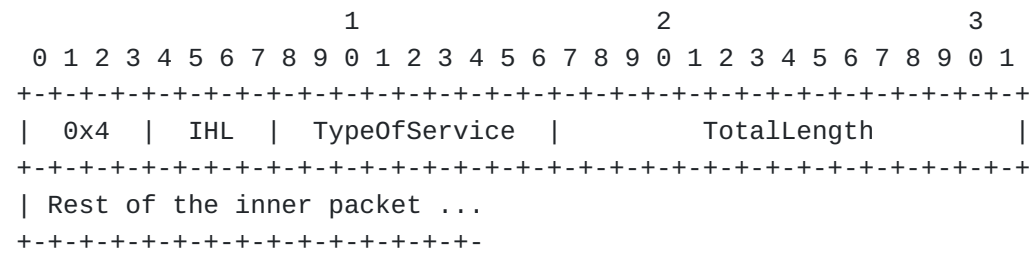

Figure 7: IPv4 Data Block format

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 "Total Length" field of the IPv4 inner packet.

6.1.3.2. IPv6 Data Block

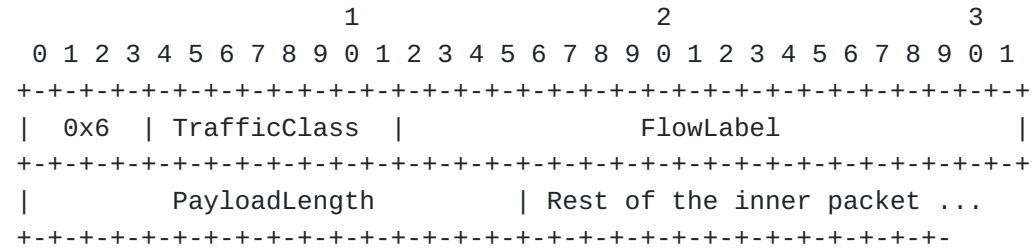

Figure 8: IPv6 Data Block format

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).

PayloadLength:
The 16-bit unsigned integer "Payload Length" field of the inner IPv6 inner packet.

6.1.3.3. Pad Data Block

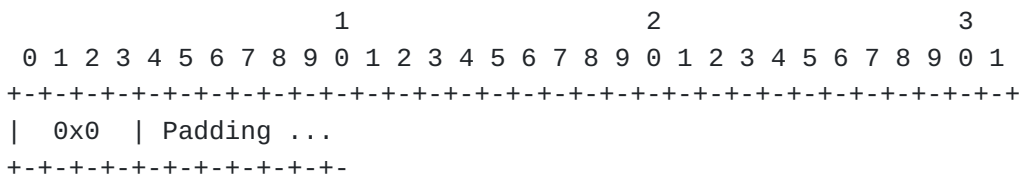

Figure 9: Pad Data Block format

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

Padding:
Extends to end of the encapsulating packet.

6.1.4. IKEv2 USE_AGGFRAG Notification Message

As discussed in Section 5.1, a notification message USE_AGGFRAG is used to negotiate use of the ESP AGGFRAG_PAYLOAD Next Header value.

The USE_AGGFRAG Notification Message State Type is 16442

The notification payload contains 1 octet of requirement flags. There are currently 2 requirement flags defined. This may be revised by later specifications.

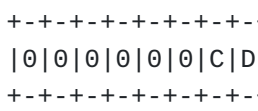

Figure 10: USE_AGGFRAG requirement flags

0:
6 bits - Reserved MUST be zero on send, unless defined by later specifications.

C:
Congestion Control bit. If set, then the sender is requiring that congestion control information MUST be returned to it periodically as defined in Section 3.

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 its USE_AGGFRAG notification.

7. IANA Considerations

7.1. ESP Next Header Value

Per the INT area directors direction, this document requests IANA allocate an IP protocol number from "Protocol Numbers - Assigned Internet Protocol Numbers" registry

Decimal:
144

Keyword:
AGGFRAG

Protocol:
AGGFRAG encapsulation payload for ESP (TEMPORARY - registered 2022-08-26, document sent to IESG Evaluation 2022-07-14)

Reference:
This document

7.2. AGGFRAG_PAYLOAD Sub-Type Registry

This document requests IANA create a registry called "AGGFRAG_PAYLOAD Sub-Type Registry" under a new category named "ESP AGGFRAG_PAYLOAD Parameters". The registration policy for this registry is "Expert Review" ([RFC8126] and [RFC7120]).

Name:
AGGFRAG_PAYLOAD Sub-Type Registry

Description:
AGGFRAG_PAYLOAD Payload Formats.

