

ADD
Internet-Draft
Intended status: Standards Track
Expires: May 12, 2022

M. Boucadair
Orange
T. Reddy
McAfee
D. Wing
Citrix
V. Smyslov
ELVIS-PLUS
November 8, 2021

Internet Key Exchange Protocol Version 2 (IKEv2) Configuration for
Encrypted DNS
draft-btw-add-ipsecme-ike-04

Abstract

This document specifies a new Internet Key Exchange Protocol Version 2 (IKEv2) Configuration Payload Attribute Types for encrypted DNS protocols such as DNS-over-HTTPS (DoH), DNS-over-TLS (DoT), and DNS-over-QUIC (DoQ).

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 May 12, 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terminology	3
3. IKEv2 Configuration Payload Attribute Types for Encrypted DNS	3
3.1. ENCDNS_IP* Configuration Payload Attributes	3
3.2. ENCDNS_DIGEST_INFO Configuration Payload Attribute	5
4. IKEv2 Protocol Exchange	7
5. Security Considerations	8
6. IANA Considerations	9
6.1. Configuration Payload Attribute Types	9
7. Acknowledgements	9
8. References	9
8.1. Normative References	9
8.2. Informative References	10
Appendix A. Sample Deployment Scenarios	11
A.1. Roaming Enterprise Users	11
A.2. VPN Service Provider	12
A.3. DNS Offload	12
Authors' Addresses	12

1. Introduction

This document specifies encrypted DNS configuration for an Internet Key Exchange Protocol Version 2 (IKEv2) [RFC7296] initiator, particularly the Authentication Domain Name (ADN) of encrypted DNS protocols such as DNS-over-HTTPS (DoH) [RFC8484], DNS-over-TLS (DoT) [RFC7858], or DNS-over-QUIC (DoQ) [I-D.ietf-dprive-dnssoquic].

This document introduces new IKEv2 Configuration Payload Attribute Types (Section 3) for the support of encrypted DNS servers. These attributes can be used to provision authentication domain names, a list of IP addresses, and a set of service parameters.

Sample use cases are discussed in Appendix A. The Configuration Payload Attribute Types defined in this document are not specific to these deployments, but can also be used in other deployment contexts. It is out of the scope of this document to provide a comprehensive list of deployment contexts.

The encrypted DNS server hosted by the VPN provider can get a domain-validate certificate from a public CA. The VPN client does not need

to be provisioned with the root certificate of a private CA to authenticate the certificate of the encrypted DNS server. The encrypted DNS server can run on private IP addresses and its access can be restricted to clients connected to the VPN.

Note that, for many years, typical designs have often considered that the DNS server was usually located inside the protected domain, but could be located outside of it. With encrypted DNS, the latter option becomes plausible.

2. Terminology

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 uses of the terms defined in [RFC8499].

Also, this document uses of the terms defined in [RFC7296]. In particular, readers should be familiar with "initiator" and "responder" terms used in that document.

This document makes use of the following terms:

'Do53': refers to unencrypted DNS.

'Encrypted DNS': refers to a scheme where DNS messages are sent over an encrypted channel. Examples of encrypted DNS are DoT, DoH, and DoQ.

'ENCDNS_IP*': refers to any IKEv2 Configuration Payload Attribute Types defined in Section 3.1.

3. IKEv2 Configuration Payload Attribute Types for Encrypted DNS

3.1. ENCDNS_IP* Configuration Payload Attributes

The ENCDNS_IP* IKEv2 Configuration Payload Attribute Types are used to configure encrypted DNS servers. All these attributes share the format shown in Figure 1.

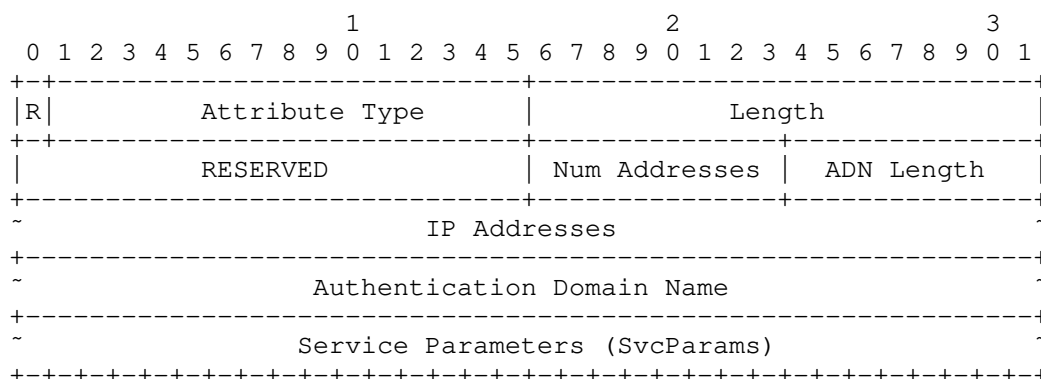


Figure 1: Attributes Format

The fields of the attribute shown in Figure 1 are as follows:

- o R (Reserved, 1 bit) - This bit MUST be set to zero and MUST be ignored on receipt (see Section 3.15.1 of [RFC7296] for details).
- o Attribute Type (15 bits) - Identifier for Configuration Attribute Type; is set to TBA1 or TBA2 values listed in Section 6.1.
- o Length (2 octets, unsigned integer) - Length of the data in octets. In particular, this field is set to:
 - * 0 if the Configuration payload has types CFG_REQUEST or CFG_ACK.
 - * (4 + Length of the ADN + N * 4 + Length of SvcParams) for ENCDNS_IP4 attributes if the Configuration payload has types CFG_REPLY or CFG_SET; N being the number of included IPv4 addresses ('Num addresses').
 - * (4 + Length of the ADN + N * 16 + Length of SvcParams) for ENCDNS_IP6 attributes if the Configuration payload has types CFG_REPLY or CFG_SET; N being the number of included IPv6 addresses ('Num addresses').
- o RESERVED (2 octets) - These bits are reserved for future use. These bits MUST be set to zero by the sender and MUST be ignored by the receiver.
- o Num Addresses (1 octet) - Indicates the number of enclosed IPv4 (for ENCDNS_IP4 attribute type) or IPv6 (for ENCDNS_IP6 attribute type) addresses. It MUST NOT be set to 0.

- o ADN Length (1 octet) - Indicates the length of the authentication-domain-name field in octets.
- o IP Address(es) (variable) - One or more IPv4 or IPv6 addresses to be used to reach the encrypted DNS server that is identified by the name in the Authentication Domain Name.
- o Authentication Domain Name (variable) - A fully qualified domain name of the encrypted DNS server following the syntax defined in [RFC5890]. The name MUST NOT contain any terminators (e.g., NULL, CR).

An example of a valid ADN for DoH server is "doh1.example.com".

- o Service Parameters (SvcParams) (variable) - Specifies a set of service parameters that are encoded following the rules in Section 2.1 of [I-D.ietf-dnsop-svcb-https]. Service parameters may include, for example, a list of ALPN protocol identifiers or alternate port numbers. The service parameters MUST NOT include "ipv4hint" or "ipv6hint" SvcParams as they are superseded by the included IP addresses.

If no port service parameter is included, this indicates that default port numbers should be used. As a reminder, the default port number is 853 for DoT and 443 for DoH.

The service parameters apply to all IP addresses in the ENCDNS_IP* Configuration Payload Attribute.

3.2. ENCDNS_DIGEST_INFO Configuration Payload Attribute

The format of ENCDNS_DIGEST_INFO configuration payload attribute is shown in Figure 2.

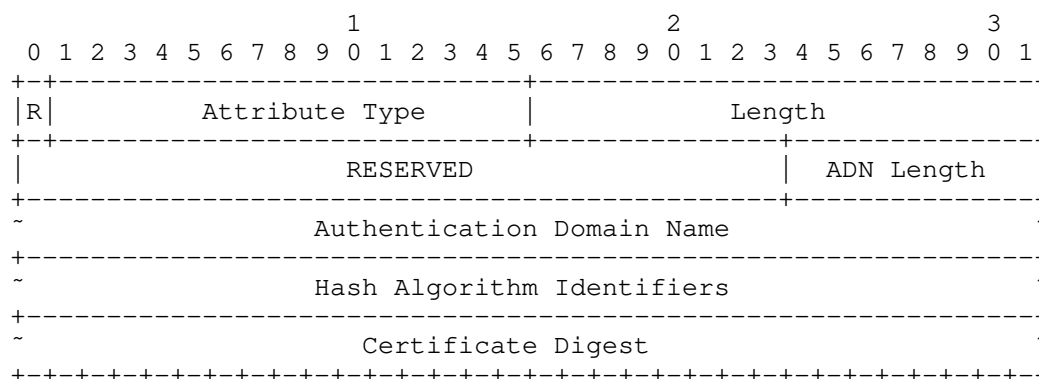


Figure 2: ENCDNS_DIGEST_INFO Attribute Format

- o R (Reserved, 1 bit) - This bit MUST be set to zero and MUST be ignored on receipt (see Section 3.15.1 of [RFC7296] for details).
- o Attribute Type (15 bits) - Identifier for Configuration Attribute Type; is set to TBA3 value listed in Section 6.1.
- o Length (2 octets, unsigned integer) - Length of the data in octets.
- o RESERVED (2 octets) - These bits are reserved for future use. These bits MUST be set to zero by the sender and MUST be ignored by the receiver.
- o ADN Length (1 octet) - Indicates the length of the authentication-domain-name field in octets. When set to '0', this means that the digest applies on the ADN conveyed in the ENCDNS_IP* Configuration Payload Attribute(s).
- o Authentication Domain Name (variable) - A fully qualified domain name of the encrypted DNS server following the syntax defined in [RFC5890]. The name MUST NOT contain any terminators (e.g., NULL, CR). A name is included only when multiple ADNs are included in the ENCDNS_IP* Configuration Payload Attributes.
- o Hash Algorithm Identifiers (variable) - In a request, this field specifies a list of 16-bit hash algorithm identifiers that are supported by the Encrypted DNS client. In a response, this field specified the 16-bit hash algorithm identifier selected by the server to generate the digest of its certificate. The values of this field are taken from the Hash Algorithm Identifiers of IANA's "Internet Key Exchange Version 2 (IKEv2) Parameters" registry [Hash].

There is no padding between the hash algorithm identifiers.

Note that SHA2-256 is mandatory to implement.

- o Certificate Digest (variable) - MUST only be present in a response. This field includes the digest of the Encrypted DNS server certificate using the algorithm identified in the 'Hash Algorithm Identifiers' field.

4. IKEv2 Protocol Exchange

This section describes how an initiator can be configured with an encrypted DNS server (e.g., DoH, DoT) using IKEv2.

Initiators indicate the support of an encrypted DNS in the CFG_REQUEST payloads by including one or two ENCDNS_IP* attributes, while responders supply the encrypted DNS configuration in the CFG_REPLY payloads. Concretely:

If the initiator supports encrypted DNS, it includes one or two ENCDNS_IP* attributes in the CFG_REQUEST. For each IP address family the initiator MUST include exactly one attribute with the Length field set to 0, so that no data is included for these attributes. The initiator MAY include the ENCDNS_DIGEST_INFO attribute with a list of hash algorithms that are supported by the Encrypted DNS client.

For each ENCDNS_IP* attribute from the CFG_REQUEST, if the responder supports the corresponding address family, and absent any policy, the responder sends back ENCDNS_IP* attribute(s) in the CFG_REPLY with an appropriate list of IP addresses, service parameters, and an ADN. The list of IP addresses MUST include at least one IP address. Multiple instances of the same ENCDNS_IP* attribute MAY be returned if distinct ADNs or service parameters are to be returned by the responder. The same or distinct IP addresses can be returned in such instances. In addition, the responder MAY return the ENCDNS_DIGEST_INFO attribute to convey a digest of the certificate of the Encrypted DNS and the identifier of the hash algorithm that is used to generate the digest.

The behavior of the responder if it receives both ENCDNS_IP* and INTERNAL_IP6_DNS (or INTERNAL_IP4_DNS) attributes is policy-based and deployment-specific. However, it is RECOMMENDED that if the responder includes at least one ENCDNS_IP* attribute in the reply, it should not include any of INTERNAL_IP4_DNS/INTERNAL_IP6_DNS attributes.

If the CFG_REQUEST includes an ENCDNS_IP* attribute but the CFG_REPLY does not include an ENCDNS_IP* matching the requested address family, this is an indication that requested address family is not supported by the responder or the responder is not configured to provide corresponding server addresses.

The DNS client establishes an encrypted DNS session (e.g., DoT, DoH, DoQ) with the address(es) conveyed in ENCDNS_IP* and uses the mechanism discussed in Section 8 of [RFC8310] to authenticate the DNS server certificate using the authentication domain name conveyed in ENCDNS_IP*.

If the CFG_REPLY includes an ENCDNS_DIGEST_INFO attribute, the DNS client has to create a digest of the DNS server certificate received in the TLS handshake using the negotiated hash algorithm in the ENCDNS_DIGEST_INFO attribute. If the computed digest for an ADN matches the one sent in the ENCDNS_DIGEST_INFO attribute, the encrypted DNS server certificate is successfully validated. If so, the client continues with the TLS connection as normal. Otherwise, the client MUST treat the server certificate validation failure as a non-recoverable error. This approach is similar to certificate usage PKIX-EE(1) defined in [RFC7671].

If the IPsec connection is a split-tunnel configuration and the initiator negotiated INTERNAL_DNS_DOMAIN as per [RFC8598], the DNS client MUST resolve the internal names using ENCDNS_IP* DNS servers.

Note: [RFC8598] requires INTERNAL_IP6_DNS (or INTERNAL_IP4_DNS) attribute to be mandatory present when INTERNAL_DNS_DOMAIN is included. This specification relaxes that constraint in the presence of ENCDNS_IP* attributes.

5. Security Considerations

This document adheres to the security considerations defined in [RFC7296]. In particular, this document does not alter the trust on the DNS configuration provided by a responder.

Networks are susceptible to internal attacks as discussed in Section 3.2 of [I-D.arkko-farrell-arch-model-t]. Hosting encrypted DNS server even in case of split-VPN configuration minimizes the attack vector (e.g., a compromised network device cannot monitor/modify DNS traffic). This specification describes a mechanism to restrict access to the DNS messages to only the parties that need to know.

The initiator may trust the encrypted DNS servers supplied by means of IKEv2 from a trusted responder more than the locally provided DNS

servers, especially in the case of connecting to unknown or untrusted networks (e.g., coffee shops or hotel networks).

If IKEv2 responder has used NULL Authentication method [RFC7619] to authenticate itself, the initiator MUST NOT use returned ENCDNS_IP* servers configuration unless it is pre-configured in the OS or the browser.

This specification does not extend the scope of accepting DNSSEC trust anchors beyond the usage guidelines defined in Section 6 of [RFC8598].

6. IANA Considerations

6.1. Configuration Payload Attribute Types

This document requests IANA to assign the following new IKEv2 Configuration Payload Attribute Types from the "IKEv2 Configuration Payload Attribute Types" namespace available at <https://www.iana.org/assignments/ikev2-parameters/ikev2-parameters.xhtml#ikev2-parameters-21>.

Value	Attribute Type	Multi-Valued	Length	Reference
TBA1	ENCDNS_IP4	YES	0 or more	RFC XXXX
TBA2	ENCDNS_IP6	YES	0 or more	RFC XXXX
TBA3	ENCDNS_ENCDNS_DIGEST_INFO	YES	0 or more	RFC XXXX

7. Acknowledgements

Many thanks to Yoav Nir, Christian Jacquenet, Paul Wouters, and Tommy Pauly for the review and comments.

Yoav and Paul suggested the use of one single attribute carrying both the name and an IP address instead of depending on the existing INTERNAL_IP6_DNS and INTERNAL_IP4_DNS attributes.

Christian Huitema suggested to return a port number in the attributes.

8. References

8.1. Normative References

[Hash] "IKEv2 Hash Algorithms",
<<https://www.iana.org/assignments/ikev2-parameters/ikev2-parameters.xhtml#hash-algorithms>>.

- [I-D.ietf-dnsop-svcb-https]
Schwartz, B., Bishop, M., and E. Nygren, "Service binding and parameter specification via the DNS (DNS SVCB and HTTPS RRs)", draft-ietf-dnsop-svcb-https-08 (work in progress), October 2021.
- [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>>.
- [RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <<https://www.rfc-editor.org/info/rfc5890>>.
- [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>>.
- [RFC8310] Dickinson, S., Gillmor, D., and T. Reddy, "Usage Profiles for DNS over TLS and DNS over DTLS", RFC 8310, DOI 10.17487/RFC8310, March 2018, <<https://www.rfc-editor.org/info/rfc8310>>.

8.2. Informative References

- [I-D.arkko-farrell-arch-model-t]
Arkko, J. and S. Farrell, "Challenges and Changes in the Internet Threat Model", draft-arkko-farrell-arch-model-t-04 (work in progress), July 2020.
- [I-D.ietf-dprive-dnsquic]
Huitema, C., Dickinson, S., and A. Mankin, "DNS over Dedicated QUIC Connections", draft-ietf-dprive-dnsquic-06 (work in progress), October 2021.
- [RFC7619] Smyslov, V. and P. Wouters, "The NULL Authentication Method in the Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 7619, DOI 10.17487/RFC7619, August 2015, <<https://www.rfc-editor.org/info/rfc7619>>.

- [RFC7671] Dukhovni, V. and W. Hardaker, "The DNS-Based Authentication of Named Entities (DANE) Protocol: Updates and Operational Guidance", RFC 7671, DOI 10.17487/RFC7671, October 2015, <<https://www.rfc-editor.org/info/rfc7671>>.
- [RFC7858] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "Specification for DNS over Transport Layer Security (TLS)", RFC 7858, DOI 10.17487/RFC7858, May 2016, <<https://www.rfc-editor.org/info/rfc7858>>.
- [RFC8484] Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", RFC 8484, DOI 10.17487/RFC8484, October 2018, <<https://www.rfc-editor.org/info/rfc8484>>.
- [RFC8499] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", BCP 219, RFC 8499, DOI 10.17487/RFC8499, January 2019, <<https://www.rfc-editor.org/info/rfc8499>>.
- [RFC8598] Pauly, T. and P. Wouters, "Split DNS Configuration for the Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 8598, DOI 10.17487/RFC8598, May 2019, <<https://www.rfc-editor.org/info/rfc8598>>.

Appendix A. Sample Deployment Scenarios

A.1. Roaming Enterprise Users

In this Enterprise scenario (Section 1.1.3 of [RFC7296]), a roaming user connects to the Enterprise network through an IPsec tunnel. The split-tunnel Virtual Private Network (VPN) configuration allows the endpoint to access hosts that resides in the Enterprise network [RFC8598] using that tunnel; other traffic not destined to the Enterprise does not traverse the tunnel. In contrast, a non-split-tunnel VPN configuration causes all traffic to traverse the tunnel into the enterprise.

For both split- and non-split-tunnel configurations, the use of encrypted DNS instead of Do53 provides privacy and integrity protection along the entire path (rather than just to the VPN termination device) and can communicate the encrypted DNS server policies.

For split-tunnel VPN configurations, the endpoint uses the Enterprise-provided encrypted DNS server to resolve internal-only domain names.

For non-split-tunnel VPN configurations, the endpoint uses the Enterprise-provided encrypted DNS server to resolve both internal and external domain names.

Enterprise networks are susceptible to internal and external attacks. To minimize that risk all enterprise traffic is encrypted (Section 2.1 of [I-D.arkko-farrell-arch-model-t]).

A.2. VPN Service Provider

Legacy VPN service providers usually preserve end-users' data confidentiality by sending all communication traffic through an encrypted tunnel. A VPN service provider can also provide guarantees about the security of the VPN network by filtering malware and phishing domains.

Browsers and OSes support DoH/DoT; VPN providers may no longer expect DNS clients to fallback to Do53 just because it is a closed network.

The encrypted DNS server hosted by the VPN service provider can be securely discovered by the endpoint using the IKEv2 Configuration Payload Attribute Type.

A.3. DNS Offload

VPN service providers typically allow split-tunnel VPN configuration in which users can choose applications that can be excluded from the tunnel. For example, users may exclude applications that restrict VPN access.

The encrypted DNS server hosted by the VPN service provider can be securely discovered by the endpoint using the IKEv2 Configuration Payload Attribute Type.

Authors' Addresses

Mohamed Boucadair
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Tirumaleswar Reddy
McAfee, Inc.
Embassy Golf Link Business Park
Bangalore, Karnataka 560071
India

Email: TirumaleswarReddy_Konda@McAfee.com

Dan Wing
Citrix Systems, Inc.
USA

Email: dwing-ietf@fuggles.com

Valery Smyslov
ELVIS-PLUS
RU

Email: svan@elvis.ru

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 12, 2022

C. Hopps
LabN Consulting, L.L.C.
November 8, 2021

IP-TFS: Aggregation and Fragmentation Mode for ESP and its Use for IP
Traffic Flow Security
draft-ietf-ipsecme-iptfs-12

Abstract

This document describes a mechanism for aggregation and fragmentation of IP packets when they are being encapsulated in ESP payload. 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 May 12, 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Terminology & Concepts	4
2. The AGGFRAG Tunnel	4
2.1. Tunnel Content	4
2.2. Payload Content	5
2.2.1. Data Blocks	6
2.2.2. End Padding	6
2.2.3. Fragmentation, Sequence Numbers and All-Pad Payloads	6
2.2.4. Empty Payload	8
2.2.5. IP Header Value Mapping	8
2.2.6. IP Time-To-Live (TTL) and Tunnel errors	9
2.2.7. Effective MTU of the Tunnel	9
2.3. Exclusive SA Use	10
2.4. Modes of Operation	10
2.4.1. Non-Congestion Controlled Mode	10
2.4.2. Congestion Controlled Mode	10
2.5. Summary of Receiver Processing	12
3. Congestion Information	12
3.1. ECN Support	14
4. Configuration of AGGFRAG Tunnels for IP-TFS	14
4.1. Bandwidth	14
4.2. Fixed Packet Size	14
4.3. Congestion Control	14
5. IKEv2	15
5.1. USE_AGGFRAG Notification Message	15
6. Packet and Data Formats	15
6.1. AGGFRAG_PAYLOAD Payload	15
6.1.1. Non-Congestion Control AGGFRAG_PAYLOAD Payload Format	16
6.1.2. Congestion Control AGGFRAG_PAYLOAD Payload Format	17
6.1.3. Data Blocks	19
6.1.4. IKEv2 USE_AGGFRAG Notification Message	20
7. IANA Considerations	21
7.1. AGGFRAG_PAYLOAD Sub-Type Registry	21
7.2. USE_AGGFRAG Notify Message Status Type	21
8. Security Considerations	22
9. References	22
9.1. Normative References	22
9.2. Informative References	22
Appendix A. Example Of An Encapsulated IP Packet Flow	24

Appendix B. A Send and Loss Event Rate Calculation	25
Appendix C. Comparisons of IP-TFS	26
C.1. Comparing Overhead	26
C.1.1. IP-TFS Overhead	26
C.1.2. ESP with Padding Overhead	26
C.2. Overhead Comparison	27
C.3. Comparing Available Bandwidth	28
C.3.1. Ethernet	28
Appendix D. Acknowledgements	30
Appendix E. Contributors	30
Author's Address	31

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 traffic pattern itself exposes 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 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 is comprised 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.

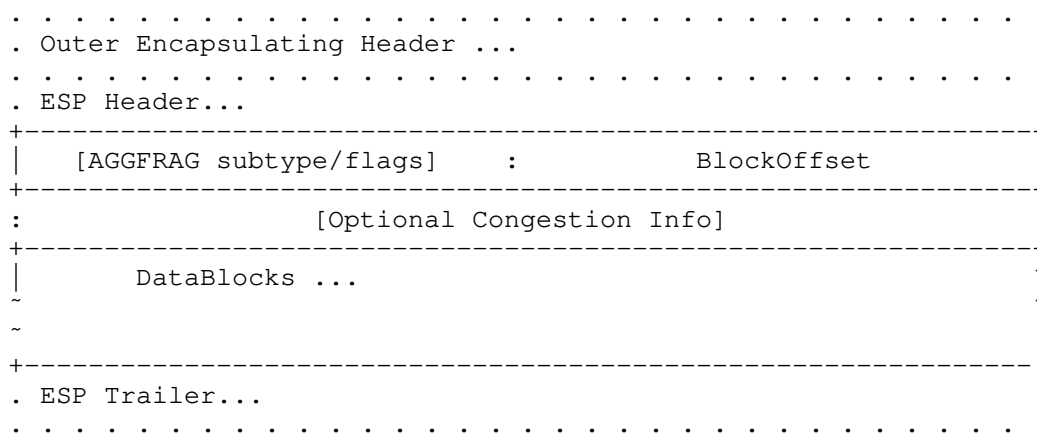


Figure 1: Layout of an AGGFRAG mode IPsec Packet

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

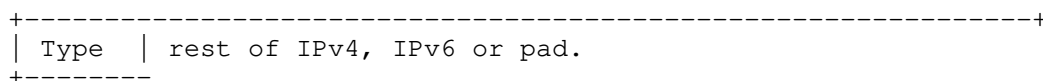


Figure 2: Layout of a DataBlock

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 may normally experience.