Reference:
This document

This initial content for this registry is as follows:

Sub-Type	Name	Reference
0	Non-Congestion Control Format	This document
1	Congestion Control Format	This document
3-255	Reserved	

7.3. USE_AGGFRAG Notify Message Status Type

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

Decimal:
16442

Name:
USE_AGGFRAG

Reference:
This document

8. Implementation Status

[RFC Ed.: please remove this entire section as well as the reference to RFC7942 prior to publication.]

[Section added during IESG review to help with evaluation]

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to RFC 7942, "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

Currently the author and contributors are aware of 1 full and completed implementation and 1 underway implementation of IP-TFS as defined in this document. These 2 are described below.

8.1. Reference Implementation - VPP + Strongswan

The entire IP-TFS protocol including congestion control mode has been implemented in VPP (Vector Packet Processor), and published to github with an Open Source (Apache 2) License. VPP is a highly efficient forwarding plane implemented in user-space utlizing direct control and polling of physical devices to provide high speed low-latency forwarding in Linux. By pinning packet processing threads directly to CPU cores for their exclusive use a high degree of control is given to the protocol designer.

The IKEv2 additions were implemented in Strongswan and are licensed using the GNU public license used by the Strongswan project.

Finally, an extensive automation suite was also created and is included with the open source implementation, which tests the functionality as well as the performance of the implementation, and most importantly verifies, through precise timing tracing and time-stamping, the decoupling of the users offered load from the tunnel packets (i.e., the Traffic Flow Security).

The verification process utilized the TREX packet generator for high bandwidth testing as well as other tools such as iperf. The test hardware included large servers with 10GE, 40GE and 100GE network interfaces, as well as small SoC (system on a chip) network appliances, and also cloud deployments.

Tested IP-TFS tunnel rates ranged from 10M all the way to 10GE on the small network appliance, for the large servers multiple 10GE tunnel rates were tested as well.

Offered loads included partial, full and oversubscribed bandwidths from various flow types consisting of small packets, large packets, random sized packets, sequential sized packets, and multiple IMIX variations sized flows. Timing analysis was done with variable rate traffic, impulse traffic and random bursty traffic.

The quality of the reference implementation should be considered production level as it underwent extensive testing and verification.

The organization responsible for this implementation is LabN Consulting, L.L.C.

URLs to the implementation follow.

* VPP+IPTFS, iptfs plugin

* Strongswan IKEv2

The implementation was last updated April, 2021.

8.2. In Progress Linux Kernel Implementation.

A second open source implementation has begun by LabN Consulting L.L.C., within the Linux IPsec xfrm stack. Development has also been coordinated with the Linux IPsec community, and was being worked by the same during the most recent IETF 114 hackathon.

Currently the quality is alpha level with aggregation-only complete and fragmentation support underway with congestion control to follow.

This implementation is licensed under the GNU public license and can be found at the following URLs

*development environment: https://github.com/LabNConsulting/iptfs-dev

*linux kernel source: https://github.com/LabNConsulting/iptfs-linux

*iproute2 source: https://github.com/LabNConsulting/iptfs-iproute2

9. Security Considerations

This document describes an aggregation and fragmentation mechanism to efficiently implement TFC for IP traffic. This approach is expected to reduce the efficacy of traffic analysis on IPsec communication. 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 in Section 3.1, 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.

As noted previously in Section 2.4.2, for TFC to be 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.

10. References

10.1. 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>.

10.2. Informative References

[AppCrypt] Schneier, B., "Applied Cryptography: Protocols, Algorithms, and Source Code in C", 1 November 2017.

[I-D.ietf-ipsecme-mib-iptfs] Fedyk, D. and E. Kinzie, "Definitions of Managed Objects for IP Traffic Flow Security", Work in Progress, Internet-Draft, draft-ietf-ipsecme-mib-iptfs-03, 18 November 2021, <https://www.ietf.org/archive/id/draft-ietf-ipsecme-mib-iptfs-03.txt>.

[I-D.ietf-ipsecme-yang-iptfs] Fedyk, D. and C. Hopps, "A YANG Data Model for IP Traffic Flow Security", Work in Progress, Internet-Draft, draft-ietf-ipsecme-yang-iptfs-10, 31 August 2022, <https://www.ietf.org/archive/id/draft-ietf-ipsecme-yang-iptfs-10.txt>.

[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>.

[RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, DOI 10.17487/RFC4821, March 2007, <https://www.rfc-editor.org/info/rfc4821>.

[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>.

[RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, DOI 10.17487/RFC6040, November 2010, <https://www.rfc-editor.org/info/rfc6040>.

[RFC7120] Cotton, M., "Early IANA Allocation of Standards Track Code Points", BCP 100, RFC 7120, DOI 10.17487/RFC7120, January 2014, <https://www.rfc-editor.org/info/rfc7120>.