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 amount of 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, one 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 with no gaps (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.

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].

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. IP Time-To-Live (TTL) and Tunnel errors

[RFC4301] specifies how to modify the inner packet TTL [RFC0791].

Any errors (e.g., ICMP errors arriving back at the tunnel ingress due to tunnel traffic) are handled the same as with non-AGGFRAG IPsec tunnels.

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 tunnels, AGGFRAG tunnels are unidirectional. Bidirectional IP-TFS functionality is achieved by setting up 2 AGGFRAG tunnels, 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 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.

Non-congestion control mode is also appropriate if ESP over TCP is in use [RFC8229].

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 packet 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 addition 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.

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 the 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 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 in-order AGGFRAG_PAYLOAD payload stream, the receiver transmits the inner-packets. Using this method will make sure the packets are sent in-order, i.e., there is no reordering possible, but the cost is that a lost packet will cause delay of up to the lost packet timer interval (or the full reorder window if no lost packet timer is used), and there will be extra burstiness in the output stream (when lost packet is dropped out from the re-order window, all outer packets received after that are then 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.

3. Congestion Information

In order to support the congestion control 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 it's 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 CC 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 senders 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.

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 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.

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: 0x5

An AGGFRAG payload is identified by the ESP Next Header value AGGFRAG_PAYLOAD which has the value 0x5. The value 5 was chosen to not conflict with other used values. The first octet of this payload indicates the format of the remaining payload data.

```

  0 1 2 3 4 5 6 7
+-+--+--+--+--+--+--+
|  Sub-type    | ...
+-+--+--+--+--+--+--+

```

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 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
+-+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|  Sub-Type (0) |  Reserved   |          BlockOffset          |
+-+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|          DataBlocks ...
+-+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+

```

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:

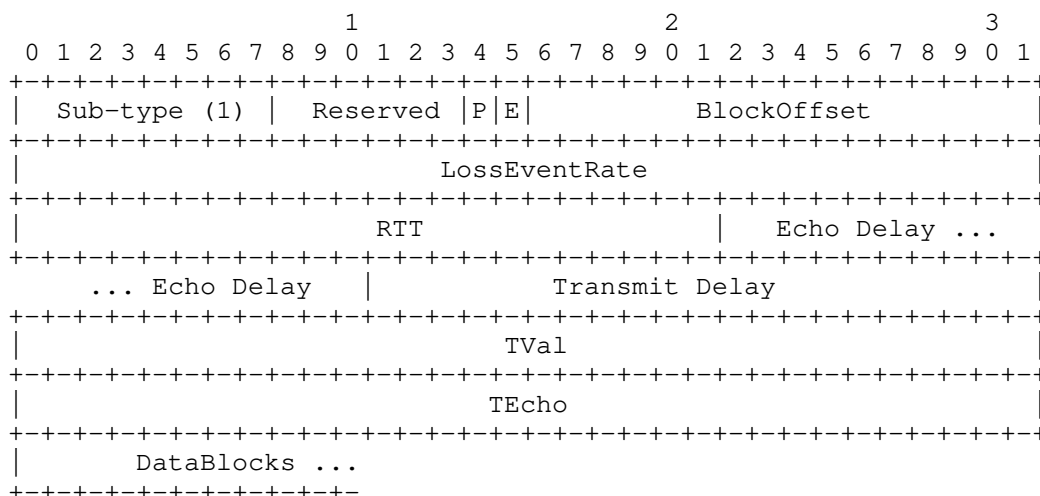
A 16-bit unsigned integer counting the number of octets of "DataBlocks" data before the start of a new data block. If the start of a new data block occurs in a subsequent payload the "BlockOffset" will point past the end of the "DataBlocks" data. In this case all the "DataBlocks" data belongs to the 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 is comprised of a 24 octet header followed by a variable amount of "DataBlocks" data as shown below.

**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 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 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 value is equal to or larger than "0x3FFFFFF" it MUST be set to "0x3FFFFFF".

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 value is equal to or larger than "0x1FFFFFF" it MUST be set to "0x1FFFFFF".

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 value is equal to or larger than "0x1FFFFFF" it MUST be set to "0x1FFFFFF".

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 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 "Total 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 |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| PayloadLength | 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).

PayloadLength:

The 16-bit unsigned integer "Payload 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.

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 (TBD2).

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

```

+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|0|0|0|0|0|0|C|D|
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+

```

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. 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	

7.2. 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.

Value:

TBD2

Name:

USE_AGGFRAG

Reference:

This document

8. Security Considerations

This document describes an aggregation and fragmentation mechanism and its use to add TFC to IP traffic. The use described 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 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 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.

- [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>>.
- [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>>.
- [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., Tuexen, M., Ruengeler, I., and T. Voelker, "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.

```

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 AGGFRAG 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 AGGFRAG 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 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:

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

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.

$$54 = 20 \text{ (IP)} + 8 \text{ (ESPH)} + 2 \text{ (ESPF)} + 8 \text{ (IV)} + 16 \text{ (ICV)}$$

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 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 (fractional allowed). The overhead as a percentage of inner packet size is a constant based on the Outer MTU size.

$$\begin{aligned} \text{OH} &= 58 / \text{Outer Payload Size} / \text{Inner Packet Size} \\ \text{OH \% of Inner Packet Size} &= 100 * \text{OH} / \text{Inner Packet Size} \\ \text{OH \% of Inner Packet Size} &= 5800 / \text{Outer Payload Size} \end{aligned}$$

Type	IP-TFS	IP-TFS	IP-TFS
MTU	576	1500	9000
PSize	518	1442	8942
<hr/>			
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 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 (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

$$NF0 = \frac{\text{Inner Payload Size}}{\text{Outer Payload Size} - \text{IP Overhead}}$$

NF = CEILING(NF0)

$$\begin{aligned} OH &= NF * (\text{IP Overhead} + \text{IPsec Overhead}) \\ &\quad - \text{IP Overhead} \\ &\quad + NF * (\text{Outer Payload Size} - \text{IP Overhead}) \\ &\quad - \text{Inner Payload Size} \end{aligned}$$

$$\begin{aligned} OH &= NF * (\text{IPsec Overhead} + \text{Outer Payload Size}) \\ &\quad - (\text{IP Overhead} + \text{Inner Payload Size}) \end{aligned}$$

$$\begin{aligned} OH &= NF * (\text{IPsec Overhead} + \text{Outer Payload Size}) \\ &\quad - \text{Inner Packet Size} \end{aligned}$$

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
<hr/>						
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 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	522	1446	8946	518	1442	8942
<hr/>						
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 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 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 and 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	PTFS	PTFS	PTFS	Enet	ESP
MTU	590	1514	9014	590	1514	9014	any	any
OH	92	92	92	96	96	96	38	74
<hr/>								
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 7: L2 Octets Per Packet

Size	E + P	E + P	E + P	PTFS	PTFS	PTFS	Enet	ESP
MTU	590	1514	9014	590	1514	9014	any	any
OH	92	92	92	96	96	96	38	74
<hr/>								
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 8: Packets Per Second on 10G Ethernet

Size	E + P	E + P	E + P	PTFS	PTFS	PTFS	Enet	ESP
	590	1514	9014	590	1514	9014	any	any
	92	92	92	96	96	96	38	74
<hr/>								
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 9: 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
<hr/>				
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 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 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

Network
Internet-Draft
Updates: 7296 (if approved)
Intended status: Standards Track
Expires: 25 September 2022

P. Wouters
Aiven
S. Prasad
Red Hat
24 March 2022

Labeled IPsec Traffic Selector support for IKEv2
draft-ietf-ipsecme-labeled-ipsec-07

Abstract

This document defines a new Traffic Selector (TS) Type for Internet Key Exchange version 2 to add support for negotiating Mandatory Access Control (MAC) security labels as a traffic selector of the Security Policy Database (SPD). Security Labels for IPsec are also known as "Labeled IPsec". The new TS type is TS_SECLABEL, which consists of a variable length opaque field specifying the security label. This document updates the IKEv2 TS negotiation specified in RFC 7296 Section 2.9.

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 25 September 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components

extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	2
1.1. Requirements Language	3
1.2. Traffic Selector clarification	3
1.3. Traffic Selector update	4
2. TS_SECLABEL Traffic Selector Type	4
2.1. TS_SECLABEL payload format	4
2.2. TS_SECLABEL properties	4
3. Traffic Selector negotiation	5
3.1. Example TS negotiation	6
3.2. Considerations for using multiple TS_TYPES in a TS	6
4. Security Considerations	7
5. IANA Considerations	7
6. Implementation Status	7
6.1. Libreswan	8
7. Acknowledgements	9
8. References	9
8.1. Normative References	9
8.2. Informative References	9
Authors' Addresses	10

1. Introduction

In computer security, Mandatory Access Control usually refers to systems in which all subjects and objects are assigned a security label. A security label is comprised of a set of security attributes. The security labels along with a system authorization policy determine access. Rules within the system authorization policy determine whether the access will be granted based on the security attributes of the subject and object.

Traditionally, security labels used by Multilevel Systems (MLS) are comprised of a sensitivity level (or classification) field and a compartment (or category) field, as defined in [FIPS188] and [RFC5570]. As MAC systems evolved, other MAC models gained in popularity. For example, SELinux, a Flux Advanced Security Kernel (FLASK) implementation, has security labels represented as colon-separated ASCII strings composed of values for identity, role, and type. The security labels are often referred to as security contexts.

Traffic Selector (TS) payloads specify the selection criteria for packets that will be forwarded over the newly set up IPsec SA as enforced by the Security Policy Database (SPD, see [RFC4301]). This document updates the Traffic Selector negotiation specified in Section 2.9 of [RFC7296].

This document specifies a new Traffic Selector Type TS_SECLABEL for IKEv2 that can be used to negotiate security labels as additional selectors for the Security Policy Database (SPD) to further restrict the type of traffic allowed to be sent and received over the IPsec SA.

1.1. Requirements Language

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.

1.2. Traffic Selector clarification

The negotiation of Traffic Selectors is specified in Section 2.9 of [RFC7296] where it defines two TS Types (TS_IPV4_ADDR_RANGE and TS_IPV6_ADDR_RANGE). The Traffic Selector payload format is specified in Section 3.13 of [RFC7296]. However, the term Traffic Selector is used to denote the traffic selector payloads and individual traffic selectors of that payload. Sometimes the exact meaning can only be learned from context or if the item is written in plural ("Traffic Selectors" or "TSs"). This section clarifies these terms as follows:

A Traffic Selector (no acronym) is one selector for traffic of a specific Traffic Selector Type (TS_TYPE). For example a Traffic Selector of TS_TYPE TS_IPV4_ADDR_RANGE for UDP traffic in the IP network 198.51.100.0/24 covering all ports, is denoted as (17, 0, 198.51.100.0-198.51.100.255)

A Traffic Selector payload (TS) is a set of one or more Traffic Selectors of the same or different TS_TYPES, but MUST include at least one TS_TYPE of TS_IPV4_ADDR_RANGE or TS_IPV6_ADDR_RANGE. For example, the above Traffic Selector by itself in a TS payload is denoted as TS((17, 0, 198.51.100.0-198.51.100.255))

1.3. Traffic Selector update

The negotiation of Traffic Selectors is specified in Section 2.9 of [RFC7296] and states that the TSi/TSr payloads MUST contain at least one Traffic Selector type. This document updates the text to mean that the TSi/TSr payloads MUST contain at least one Traffic Selector of type TS_IPV4_ADDR_RANGE or TS_IPV6_ADDR_RANGE, as other Traffic Selector types can be defined that are complimentary to these Traffic Selector Types and cannot be selected on their own without TS_IPV4_ADDR_RANGE or TS_IPV6_ADDR_RANGE. The below defined TS_SECLABEL Traffic Selector Type is an example of this.

2. TS_SECLABEL Traffic Selector Type

This document defines a new TS Type, TS_SECLABEL that contains a single new opaque Security Label.

2.1. TS_SECLABEL payload format

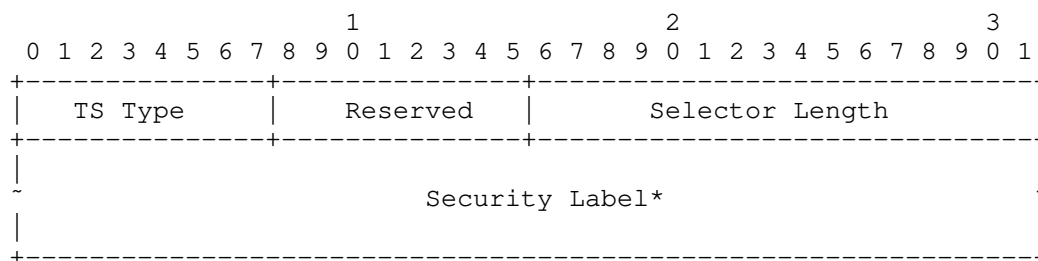


Figure 1: Labeled IPsec Traffic Selector

*Note: All fields other than TS Type and Selector Length depend on the TS Type. The fields shown is for TS Type TS_SECLABEL, the selector this document defines.

- * TS Type (one octet) - Set to 10 for TS_SECLABEL,
- * Selector Length (2 octets, unsigned integer) - Specifies the length of this Traffic Selector substructure including the header.
- * Security Label - An opaque byte stream of at least one octet.

2.2. TS_SECLABEL properties

The TS_SECLABEL Traffic Selector Type does not support narrowing or wildcards. It MUST be used as an exact match value.

The Security Label contents are opaque to the IKE implementation. That is, the IKE implementation might not have any knowledge of the meaning of this selector, other than as a type and opaque value to pass to the SPD.

A zero length Security Label MUST NOT be used. If a received TS payload contains a TS_TYPE of TS_SECLABEL with a zero length Security Label, that specific Traffic Selector MUST be ignored. If no other Traffic Selector of TS_TYPE TS_SECLABEL can be selected, a TS_UNACCEPTABLE Error Notify message MUST be returned. A zero length Security Label MUST NOT be interpreted as a wildcard security label.

If multiple Security Labels are allowed for a given IP protocol, start and end address/port match, the initiator includes all of the acceptable TS_SECLABEL's and the responder MUST select one of them.

If the Security Label traffic selector is optional from a configuration point of view, the initiator will have to choose which TS payload to attempt first. If it includes the Security Label and receives a TS_UNACCEPTABLE, it can attempt a new Child SA negotiation without that Security Label.

A responder that selected a TS with TS_SECLABEL MUST use the Security Label for all selector operations on the resulting TS. It MUST NOT select a TS_SECLABEL without using the specified Security Label, even if it deems the Security Label optional, as the initiator has indicated (and expects) that Security Label will be set for all traffic matching the negotiated TS.

3. Traffic Selector negotiation

This document updates the [RFC7296] specification as follows:

Each TS payload (TSi and TSr) MUST contain at least one TS_TYPE of TS_IPV4_ADDR_RANGE or TS_IPV6_ADDR_RANGE.

Each TS payload (TSi or TSr) MAY contain one or more other TS_TYPES, such as TS_SECLABEL.

A responder MUST create each TS response by creating one of more (narrowed or not) TS_IPV4_ADDR_RANGE or TS_IPV6_ADDR_RANGE entries, plus one of each further TS_TYPE present in the offered TS by the initiator. If this is not possible, it MUST return a TS_UNACCEPTABLE Error Notify payload.

If a specific TS_TYPE (other than TS_IPV4_ADDR_RANGE or TS_IPV6_ADDR_RANGE which are mandatory) is deemed optional, the initiator SHOULD first try to negotiate the Child SA with the TS

payload including the optional TS_TYPE. Upon receiving TS_UNACCEPTABLE, it SHOULD attempt a new Child SA negotiation using the same TS but without the optional TS_TYPE.

3.1. Example TS negotiation

An initiator could send:

```
TSi = ((17,24233,198.51.12-198.51.12),  
      (17,0,192.0.2.0-192.0.2.255),  
      (0,0,198.51.0-198.51.255),  
      TS_SECLABEL1, TS_SECLABEL2)  
  
TSr = ((17,53,203.0.113.1-203.0.113.1),  
      (17,0,203.0.113.0-203.0.113.255),  
      (0,0,203.0.113.0-203.0.113.255),  
      TS_SECLABEL1, TS_SECLABEL2)
```

Figure 2: initiator TS payloads example

The responder could answer with the following example:

```
TSi = ((0,0,198.51.0-198.51.255),  
      TS_SECLABEL1)  
  
TSr = (((0,0,203.0.113.0-203.0.113.255),  
      TS_SECLABEL1)
```

Figure 3: responder TS payloads example

3.2. Considerations for using multiple TS_TYPES in a TS

It would be unlikely that the traffic for TSi and TSr would have a different Security Label, but this specification does allow this to be specified. If the initiator does not support this, and wants to prevent the responder from picking different labels for the TSi / TSr payloads, it should attempt a Child SA negotiation with only the first Security Label first, and upon failure retry a new Child SA negotiation with only the second Security Label.

If different IP ranges can only use different specific Security Labels, than these should be negotiated in two different Child SA negotiations. If in the example above, the initiator only allows 192.0.2.0/24 with TS_SECLABEL1, and 198.51.0/24 with TS_SECLABEL2, than it MUST NOT combine these two ranges and security labels into one Child SA negotiation.

The mechanism of narrowing of Traffic Selectors with TS_IPV4_ADDR_RANGE and TS_IPV6_ADDR_RANGE does not apply to TS_SECLABEL as the Security Label itself is not interpreted and cannot be narrowed. It MUST be matched exactly. Since a rekey MUST NOT narrow down the Traffic Selectors narrower than the scope currently in use, the only valid choice of TS_SECLABEL for a rekey is the identical TS_SECLABEL that is in use by the Child SA being rekeyed. If the TS_LABEL is missing from the TS during the rekey negotiation, the negotiation MUST fail with TS_UNACCEPTABLE.

4. Security Considerations

It is assumed that the Security Label can be matched by the IKE implementation to its own configured value, even if the IKE implementation itself cannot interpret the Security Label value.

A packet that matches an SPD entry for all components except the Security Label would be treated as "not matching". If no other SPD entries match, the (mis-labeled) traffic might end up being transmitted in the clear. It is presumed that other Mandatory Access Control methods are in place to prevent mis-labeled traffic from reaching the IPsec subsystem, or that the IPsec subsystem itself would install a REJECT/DISCARD rule in the SPD to prevent unlabeled traffic otherwise matching a labeled security SPD rule from being transmitted without IPsec protection.

5. IANA Considerations

This document defines one new entry in the IKEv2 Traffic Selector Types registry:

[Note to RFC Editor (please remove before publication): This value has already been added via Early Allocation.]

Value	TS Type	Reference
-----	-----	-----
10	TS_SECLABEL	[this document]

Figure 4

6. Implementation Status

[Note to RFC Editor: Please remove this section and the reference to [RFC7942] before publication.]

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. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. 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 [RFC7942], "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".

Authors are requested to add a note to the RFC Editor at the top of this section, advising the Editor to remove the entire section before publication, as well as the reference to [RFC7942].

6.1. Libreswan

Organization: The Libreswan Project

Name: <https://lists.libreswan.org/mailman/listinfo/swan-dev/>

Description: Implementation has been released as part of libreswan version 4.4.

Level of maturity: beta

Coverage: Implements the entire draft using SELinux based labels

Licensing: GPLv2

Implementation experience: No interop testing has been done yet.
The code works as proof of concept, but is not yet production ready when using multiple different labels with on-demand kernel ACQUIRES.

Contact: Libreswan Development: swan-dev@libreswan.org

7. Acknowledgements

A large part of the introduction text was taken verbatim from [draft-jml-ipsec-ikev2-security-label] whose authors are J Latten, D. Quigley and J. Lu. Valery Smyslov provided valuable input regarding IKEv2 Traffic Selector semantics.

8. References

8.1. 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>>.
- [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>>.

8.2. Informative References

- [draft-jml-ipsec-ikev2-security-label] Latten, J., Quigley, D., and J. Lu, "Security Label Extension to IKE", 28 January 2011.
- [FIPS188] NIST, "National Institute of Standards and Technology, "Standard Security Label for Information Transfer"", Federal Information Processing Standard (FIPS) Publication 188, September 1994, <<https://csrc.nist.gov/publications/detail/fips/188/archive/1994-09-06>>.
- [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>>.
- [RFC5570] StJohns, M., Atkinson, R., and G. Thomas, "Common Architecture Label IPv6 Security Option (CALIPSO)", RFC 5570, DOI 10.17487/RFC5570, July 2009, <<https://www.rfc-editor.org/info/rfc5570>>.

[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>>.

Authors' Addresses

Paul Wouters
Aiven
Email: paul.wouters@aiven.io

Sahana Prasad
Red Hat
Email: sahana@redhat.com

Network
Internet-Draft
Updates: 7296,8221,8247 (if approved)
Intended status: Standards Track
Expires: August 25, 2021

P. Wouters, Ed.
Red Hat
February 21, 2021

Deprecation of IKEv1 and obsoleted algorithms
draft-pwouters-ikev1-ipsec-graveyard-06

Abstract

Internet Key Exchange version 1 (IKEv1) is deprecated. Accordingly, IKEv1 has been moved to Historic status. A number of old algorithms that are associated with IKEv1, and not widely implemented for IKEv2 are deprecated as well. IANA is instructed to close all IKEv1 registries.

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 August 25, 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Requirements Language	2
3. RFC 2409 to Historic	3
4. Deprecating obsolete algorithms	3
5. Security Considerations	3
6. IANA Considerations	4
7. References	5
7.1. Normative References	5
7.2. Informative References	6
Author's Address	6

1. Introduction

IKEv1 [RFC2409] and its related documents for ISAKMP [RFC2408] and IPsec DOI [RFC2407] were obsoleted by IKEv2 [RFC4306] in December 2005. The latest version of IKEv2 at the time of writing was published in 2014 in [RFC7296]. The Internet Key Exchange (IKE) version 2 has replaced version 1 over 15 years ago. IKEv2 has now seen wide deployment and provides a full replacement for all IKEv1 functionality. No new modifications or new algorithms have been accepted for IKEv1 for at least a decade. IKEv2 addresses various issues present in IKEv1, such as IKEv1 being vulnerable to amplification attacks. IKEv1 has been moved to Historic status, and this document requests IANA to close all IKEv1 registries.

Algorithm implementation requirements and usage guidelines for IKEv2 [RFC8247] and ESP/AH [RFC8223] gives guidance to implementors but limits that guidance to avoid broken or weak algorithms. It does not deprecate algorithms that have aged and are not in use, but leave these algorithms in a state of "MAY be used". This document deprecates those algorithms that are no longer advised but for which there are no known attacks resulting in their earlier deprecation.

2. Requirements Language

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.

3. RFC 2409 to Historic

IKEv1 is deprecated. Systems running IKEv1 should be upgraded and reconfigured to run IKEv2. Systems that support IKEv1 but not IKEv2 are most likely also unsuitable candidates for continued operation. Such unsupported systems have a much higher chance of containing an implementation vulnerability that will never be patched. IKEv1 systems can be abused for packet amplification attacks. IKEv1 systems most likely do not support modern algorithms such as AES-GCM or CHACHA20_POLY1305 and quite often only support or have been configured to use the very weak Diffie-Hellman Groups 2 and 5. IKEv1 systems should be upgraded or replaced by IKEv2 systems.

IKEv1 and its way of using Preshared Keys (PSKs) protects against quantum computer based attacks. IKEv2 updated its use of PSK to improve the error reporting, but at the expense of post-quantum security. If post-quantum security is required, these systems should be migrated to use IKEv2 Postquantum Preshared Keys (PPK) [RFC8784]

Some IKEv1 implementations support Labeled IPsec, a method to negotiate an additional Security Context selector to the SPD, but this method was never standardized in IKEv1. Those IKEv1 systems that require Labeled IPsec should migrate to an IKEv2 system supporting Labeled IPsec as specified in [draft-ietf-ipsecme-labeled-ipsec].

4. Deprecating obsolete algorithms

This document deprecates the following algorithms:

- o Encryption Algorithms: RC5, IDEA, CAST, Blowfish, and the unspecified 3IDEA, ENCR_DES_IV64 and ENCR_DES_IV32
- o PRF Algorithms: the unspecified PRF_HMAC_TIGER
- o Integrity Algorithms: HMAC-MD5-128
- o Diffie-Hellman groups: none

5. Security Considerations

There are only security benefits by deprecating IKEv1 for IKEv2.

The deprecated algorithms have long been in disuse and are no longer actively deployed or researched. It presents an unknown security risk that is best avoided. Additionally, these algorithms not being supported in implementations simplifies those implementations and reduces the accidental use of these deprecated algorithms through misconfiguration or downgrade attacks.

6. IANA Considerations

This document instructs IANA to mark all IKEv1 registries as DEPRECATED.