[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>.

[RFC7893] Stein, Y(J)., Black, D., and B. Briscoe, "Pseudowire Congestion Considerations", RFC 7893, DOI 10.17487/RFC7893, June 2016, <https://www.rfc-editor.org/info/rfc7893>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/info/rfc7942>.

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

[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>.

[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>.

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

[RFC8546] Trammell, B. and M. Kuehlewind, "The Wire Image of a Network Protocol", RFC 8546, DOI 10.17487/RFC8546, April 2019, <https://www.rfc-editor.org/info/rfc8546>.

[RFC8899] Fairhurst, G., Jones, T., Tüxen, M., Rüngeler, I., and T. Völker, "Packetization Layer Path MTU Discovery for Datagram Transports", RFC 8899, DOI 10.17487/RFC8899, September 2020, <https://www.rfc-editor.org/info/rfc8899>.

Appendix A. Example of An Encapsulated IP Packet Flow

Below, an example inner IP packet flow within the encapsulating tunnel packet stream is shown. 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.

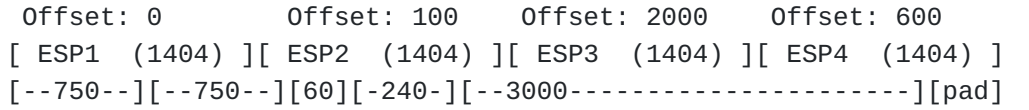

Figure 11: Inner and outer packet flow

Each outer encapsulating ESPupayload space is a fixed-size of 1404 octets the first 4 octets of which contains the AGGFRAG header. The encapsulated IP packet flow (lengths include IP header and payload) is as follows: a 750-octet packet, a 750-octet packet, a 60-octet packet, a 240-octet packet, a 3000-octet packet.

The BlockOffset values in the 4 AGGFRAG payload headers for this packet flow would thus be: 0, 100, 2000, 600 respectively. The first encapsulating packet (ESP1) has a zero BlockOffset which points at the IP data block immediately following the AGGFRAG header. The following packet's (ESP2) 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 3000-octet data block so

the offset points past its end and into the fourth encapsulating packet. The fourth packet's (ESP4) offset is 600, pointing at the padding which follows the completion of the continued 3000-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 this IP-TFS use case (as with [RFC4342]), the (fixed) packet size is used as the segment size for the algorithm. The main formula in the algorithm for the send rate is then as follows:

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

Where X 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).

In addition, the algorithm in [RFC5348] also uses an X_recv value (the receiver's receive rate). For IP-TFS one MAY set this value according to the sender's current tunnel send-rate (X).

The IP-TFS receiver, having the RTT estimate from the sender can use the same method as described in [RFC5348] and [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 AGGFRAG_PAYLOAD payload header field LossEventRate.

The IP-TFS sender now has both the R and p values and can calculate the correct sending rate. 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

For comparing overhead, the overhead of ESP for both normal and AGGFRAG tunnel packets must be calculated, and so an algorithm for encryption and authentication must be chosen. For the data below AES-GCM-256 was selected. This leads to an IP+ESP overhead of 54.

54 = 20 (IP) + 8 (ESPH) + 2 (ESPF) + 8 (IV) + 16 (ICV)

Additionally, for IP-TFS, non-congestion control AGGFRAG_PAYLOAD headers were chosen which adds 4 octets for a total overhead of 58.

C.1.1. IP-TFS Overhead

For comparison, the overhead of an AGGFRAG payload is 58 octets per outer packet. Therefore, the octet overhead per inner packet is 58 divided by the number of outer packets required (fractions allowed). The overhead as a percentage of inner packet size is a constant based on the Outer MTU size.

OH = 58 / Outer Payload Size / Inner Packet Size
OH % of Inner Packet Size = 100 * OH / Inner Packet Size
OH % of Inner Packet Size = 5800 / Outer Payload Size

Type	IP-TFS	IP-TFS	IP-TFS
MTU	576	1500	9000
PSize	518	1442	8942
40	11.20%	4.02%	0.65%
576	11.20%	4.02%	0.65%
1500	11.20%	4.02%	0.65%
9000	11.20%	4.02%	0.65%

Figure 12: 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 (54) 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 Payload 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	522	1446	8946	518	1442	8942
40	482	1406	8906	4.5	1.6	0.3
128	394	1318	8818	14.3	5.1	0.8
256	266	1190	8690	28.7	10.3	1.7
518	4	928	8428	58.0	20.8	3.4
576	576	870	8370	64.5	23.2	3.7
1442	286	4	7504	161.5	58.0	9.4
1500	228	1500	7446	168.0	60.3	9.7
8942	1426	1558	4	1001.2	359.7	58.0
9000	1368	1500	9000	1007.7	362.0	58.4