Additionally, this document instructs IANA to add an additional Status column to the IKEv2 Transform Type registries and mark the following entries as DEPRECATED:

Transform Type 1 - Encryption Algorithm IDs

Number	Name	Status
-----	-----	-----
1	ENCR_DES_IV64	DEPRECATED [this document]
2	ENCR_DES	DEPRECATED [RFC8247]
4	ENCR_RC5	DEPRECATED [this document]
5	ENCR_IDEA	DEPRECATED [this document]
6	ENCR_CAST	DEPRECATED [this document]
7	ENCR_BLOWFISH	DEPRECATED [this document]
8	ENCR_3IDEA	DEPRECATED [this document]
9	ENCR_DES_IV32	DEPRECATED [this document]

Figure 1

Transform Type 2 - Pseudorandom Function Transform IDs

Number	Name	Status
-----	-----	-----
1	PRF_HMAC_MD5	DEPRECATED [RFC8247]
1	PRF_HMAC_TIGER	DEPRECATED [this document]

Figure 2

Transform Type 3 - Integrity Algorithm Transform IDs

Number	Name	Status
-----	-----	-----
1	AUTH_HMAC_MD5_96	DEPRECATED [RFC8247]
3	AUTH_DES_MAC	DEPRECATED [RFC8247]
4	AUTH_KPDK_MD5	DEPRECATED [RFC8247]
6	AUTH_HMAC_MD5_128	DEPRECATED [this document]
7	AUTH_HMAC_SHA1_160	DEPRECATED [this document]

Figure 3

Transform Type 4 - Diffie Hellman Group Transform IDs

Number	Name	Status
-----	-----	-----
1	768-bit MODP Group	DEPRECATED [RFC8247]
22	1024-bit MODP Group with 160-bit Prime Order Subgroup	DEPRECATED [RFC8247]

Figure 4

All entries not mentioned here should receive no value in the new Status field.

This document instructs IANA to close and mark as obsolete the Internet Key Exchange (IKE) Attributes registries as well as the "Magic Numbers" for ISAKMP Protocol registries.

The IESG is requested to designate IKEv1 to Historic.

7. References

7.1. 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>>.
- [RFC2407] Piper, D., "The Internet IP Security Domain of Interpretation for ISAKMP", RFC 2407, DOI 10.17487/RFC2407, November 1998, <<https://www.rfc-editor.org/info/rfc2407>>.
- [RFC2408] Maughan, D., Schertler, M., Schneider, M., and J. Turner, "Internet Security Association and Key Management Protocol (ISAKMP)", RFC 2408, DOI 10.17487/RFC2408, November 1998, <<https://www.rfc-editor.org/info/rfc2408>>.
- [RFC2409] Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, DOI 10.17487/RFC2409, November 1998, <<https://www.rfc-editor.org/info/rfc2409>>.
- [RFC4306] Kaufman, C., Ed., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, DOI 10.17487/RFC4306, December 2005, <<https://www.rfc-editor.org/info/rfc4306>>.

- [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>>.
- [RFC8223] Esale, S., Torvi, R., Jalil, L., Chunduri, U., and K. Raza, "Application-Aware Targeted LDP", RFC 8223, DOI 10.17487/RFC8223, August 2017, <<https://www.rfc-editor.org/info/rfc8223>>.
- [RFC8247] Nir, Y., Kivinen, T., Wouters, P., and D. Migault, "Algorithm Implementation Requirements and Usage Guidance for the Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 8247, DOI 10.17487/RFC8247, September 2017, <<https://www.rfc-editor.org/info/rfc8247>>.
- [RFC8784] Fluhrer, S., Kampanakis, P., McGrew, D., and V. Smysov, "Mixing Preshared Keys in the Internet Key Exchange Protocol Version 2 (IKEv2) for Post-quantum Security", RFC 8784, DOI 10.17487/RFC8784, June 2020, <<https://www.rfc-editor.org/info/rfc8784>>.

7.2. Informative References

- [draft-ietf-ipsecme-labeled-ipsec] Wouters, P. and S. Prasad, "Labeled IPsec Traffic Selector support for IKEv2", draft-ietf-ipsecme-labeled-ipsec (work in progress), March 2019.

Author's Address

Paul Wouters (editor)
Red Hat

Email: pwouters@redhat.com

Network
Internet-Draft
Intended status: Standards Track
Expires: August 26, 2021

A. Antony
S. Klassert
secunet
P. Wouters
Red Hat
February 22, 2021

IKEv2 support for per-queue Child SAs
draft-pwouters-multi-sa-performance-01

Abstract

This document defines two Notification Payloads for the Internet Key Exchange Protocol Version 2 (IKEv2): NUM_QUEUES and QUEUE_INFO. These payloads add support for indicating that the negotiating of multiple identical Child SAs are to be used to optimize performance based on the number of queues or CPUs, or to create multiple Child SAs for different Quality of Service (QoS) levels. It indicates that a newer identical Child SA should not be interpreted as a replacement Child SA.

Using multiple identical Child Sa's has the benefit that each stream has its own Sequence Number, ensuring that CPU's don't have to synchronize their crypto state or disable their packet replay detection.

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 August 26, 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Requirements Language	3
2. Performance bottlenecks	3
3. Negotiation of performance specific Child SAs	3
4. Implementation specifics	4
4.1. One CPU per Child	4
4.2. QoS Child SA's	6
5. Payload Format	6
5.1. NUM_QUEUES Notify Payload	7
5.2. QUEUE_INFO Notify Payload	7
6. Security Considerations	8
7. Implementation Status	8
7.1. Linux XFRM	8
7.2. Libreswan	9
7.3. strongSWAN	9
8. IANA Considerations	10
9. References	10
9.1. Normative References	10
9.2. Informative References	10
Authors' Addresses	11

1. Introduction

IPsec implementations are currently limited to using one queue or CPU per Child SA. The result is that a machine with many queues/CPU's is limited to only using one these per Child SA. This severely limits the speeds that can be obtained. An unencrypted link of 10gbps or more is commonly reduced to 2-3gbps when IPsec is used to encrypt the link, for example when using AES-GCM.

Furthermore IPsec implementations are currently limited to use the same Child SA for all Quality of Service (QoS) types because the QoS type is not a part of the TS. The result is that IPsec can't do active Quality of Service prioritizing without disabling the anti replay detection.

While this could be mitigated by setting up multiple narrowed Child SA's, for example using Populate From Packet (PFP) as specified in [RFC4301], this IPsec feature is not widely implemented.

To make better use of multiple network queues and CPUs, it can be beneficial to negotiate and install multiple identical Child SAs. IKEv2 [RFC7296] already allows installing multiple identical Child SAs, but often implementations will assume the older Child SA is being replaced by the newer Child SA, even when no INITIAL_CONTACT notify payload was received.

When two IKEv2 peers want to negotiate multiple Child SAs, it is useful to be able to convey how many Child SAs are required for optimized traffic. This avoids triggering CREATE_CHILD_SA exchanges that will only be rejected by the peer.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Performance bottlenecks

Currently, most IPsec implementations are limited by using one CPU or network queue per Child SA. There are a number of practical reasons for this, but a key limitation is that sharing the AEAD state, counters and sequence numbers between multiple CPUs is not feasible without a significant performance penalty. There is a need to negotiate and establish multiple Child SA's with identical TSi/TSr on a per-queue or per-CPU basis.

3. Negotiation of performance specific Child SAs

The number of Child SA's notify payload refers to the number of instances for this particular TSi/TSr combination beyond the initial Child SA. Both peers send their minimum number of Child SAs they prefer to install. Both peers pick the maximum of the two numbers (within reason). That is if one peer prefers 16 and the other peer prefers 48, then the number negotiated is 48. If a 49th Child SA is

attempted with QUEUE_INFO notify payload, it can be rejected using TS_UNACCEPTABLE.

The NUM_QUEUES Notify payload is sent as part of the IKE_AUTH or as part of an CREATE_CHILD_SA Exchange for an initial new Child SA request. It identifies the initial Child SA of a set, and allows the peers to ensure that the initial Child SA (or its rekeyed version) remains active for the lifetime of the IPsec connection. Further CREATE_CHILD_SA messages for subsequent copies of the original Child SA MUST NOT contain the NUM_QUEUES notify payload. This initial Child SA (or its REKEYed successor) MUST remain active for the lifetime of the IPsec session to ensure there is always a CHILD SA that can be selected to send traffic over. Subsequent Child SA's can be installed with an additional selector, such as CPU or queue, or ToS value.

The QUEUE_INFO Notify MUST be sent in CREATE_CHILD_SA for subsequent copies of the original Child SA. It is used to indicate the queue or CPU or QoS value of this specific copy of the initial Child SA. These additional Child SA's can be started on-demand or all at once and can also be deleted if a peer deems this specific queue or CPU or QoS value to be idle. During CREATE_CHILD_SA's sent for Child SA rekey, the QUEUE_INFO Notify MUST NOT be included. As with Traffic Selector payloads, the QUEUE_INFO may not be different from the Child SA being rekeyed.

This implies a CREATE_CHILD_SA exchange can only have either a QUEUE_INFO or NUM_QUEUES notify. If both Notify types are received, NUM_QUEUES has precedence and QUEUE_INFO MUST be ignored.

The NUM_QUEUES notify, even though it can be sent in IKE_AUTH exchange with TS, is not an attribute of the IKE peer. It is an attribute of the Child SA, similar as how the USE_TRANSPORT notify payload. This allows an IKE peer to have multiple Child SA's covering different traffic selectors and selectively decide whether or not to use multiple Child SA's for those different Child SA's.

4. Implementation specifics

There are various considerations that an implementation can use to determine the best way to install the multiple Child SAs. Below are examples of such strategies.

4.1. One CPU per Child

A simple distribution could be to install one Child SA on each CPU. Note that at least one of the Child SAs must be the "fallback" in case there is no specific Child SA on a specific CPU. This role is

performed by the initial Child SA of the set of identical Child SAs. This ensures that any CPU generating traffic to be encrypted has an available (if not optimal) Child SA to use. Any subsequent Child SA's with identical TSi/TSr are installed in such a way to only be used by a single CPU.

Implementations supporting per-CPU SAs SHOULD extend their mechanism of on-demand negotiation that is triggered by traffic to include a CPU (or queue) identifier in their ACQUIRE message from the SPD to the IKE daemon (eg via NETLINK of PFKEYv2). If the ACQUIRE message does not support sending a per-CPU identifier, then the IKE daemon may initiate all its Child SAs immediately upon receiving an ACQUIRE.

Performing per-CPU Child SA negotiations can result in both peers initiating additional Child SAs at once. This is especially likely in the per-CPU acquire case. Responders should install the additional Child SA on a CPU with the least amount of additional Child SA's for this TSi/TSr pair. It should count outstanding ACQUIRES as an assigned additional Child SA. It is still possible that when the peers only have one slot left to assign, that both peers send an ACQUIRE at the same time. The initiator that receives the CREATE_CHID_SA response last, eg the initiator of the slowest duplicate Child SA, MAY send a delete to delete the duplicate additional Child SA.

As an optimization, additional Child SAs that see little traffic MAY be deleted. The initial Child SA that is not limited to a single CPU MUST NOT be deleted when idle, as it is likely to be idle if enough per-CPU Child SA's are installed. However, if one of those per-CPU child SA's is deleted because it was idle, and subsequently that CPU starts the generate traffic again, that traffic should be encrypted by the initial non-CPU specific Child SA while the IKE daemon processes the ACQUIRE to bring up a new per-CPU Child SA.

When the number of queues or CPUs are different between the peers, the peer with the least amount of queues or CPUs MAY decide to not install a second outbound Child SA as it will never use that Child SA to send traffic. However, it MUST install all inbound Child SA's as it has committed to receiving traffic on these negotiated Child SAs. It MUST NOT generate an error when deleting the (missing) outbound SA component of such a Child SA.

A per-CPU ACQUIRE message SHOULD still send the Traffic Selector (TSi) entry containing the information of the trigger packet. This information MAY be used by the peer to select the most optimal target CPU to install the additional Child SA on. For example, if the trigger packet was for a TCP destination to port 25 (SMTP), it might be able to install the Child SA on the CPU that is also running the

mail server process. Trigger packet Traffic Selectors are documented in [RFC7296] Section 2.9.

The QUEUE_INFO Notify payload MUST be sent in the CREATE_CHILD_SA request for the additional Child SAs. It is used to convey the QoS stream or CPU id. Note that this ID value does not necessarily have to match any physical CPU IDs.

[Clarify narrowing Traffic Selectors. Should it be allowed/forbidden ?]

[Clarify CP / INTERNAL_ADDRESS. Should it be allowed/forbidden ?]

[UDP encaps Due to the nature handling of UDP encapsulated ESP at the receiver NIC queue and intermediate routers for parallel paths, UDP encapsulated ESP may use multiple source ports. We need define a way to select UDP source ports for the Sub SA while IKE SA and the Head remain on UDP port 4500 - 4500. NOTE: libreswan has an experimental implementation for Linux XFRM.]

[Add text about how this parallel SA use may inter operate with 6311? may be not?]

4.2. QoS Child SA's

To install multiple Child SA's for different QoS levels, a method similar to per-CPU is used. The initial Child SA is used for all QoS levels not matched by more specific Child SA's. Additional Child SA's are installed per QoS level, which can be done on-demand if the kernel's IPsec subsystem can send per-QoS level ACQUIRES to the IKE daemon.

A request for a Child SA for a specific QoS value MUST include the QUEUE_INFO Notify payload set to the required QoS value so that both endpoints use the same Child SA for the same QoS level. If a certain QoS level proposed is not acceptable to the responder, TS_UNACCEPTABLE MUST be returned. During Child SA REKEY, the QUEUE_INFO Notify MUST NOT be included and MUST be ignored when received.

5. Payload Format

All multi-octet fields representing integers are laid out in big endian order (also known as "most significant byte first", or "network byte order").

5.1. NUM_QUEUES Notify Payload

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
! Next Payload										!C! RESERVED										Payload Length										!	
! Protocol ID										! SPI Size										Notify Message Type										!	
! Minimum number of IPsec SAs																														!	

- o Protocol ID (1 octet) - MUST be 0. MUST be ignored if not 0.
- o SPI Size (1 octet) - MUST be 0. MUST be ignored if not 0.
- o Notify Message Type (2 octets) - set to [TBD]
- o Minimum number of per-CPU IPsec SAs (4 octets). initiator value Value MUST be greater than 0. If 0 is received, it MUST be interpreted as 1.

Note: The first Child SA that is not bound to a single CPU is not counted as part of these numbers.

5.2. QUEUE_INFO Notify Payload

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
! Next Payload										!C! RESERVED										Payload Length										!	
! Protocol ID										! SPI Size										Notify Message Type										!	
!																														!	
~										Optional payload data																				~	
!																														!	

- o Protocol ID (1 octet) - MUST be 0. MUST be ignored if not 0.
- o SPI Size (1 octet) - MUST be 0. MUST be ignored if not 0.
- o Notify Message Type (2 octets) - set to [TBD]
- o Optional Payload Data. This can be set to identify the QoS value or the CPU ID. The interpretation of the value is left to local

implementations? [Probable needs to be specified by this document]

6. Security Considerations

[TO DO]

7. Implementation Status

[Note to RFC Editor: Please remove this section and the reference to [RFC6982] before publication.]

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. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. 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 [RFC7942], "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".

Authors are requested to add a note to the RFC Editor at the top of this section, advising the Editor to remove the entire section before publication, as well as the reference to [RFC7942].

7.1. Linux XFRM

Organization: Linux kernel XFRM

Name: XFRM-PCPU-v1
<https://git.kernel.org/pub/scm/linux/kernel/git/klasert/linux-stk.git/log/?h=xfrm-pcpu-v1>

Description: An initial Kernel IPsec implementation of the per-CPU method.

Level of maturity: Alpha

Coverage: Implements Initial Child SA and per-CPU additional Child SA's. Also implements per-CPU ACQUIRES using NETLINK. PFKEYv2 is not supported.

Licensing: GPLv2

Implementation experience: TBD

Contact: Linux IPsec: members@linux-ipsec.org

7.2. Libreswan

Organization: The Libreswan Project

Name: pcpu-3 https://libreswan.org/wiki/XFRM_pCPU

Description: An initial IKE implementation of the per-CPU method.

Level of maturity: Alpha

Coverage: implements Initial Child SA and per-CPU additional Child SA's

Licensing: GPLv2

Implementation experience: TBD

Contact: Libreswan Development: swan-dev@libreswan.org

7.3. strongSWAN

Organization: Secunet

Name: StrongSWAN <https://github.com/antonyantony/strongswan/>

Description: An initial IKE implementation of the per-CPU method.

Level of maturity: Alpha

Coverage: implements Initial Child SA and per-CPU additional Child SA's

Licensing: GPLv2

Implementation experience: the Linux XFRM implemenation needs an additional flag on the SPD entry, XFRM_POLICY_CPU_ACQUIRE. It should be set only on the "outgoing" policy. The flag should be disabled when the policy is a trap policy without SPD state.

After a successful negotiation of NUM_QUEUES, the SPD policy is updated to enable the XFRM_POLICY_CPU_ACQUIRE flag. For the outgoing additional Child SAs, the u32 XFRMA_SA_PCPU attribute is set, starting from 0. The incoming SA do not need XFRMA_SA_PCPU. The kernel internally set the value 0xFFFFFFFF. The strongswan implementation uses private space values for NUM_QUEUES (40970) and QUEUE_INFO (40971). The iproute2 software that supports these two attributes is available at <https://github.com/antonyantony/iproute2/tree/pcpu-v1>

Contact: Antony Antony: antony.antony@secunet.com.

8. IANA Considerations

This document defines two new IKEv2 Notify messages for the IANA "IKEv2 Notify Message Types - Status Types" registry.

Value	Notify Messages - Status Types	Reference
[TBD]	NUM_QUEUES	[this document]
[TBD]	QUEUE_INFO	[this document]

Figure 1

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>>.
- [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

- [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>>.

[RFC6982] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", RFC 6982, DOI 10.17487/RFC6982, July 2013, <<https://www.rfc-editor.org/info/rfc6982>>.

[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>>.

Authors' Addresses

Antony Antony
secunet Security Networks AG

Email: antony.antony@secunet.com

Steffen Klassert
secunet Security Networks AG

Email: steffen.klassert@secunet.com

Paul Wouters
Red Hat

Email: pwouters@redhat.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: October 20, 2022

V. Smyslov
ELVIS-PLUS
April 18, 2022

Revised Cookie Processing in the IKEv2 Protocol
draft-smyslov-ipsecme-ikev2-cookie-revised-03

Abstract

This document defines a revised processing of cookies in the Internet Key Exchange protocol Version 2 (IKEv2). It is intended to solve a problem in IKEv2 when due to packets loss and reordering peers may erroneously fail to authenticate each other when cookies are used in the initial IKEv2 exchange.

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 October 20, 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terminology and Notation	2
3. Using Cookies in IKEv2	3
4. Problem Description	3
5. Revised Cookie Processing	6
5.1. Negotiation of Revised Cookie Processing	6
5.2. Processing of REVISED_COOKIE Notification	7
5.3. Changes in AUTH Payload Calculation	7
6. Security Considerations	8
7. IANA Considerations	9
8. Acknowledgements	9
9. Normative References	9
Author's Address	10

1. Introduction

The Internet Key Exchange protocol Version 2 (IKEv2) described in [RFC7296] includes mechanism to defend against DoS attacks. The mechanism is based on cookie which a responder can request an initiator to return in a subsequent request. This allows the responder avoid creating state until it is sure that the initiator's IP address is not spoofed. The cookie mechanism is optional and it is up to the responder whether to use it or not.

When cookie mechanism is used in networks with high probability of packets loss and reordering, it is possible that peers end up with different views on whether cookies were used or not or which content the used cookie had. Since cookie, if used, is a part of an IKEv2 message that is included into calculation of authentication data by both peers, the different views leads to the situation when peers erroneously fail to authenticate each other.

This specification revises processing of cookies in IKEv2 in such a way that peers supporting it exclude cookies from data to be authenticated. This allows them to complete authentication even in the situation described above.

2. Terminology and Notation

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.

3. Using Cookies in IKEv2

Section 2.6 of [RFC7296] specifies that when a responder detects a large number of half-open IKE SAs, it SHOULD reply to an IKE_SA_INIT request with a response containing the COOKIE notification and If an IKE_SA_INIT response includes the COOKIE notification, the initiator MUST then retry the IKE_SA_INIT request, and include the COOKIE notification containing the received data as the very first payload in it, retaining all other payloads intact. This process is illustrated in Figure 1.

Initiator		Responder
-----		-----
send req1:		
HDR, SAi1, KEi, Ni	-->	recv req1, send resp1:
	<--	HDR, N(COOKIE,c)
recv resp1, send req2:		
HDR, N(COOKIE,c),		
SAi1, KEi, Ni	-->	recv req2, send resp2:
	<--	HDR, SAr1, KEr, Nr
recv resp2		

Figure 1

Note, that the responder saves no state when it sends message resp1. This is achieved due to the way cookies are generated. A good way to generate a cookie is described in Section 2.6 of [RFC7296]:

Cookie = <VersionIDofSecret> | Hash(Ni | IPi | SPIi | <secret>)

where <secret> is a randomly generated secret known only to the responder which is periodically changed. [RFC7296] advises the responder to change the value of <secret> frequently, especially if under attack.

Later in the IKE_AUTH exchange the IKE_SA_INIT messages are authenticated by including their content intact into the data that is signed (or MAC'ed) using peers' credentials (see Section 2.15 of [RFC7296] for details).

4. Problem Description

To successfully complete authentication it is important that both peers use the same content of the IKE_SA_INIT messages when calculating authentication data. However, when cookies are employed, the IKE_SA_INIT request is sent at least twice with different content. Section 2.15. of [RFC7296] states that if the first message of the IKE_SA_INIT exchange is sent multiple times with different

content (e.g. with a cookie), it is the latest version of the message that is used for authentication. However, in situations when network packets can be lost and reordered peers may end up with different views on what is "the latest version of the message". Two examples of such situations are shown below.

Consider a situation when at the time the initiator starts creating IKE SA by sending req1 message the responder thinks it's under attack and responds with a resp2 message containing cookie request. However, this message is delayed in the network.

Initiator		Responder
-----		-----
send req1:		
HDR, SAi1, KEi, Ni	-->	recv req1, send resp1:
(delayed)	<--	HDR, N(COOKIE,c)

Since the initiator hasn't received any response, it retransmits its initial request message req1 after some time. During this time the situation on the responder has changed and it doesn't think it's under attack anymore, so it responds with resp2 message and considers the IKE_SA_INIT exchange completed. This message is also delayed in the network.

Initiator		Responder
-----		-----
re-send req1:		
HDR, SAi1, KEi, Ni	-->	recv req1, send resp2:
(delayed)	<--	HDR, SAR1, KEr, Nr

After some time the initiator eventually receives the first initiator's response resp1, which contains cookie request. It is the first response the initiator receives from the responder, so it re-sends the request adding the received cookie into it (req2). However, this message is lost and never reaches the responder. Shortly after sending req2 the initiator receives resp2 message.

Initiator		Responder
-----		-----
recv resp1, send req2:		
HDR, N(COOKIE,c),		
SAi1, KEi, Ni	-->	(lost)
recv resp2		

At this point both peers have completed the IKE_SA_INIT exchange and the KE_AUTH exchange is ready to start. However, the peers have different opinions on what the latest IKE_SA_INIT request message was - the initiator thinks it was req2, while the responder thinks it was

req1. As a result - the authentication in the IKE_AUTH exchange will fail.

Let's consider another possible sequence, that leads to the same result. As with the previous example the initiator starts creating IKE SA by sending req1 message. The responder thinks it's under attack and responds with a resp2 message containing cookie request with cookie c1. However, this message is delayed in the network.

Initiator		Responder
-----		-----
send req1:		
HDR, SAi1, KEi, Ni	-->	recv req1, send resp1:
(delayed)	<--	HDR, N(COOKIE, c1)

As with the first example, the initiator hasn't received any response and retransmits its initial request message req1 after some time. It happens that within this time the value of <secret> has changed (note, that [RFC7296] advises to do it frequently, especially when under attack). Since the responder cannot verify cookie c1 and it still thinks it is under attack, it acts as if req1 contains no cookie and sends back resp2 message also containing cookie request, with a new cookie c2 (that was calculated using the same input data and the new value of <secret>).

Initiator		Responder
-----		-----
re-send req1:		
HDR, SAi1, KEi, Ni	-->	recv req1, send resp2:
	<--	HDR, N(COOKIE, c2)

This message is not delayed, it reaches the initiator and the initiator sends a new request req2 containing cookie c2. The responder receives this message and successfully verifies the cookie using its current value of <secret>. Since the cookie verification is successful, the responder sends back resp3 message and considers the IKE_SA_INIT exchange completed. However, resp3 message is delayed.