Figure 13: 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	522	1446	8946	518	1442	8942
40	1205.0%	3515.0%	22265.0%	11.20%	4.02%	0.65%
128	307.8%	1029.7%	6889.1%	11.20%	4.02%	0.65%
256	103.9%	464.8%	3394.5%	11.20%	4.02%	0.65%
518	0.8%	179.2%	1627.0%	11.20%	4.02%	0.65%
576	100.0%	151.0%	1453.1%	11.20%	4.02%	0.65%
1442	19.8%	0.3%	520.4%	11.20%	4.02%	0.65%
1500	15.2%	100.0%	496.4%	11.20%	4.02%	0.65%
8942	15.9%	17.4%	0.0%	11.20%	4.02%	0.65%
9000	15.2%	16.7%	100.0%	11.20%	4.02%	0.65%

Figure 14: 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 normal (unencrypted) Ethernet as well as normal ESP values are included.

C.3.1. Ethernet

In order to calculate the available bandwidth the per packet overhead is calculated first. The total overhead of Ethernet is 14+4 octets of header and CRC plus an additional 20 octets of framing (preamble, start, and inter-packet gap), for a total of 38 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	92	92	92	96	96	96	38	74
40	614	1538	9038	47	42	40	84	114
128	614	1538	9038	151	136	129	166	202
256	614	1538	9038	303	273	258	294	330
518	614	1538	9038	614	552	523	574	610
576	1228	1538	9038	682	614	582	614	650
1442	1842	1538	9038	1709	1538	1457	1498	1534
1500	1842	3076	9038	1777	1599	1516	1538	1574
8942	11052	10766	9038	10599	9537	9038	8998	9034
9000	11052	10766	18076	10667	9599	9096	9038	9074

Figure 15: 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	92	92	92	96	96	96	38	74
40	2.0M	0.8M	0.1M	26.4M	29.3M	30.9M	14.9M	11.0M
128	2.0M	0.8M	0.1M	8.2M	9.2M	9.7M	7.5M	6.2M
256	2.0M	0.8M	0.1M	4.1M	4.6M	4.8M	4.3M	3.8M
518	2.0M	0.8M	0.1M	2.0M	2.3M	2.4M	2.2M	2.1M
576	1.0M	0.8M	0.1M	1.8M	2.0M	2.1M	2.0M	1.9M
1442	678K	812K	138K	731K	812K	857K	844K	824K
1500	678K	406K	138K	703K	781K	824K	812K	794K
8942	113K	116K	138K	117K	131K	138K	139K	138K
9000	113K	116K	69K	117K	130K	137K	138K	137K

Figure 16: 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
	92	92	92	96	96	96	38	74
40	6.51%	2.60%	0.44%	84.36%	93.76%	98.94%	47.62%	35.09%
128	20.85%	8.32%	1.42%	84.36%	93.76%	98.94%	77.11%	63.37%
256	41.69%	16.64%	2.83%	84.36%	93.76%	98.94%	87.07%	77.58%
518	84.36%	33.68%	5.73%	84.36%	93.76%	98.94%	93.17%	87.50%
576	46.91%	37.45%	6.37%	84.36%	93.76%	98.94%	93.81%	88.62%
1442	78.28%	93.76%	15.95%	84.36%	93.76%	98.94%	97.43%	95.12%
1500	81.43%	48.76%	16.60%	84.36%	93.76%	98.94%	97.53%	95.30%
8942	80.91%	83.06%	98.94%	84.36%	93.76%	98.94%	99.58%	99.18%
9000	81.43%	83.60%	49.79%	84.36%	93.76%	98.94%	99.58%	99.18%

Figure 17: Percentage of Bandwidth on 10G Ethernet

A sometimes unexpected result of using an AGGFRAG tunnel (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 1500	ESP+Pad 9000	IP-TFS 1500	IP-TFS 9000
40	1.12 us	7.12 us	1.17 us	7.17 us
128	1.05 us	7.05 us	1.10 us	7.10 us
256	0.95 us	6.95 us	1.00 us	7.00 us
518	0.74 us	6.74 us	0.79 us	6.79 us
576	0.70 us	6.70 us	0.74 us	6.74 us
1442	0.00 us	6.00 us	0.05 us	6.05 us
1500	1.20 us	5.96 us	0.00 us	6.00 us

Figure 18: 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 and editing this work. We would also like to thank Michael Richardson, Sean Turner, Valery Smyslov and Tero Kivinen for reviews and many suggestions for improvements, as well as Joseph Touch for the transport area review and suggested improvements.

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