Initiator		Responder
-----		-----
recv resp2, send req2:		
HDR, N(COOKIE, c2),		
SAi1, KEi, Ni	-->	recv req2, send resp3:
(delayed)	<--	HDR, SAr1, KEr, Nr

After some time the initiator eventually receives resp1 message, which was delayed. This message contains another value of cookie -

c1. Since this message was received later than resp2 message, the initiator thinks the value c1 is fresher than c2 and sends a new request message req3 now with c1 cookie. This message is lost in the network and never reaches the responder. Shortly after sending req3 the initiator receives resp3 message.

Initiator		Responder
-----		-----
recv resp1, send req3:		
HDR, N(COOKIE, c1),		
SAil, KEi, Ni	-->	(lost)
recv resp3		

As with the first example, at this point both peers have completed the IKE_SA_INIT exchange and are ready for the KE_AUTH exchange. However, the peers again have different opinions on what the latest IKE_SA_INIT request message was - the initiator thinks it was req3, while the responder thinks it was req2. As a result - the authentication in the IKE_AUTH exchange will fail as with the previous example.

The root of this problem is that the initial request can be re-sent several times with different content depending on the responder's current state, which can change over time. Note, that this situation is generally not possible with the INVALID_KEY_PAYLOAD notification, even that in this case the request is also sent several times. This is because the responder will always either require changing Key Exchange method or not, so it is not possible that eventually peers end up with different opinions on what Key Exchange method was negotiated in the IKE_SA_INIT exchange.

5. Revised Cookie Processing

This specification proposes to solve the problem by excluding cookie content from data to be authenticated. The rationale for this is that cookie must be verified by the responder independently at the time it is received in the IKE_SA_INIT request, so there is no need to authenticate it.

5.1. Negotiation of Revised Cookie Processing

For the purpose of using revised cookie processing a new Status Type notify REVISED_COOKIE is defined. Its Notify Message Type is <TBA by IANA>, Protocol ID and SPI Size are both set to 0. The responder includes an empty REVISED_COOKIE notification whenever it sends a response containing COOKIE notification. If the initiator doesn't support this extension it will ignore this notification and continues as described in [RFC7296]. In case the initiator supports revised

cookie processing it will re-send its initial request including the received cookie, but placing the cookie data into the REVISED_COOKIE notification instead of COOKIE notification.

Initiator -----		Responder -----
send req1:		
HDR, SAi1, KEi, Ni	-->	recv req1, send resp1:
		HDR, N(COOKIE,c),
	<--	N(REVISED_COOKIE)
recv resp1, send req2:		
HDR, N(REVISED_COOKIE,c),		
SAi1, KEi, Ni	-->	recv req2, send resp2:
	<--	HDR, SAr1, KEr, Nr
recv resp2		

Figure 2

5.2. Processing of REVISED_COOKIE Notification

If the responder has sent the REVISED_COOKIE notification in the message requesting cookie it should be prepared to receive the re-sent IKE_SA_INIT request with the REVISED_COOKIE notification containing the cookie and with no COOKIE notification. Processing of the REVISED_COOKIE notification by the responder MUST be identical to the processing of COOKIE notification which is described in Sections 2.6 and 2.7 of [RFC7296].

5.3. Changes in AUTH Payload Calculation

In the subsequent IKE_AUTH exchange peers authenticate each other by signing (or MAC'ing) blobs of data. These blobs are defined in Section 2.15 of [RFC7296]. In particular, initiator's blob is defined as follows:

```

InitiatorSignedOctets = RealMessage1 | NonceRData | MACedIDForI
GenIKEHDR = [ four octets 0 if using port 4500 ] | RealIKEHDR
RealIKEHDR = SPIi | SPIr | . . . | Length
RealMessage1 = RealIKEHDR | RestOfMessage1
NonceRPayload = PayloadHeader | NonceRData
InitiatorIDPayload = PayloadHeader | RestOfInitIDPayload
RestOfInitIDPayload = IDType | RESERVED | InitIDData
MACedIDForI = prf(SK_pi, RestOfInitIDPayload)

```

Figure 3

In Figure 3 RealMessage1 is the latest version of the IKE_SA_INIT request message.

If the very first payload in `RealMessage1` is the `REVISED_COOKIE` notify, then `InitiatorSignedOctets` are computed as shown in Figure 4. In particular:

1. The content of the `REVISED_COOKIE` notify payload is eliminated from the message;
2. The Next Payload field in the IKE Header is set to the value of the Next Payload field in the header of eliminated payload;
3. The length of the eliminated payload (indicated in the Length field in its header) is subtracted from the Length field in the IKE Header.

```

InitiatorSignedOctets = PseudoMessage1 | NonceRData | MACedIDForI
RealMessage1 = RealIKEHDR | NotifyREVISED_COOKIE | RestOfMessage1
NotifyREVISED_COOKIE = NextPld | 0 | PldLength | RestOfNotifyCOOKIE
GenIKEHDR = [ four octets 0 if using port 4500 ] | RealIKEHDR
RealIKEHDR = SPIi | SPIr | HdrNextPld | . . . | MsgLength
PseudoKEHDR = SPIi | SPIr | NewHdrNextPld | . . . | NewMsgLength
NewHdrNextPld = NextPld
NewMsgLength = MsgLength - PldLength
PseudoMessage1 = PseudoIKEHDR | RestOfMessage1
NonceRPayload = PayloadHeader | NonceRData
InitiatorIDPayload = PayloadHeader | RestOfInitIDPayload
RestOfInitIDPayload = IDType | RESERVED | InitIDData
MACedIDForI = prf(SK_pi, RestOfInitIDPayload)

```

Figure 4

In brief, if `RealMessage1` doesn't contain the `REVISED_COOKIE` notification then it is used in the authentication as is (Figure 3). Otherwise a new pseudo message `PseudoMessage1` is used which is constructed from `RealMessage1` as if it doesn't contain the `REVISED_COOKIE` notification (Figure 4).

This modification excludes Notify payload containing cookie from the input to the AUTH payload calculation, thus solving the problem described in Section 4.

6. Security Considerations

This extension modifies the way IKE initiator is authenticated to the IKE responder. In particular, the cookie, created by the responder and returned by the initiator in the `IKE_SA_INIT` request is excluded from the data to be authenticated. IKEv2 specification requires that cookie (if present in the request) be verified by the responder at the early stage of the `IKE_SA_INIT` request message processing. If

this verification fails, then the responder must act as if no cookie were present (see Section 2.6 of [RFC7296]), which in most cases results in requesting a new cookie. An adversary that is able to modify cookie content (or remove it from the request) will get no new advantages if this extension is used: either the responder requests a new cookie, or it doesn't care about the cookie at the moment and the IKE_SA_INIT exchange succeeded with invalid cookie. In the later case if revised cookie processing is used the subsequent IKE_AUTH exchange will also succeed and IKE SA will be created, which is different from the current situation, when authentication will fail in the IKE_AUTH if cookie is changed by the attacker.

Excluding cookie from the data to be authenticated doesn't degrade security properties of IKEv2, because the content of the cookie is generated by the responder and must be verified by the responder well before the authentication takes place. The initiator doesn't participate in generation of cookie, it only returns it back as a blob.

Compared to the current processing of cookie the difference caused by the revised processing in a situation when an attacker changes cookie in the IKE_SA_INIT request is that IKE SA will still be created (provided no other obstacles exists), but only if the responder at the moment doesn't care about validity of the received cookie (it means that it is not under attack).

7. IANA Considerations

This document defines a new Notify Message Types in the "Notify Message Types - Status Types" registry:

<TBA> REVISED_COOKIE

8. Acknowledgements

Author is grateful to Tero Kivinen and Wei Pan for sharing their thoughts about this problem and potential ways to solve it.

9. 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>>.
- [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>>.

[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>>.

Author's Address

Valery Smyslov
ELVIS-PLUS
PO Box 81
Moscow (Zelenograd) 124460
RU

Phone: +7 495 276 0211
Email: svan@elvis.ru

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 1, 2022

CJ. Tjhai
Post-Quantum
T. Heider
genua GmbH
V. Smyslov
ELVIS-PLUS
January 28, 2022

Beyond 64KB Limit of IKEv2 Payloads
draft-tjhai-ikev2-beyond-64k-limit-02

Abstract

The maximum Internet Key Exchange Version 2 (IKEv2) payload size is limited to 64KB. This makes IKEv2 not usable for conservative post-quantum cryptosystem whose public-key is larger than 64KB. This document discusses the considerations and defines a mechanism to exchange large post-quantum public keys and signatures in IKEv2.

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 August 1, 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Terminology	4
2. Proposed Solution Overview	4
3. Protocol Details	6
4. Operational Considerations	8
5. Denial of Service Considerations	9
6. Security Considerations	9
7. IANA Considerations	10
8. References	10
8.1. Normative References	10
8.2. Informative References	11
Appendix A. Alternative Approaches	11
A.1. Hash and URL	11
A.1.1. Key Exchange Payload	12
A.1.2. Certificate Payload	13
A.2. Incremental Transfer and Confirmation	13
Authors' Addresses	14

1. Introduction

Digital communications are secured by public-key cryptography algorithms that rely on computational hardness assumptions such as the difficulty in factoring large integers or that of finding the discrete logarithm on an elliptic curve group or finite-field. Recent advances in quantum computing, however, have caused some concerns on the security of these assumptions. It is conjectured that these hard computational problems can be solved in polynomial time when sufficiently large quantum computers become available. The concerns have prompted the National Institute of Standards and Technology (NIST) to initiate a process to standardize one or more public-key algorithms that are quantum-resistant. This family of algorithms is known as post-quantum or quantum-resistant cryptographic algorithms.

It would be ideal if these cryptographic algorithms can be drop-in replacements to the classical algorithms we currently use. Unfortunately, almost all of the post-quantum cryptography algorithms have either public-key, ciphertext or signature size that is many times larger than their classical counterparts. One of the issues that this will cause, in particular for UDP-based protocols such as IPsec, is fragmentation of packets at IP layer. In the context of IPsec/IKEv2 post-quantum key exchange, the fragmentation issue can be

addressed by sending the post-quantum exchange data in IKE_INTERMEDIATE [I-D.ietf-ipsecme-ikev2-intermediate], which is the intermediary state between IKE_SA_INIT and IKE_AUTH. This is the approach taken in [I-D.ietf-ipsecme-ikev2-multiple-ke] whereby a classical and one or more post-quantum key exchanges are combined in order to establish security associations that are quantum-resistant.

Because all public-key cryptography algorithms rely on computational hardness assumptions, the confidence of a cryptographic algorithm is an important consideration. An algorithm that has been well-studied and field-tested is generally better trusted than newer ones. Unfortunately, the confidence of post-quantum cryptographic algorithms is relatively low. All of the algorithms submitted to NIST post-quantum standardization are based on new computational hardness assumptions and despite being conjectured to be resistant to quantum computer attacks, they have not been well cryptanalyzed compared to the classical counterparts. An exception to this is the Goppa-code based McEliece cryptosystem [McEliece] which has withstood years of cryptanalysis since 1978 and still remains unbroken. It is not surprising that a more efficient and CCA secure version of McEliece cryptosystem, Classic McEliece [CM], is selected as one of the finalists in NIST post-quantum cryptography standardization (at the time of writing this document) [NIST]. Furthermore, this cryptosystem has also been recommended for long-term confidentiality protection of data, see [BSI].

While there is interest in using McEliece cryptosystem, in particular for information that needs to remain secure for a long time, there is a challenge in integrating it with IKEv2 [RFC7296]. One characteristic of McElieces cryptosystem is the very asymmetric size of its ciphertext and public-key. While its ciphertext is the smallest compared to all other post-quantum key-establishment algorithms submitted to NIST, the size of its public-key, however, is the largest. The smallest public-key size of Classic McEliece is 255KB. This presents a problem if one were to use Classic McEliece for key-establishment with IKEv2, as the maximum payload size supported by IKEv2 is limited to 64KB. This document describes a mechanism to support IKEv2 key-exchange with key size larger than 64KB, building on the works in [I-D.ietf-ipsecme-ikev2-multiple-ke] and [I-D.ietf-ipsecme-ikev2-intermediate].

In addition, some post-quantum digital signature algorithms that are finalists or alternate candidates of NIST post-quantum cryptography standardization (at the time of writing this document) [NIST], also have either public key size or signature size greater than 64 KB. This makes impossible to use them in IKEv2 as drop-in replacement for classic signature algorithms.

This document is focused on providing a solution for using large post-quantum algorithms related data (public keys and signatures) in IKEv2. It is not a goal of this document to provide a generic solution to transport large data blocks of arbitrary type in IKEv2.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] and [RFC8174].

This document assumes familiarity with IKEv2 concept described in [RFC7296].

2. Proposed Solution Overview

While the Length field in IKEv2 header has a size of 32 bits, so that the maximum size of an IKEv2 message can theoretically reach 4 GB, the size of any individual payload inside a message is limited to 64 KB due to the fact that the Payload Length field in generic payload header consumes 16 bits only. This makes impossible to transfer blocks of data greater than 64 KB, such as public keys of some post-quantum key exchange methods or some post-quantum signatures. In IKEv2 three types of payloads may contain large amounts of data related to post-quantum algorithms:

- o Key Exchange (KE) payload in case of large public key of a post-quantum key exchange method
- o Authentication (AUTH) payload in case of large signature of a post-quantum signature algorithm
- o Certificate (CERT) payloads in case of large public key of a post-quantum signature algorithm

This specification proposes the following solution to the problem: when block of data of a particular type (public key, signature) exceeds 64 KB in size, it is split into a series of chunks smaller than 64 KB. Each chunk then is placed in its own payload, so that the large block of data is eventually transferred in a series of adjacent payloads of the same type. All these payloads MUST have the same values in their headers (except for Next Payload and Payload Length fields) and MUST be transferred adjacent to each other, so that no other payload should appear between them.

This approach works well for KE and AUTH payloads, since only one such large block is transferred in a message and there is no

ambiguity when it is split over multiple payloads. However, when multiple certificates containing large public keys are transferred and each of them is further splitted into several CERT payloads, there must be a way to find boundaries between these certificates on a receiving side. To solve this problem an empty CERT payload MUST be inserted between other non-empty CERT payloads to mark boundaries between individual certificates. Note that large certificates can also be transferred using "Hash and URL" format (see Section 3.6 of [RFC7296]).

The resulting message would exceed 64 KB in size, so that it would not fit into a single UDP datagram. Even if TCP transport [I-D.ietf-ipsecme-rfc8229bis] is used, the size of any individual IKE message in a TCP stream is still limited to 64 KB. For this reason, IKE Fragmentation [RFC7383] MUST be used regardless of the transport protocol if peers are going to transfer large blocks of data. In the case of TCP, the size of fragments is not related to path MTU and can reach 64 KB.

Since IKE Fragmentation is mandatory with this extension and it only can be used on encrypted IKE messages, large blocks of data cannot be transferred in the IKE_SA_INIT exchange.

While mandatory IKE Fragmentation makes it possible to transfer large blocks of data using UDP transport, in practice it may be problematic for the following reason. When fragmenting large messages the number of fragments would be high and all of them are sent at once. If any of these fragment were lost, all the fragments should be re-sent. In congested network environments this would have a negative effect, worsening the congestion. Moreover, the number of IKE message fragments is limited to 2^{16} . With typical size of IKE message fragment equal to PMTU or less, this would limit the size of a single large block of data to ~30-100 MB. While this is enough for current applications of this specification, it may be a limitation in the future.

TCP transport has built-in acknowledgement and congestion control mechanisms and does not suffer from these problems. In addition, since the size of IKE message fragments in case of TCP may be up to 64 KB, the size of a single large block of data can in theory reach 4 GB. However, [I-D.ietf-ipsecme-rfc8229bis] implies that if TCP is used as transport for IKE, it is also used for ESP. Encapsulation ESP in TCP has a lot of negative effects on performance and on ESP functionality (see Section 10 of [I-D.ietf-ipsecme-rfc8229bis]).

This specifications proposes a mixed transport mode as a solution to the problem. In this mode, IKE uses TCP as its transport, while ESP packets are still sent over IP or are encapsulated in UDP. The use

of mixed transport mode is optional and is negotiated in the IKE_SA_INIT exchange.

3. Protocol Details

The initiator starts creating an IKEv2 SA by sending the IKE_SA_INIT request message. If the initiator is going to transfer large blocks of data (e.g. large public keys), then it should make some preparations:

- o IKEV2_FRAGMENTATION_SUPPORTED notification MUST be included to negotiate support for IKE Fragmentation
- o INTERMEDIATE_EXCHANGE_SUPPORTED notification MUST be included if the initiator proposes key exchange methods with public keys greater than 64 KB
- o If the initiator is going to use mixed transport mode then it starts the IKE_SA_INIT request using UDP port 4500 and includes a new status type notification IKE_OVER_TCP (<TBA by IANA>), which has protocol 0, SPI size 0 and contains no data; if the initiator starts the IKE_SA_INIT over TCP, then the mixed transport mode cannot be used and this notification SHOULD NOT be included, it MUST be ignored by the responder if it is still included in the message

Note that UDP port 4500 (and not port 500) is used for the IKE_SA_INIT messages, which is allowed by [RFC7296]. Using port 4500 allows return routability check for UDP messages to be carried out and ensures ESP packets can get through if they are UDP encapsulated.

The responder supporting this specification MUST agree on using IKE Fragmentation by sending back IKEV2_FRAGMENTATION_SUPPORTED notification. If it selects proposal with key exchange method having public key greater than 64 KB, then it MUST agree on using the IKE_INTERMEDIATE exchange by sending back INTERMEDIATE_EXCHANGE_SUPPORTED notification.

If the initiator proposed using mixed transport mode by initiating the IKE_SA_INIT exchange over UDP port 4500 and including IKE_OVER_TCP notification and the responder supports this mode and is willing to use it, then it sends this notification back in the IKE_SA_INIT response. In this case the initiator MUST switch to TCP using destination port 4500 in the next exchange (IKE_INTERMEDIATE or IKE_AUTH) and the responder MUST be prepared to receive the next exchange request message on TCP port 4500. Once switched all subsequent IKE exchanges MUST use TCP transport as described in [I-D.ietf-ipsecme-rfc8229bis], but ESP packets MUST NOT be sent using

TCP, instead they are sent either over IP or using UDP encapsulation, depending on the presence of NAT, which is determined in the IKE_SA_INIT exchange. Note, that if NAT is detected and UDP encapsulation of ESP is used, then NAT keepalive messages MUST be sent by the peer that is behind NAT over UDP using ports from the IKE_SA_INIT exchange, as defined in [RFC3948].

If the responder does not support mixed transport mode, then it ignores the IKE_OVER_TCP notification and all subsequent IKE exchanges will use UDP transport. Note, that in case the initiator started the IKE_SA_INIT over TCP then the IKE_OVER_TCP notification would not be included in the request message and there would be no option for mixed transport mode.

Initiator	Responder

HDR, SAi1, KEi1, Ni, N(NAT_DETECTION_SOURCE_IP), N(NAT_DETECTION_DESTINATION_IP), N(IKEV2_FRAGMENTATION_SUPPORTED), [N(INTERMEDIATE_EXCHANGE_SUPPORTED),] [N(IKE_OVER_TCP)] --->	HDR, SAR1, KER1, Nr, N(NAT_DETECTION_SOURCE_IP), N(NAT_DETECTION_DESTINATION_IP), N(IKEV2_FRAGMENTATION_SUPPORTED), [N(INTERMEDIATE_EXCHANGE_SUPPORTED),] <--- [N(IKE_OVER_TCP)]

Once the IKE_SA_INIT exchange is completed, the peers continue with the following exchanges: one or more IKE_INTERMEDIATE exchanges in case multiple key exchanges are negotiated or the IKE_AUTH exchange, as shown below. Note that all messages containing large blocks of data are sent fragmented using IKE Fragmentation mechanism, but they are not shown here for the sake of simplicity.

Initiator	Responder

HDR, SK{KEi2.1, KEi2.2, KEi2.3, ...} --->	<--- HDR, SK{KEr2.1, KEr2.2, ...}
HDR, SK{KEi3.1, KEi3.2, ...} --->	<--- HDR, SK{KEr3.1, KEr3.2, ...}
...	
HDR, SK{IDi, [IDr,] [CERTi1, CERTi2, ...] [CERTREQ,] [IDr,] AUTHi1, AUTHi2, ... SAi2, TSi, TSr} --->	<--- HDR, SK{IDr, [CERTr1, CERTr2, ...] AUTHr1, AUTHr2, ... SAr2, TSi, TSr}

Since the Payload Length field in the generic IKE payload header has a size of 16 bits, it is impossible to set a proper value for it in the Encrypted Payload header when it contains inner payloads with total length greater than 64 KB. However, using IKE Fragmentation is mandatory when transferring large blocks of data (even in case of TCP transport) and with IKE Fragmentation, the Payload Length field in the Encrypted payload is never transmitted. Instead, the IKE message fragments that appear on the wire are limited to 64 KB in size, so there is no problem with setting a proper value in the Length field of Encrypted Fragment payloads. However, when IKE_INTERMEDIATE exchanges are being authenticated, the content of the Encrypted Payload before encryption and fragmentation is fed to the prf. In this case if the size of the Encrypted payload content exceeds 64 KB then the Payload Length field in the Encrypted Payload header MUST be set to zero when feeding into the prf. On receipt it MUST be checked that the total size of unencrypted payloads the Encrypted Payload contains matches the size of the Encrypted payload calculated from the size of the received message.

4. Operational Considerations

The IKE fragmentation does not require additional infrastructure, however, there is non-zero probability of lost packets when sending a large number of fragments over a UDP connection. Given a set of fragments, when transmitted, each one of them is not individually acknowledged and if any one of them is lost, the entire set will have to be retransmitted. As a consequence, given the size of the payload and also the potential of multiple retransmissions, there may be a noticeable delay in establishing a security association (SA), in particular in lossy network conditions. Therefore, implementations

MAY use a longer timeout value for the purpose of dead-peer detection, but a balance needs to be struck as too large of a value will open up security vulnerabilities as discussed in the following section. In the unlikely event where there is a frequent retransmission due to loss of fragments, implementations MAY send the IKE messages over a TCP connection as specified in [I-D.ietf-ipsecme-rfc8229bis]. If TCP is used as IKE transport, then using mixed transport mode is RECOMMENDED to allow better ESP performance.

5. Denial of Service Considerations

Malicious peers may send a large number of fragments, but incomplete, to the legitimate peer causing memory exhaustion. It is RECOMMENDED that the strategies and recommendations described in [RFC8019] be implemented to counter possible DoS attacks.

An alternative arrangement, if peers do not support [RFC8019], is to allow the transfer of large block of data only after peers are authenticated. In other words, key-establishment using large public-key should not be done to establish an IKE SA, but it should only be used to establish a Child SA or rekeying an IKE SA. In order to protect IKE messages from quantum threats, multiple key-exchanges using a combination of classical and post-quantum ciphers, as described in [I-D.ietf-ipsecme-ikev2-multiple-ke] can be used. Nonetheless, this approach has a limitation whereby if a digital signature scheme with large public-key or signature payload is used, it is still susceptible to DoS attacks.

*** More to be populated in the next version ***

6. Security Considerations

If TCP encapsulation is used, refer to the security considerations in [I-D.ietf-ipsecme-rfc8229bis].

Downloading or transferring large amounts of data is an expensive operation, bandwidth and memory wise. Consequently, implementations should consider using a longer rekeying interval or SHOULD consider relaxing forward secrecy requirements but using CCA-secure key-establishment algorithms only. With chosen-ciphertext attack (CCA)-secure schemes, there is no loss in security if the public-key is reused.

7. IANA Considerations

This document defines a new Notify Message Type in the "Notify Message Types - Status Types" registry:

<TBA> IKE_OVER_TCP

8. References

8.1. Normative References

- [I-D.ietf-ipsecme-ikev2-intermediate]
Smyslov, V., "Intermediate Exchange in the IKEv2 Protocol", draft-ietf-ipsecme-ikev2-intermediate-07 (work in progress), August 2021.
- [I-D.ietf-ipsecme-ikev2-multiple-ke]
Tjhai, C., Tomlinson, M., Bartlett, G., Fluhrer, S., Geest, D. V., Garcia-Morchon, O., and V. Smyslov, "Multiple Key Exchanges in IKEv2", draft-ietf-ipsecme-ikev2-multiple-ke-04 (work in progress), September 2021.
- [I-D.ietf-ipsecme-rfc8229bis]
Smyslov, V. and T. Pauly, "TCP Encapsulation of IKE and IPsec Packets", draft-ietf-ipsecme-rfc8229bis-01 (work in progress), October 2021.
- [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>>.
- [RFC3948] Huttunen, A., Swander, B., Volpe, V., DiBurro, L., and M. Stenberg, "UDP Encapsulation of IPsec ESP Packets", RFC 3948, DOI 10.17487/RFC3948, January 2005, <<https://www.rfc-editor.org/info/rfc3948>>.
- [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>>.
- [RFC7383] Smyslov, V., "Internet Key Exchange Protocol Version 2 (IKEv2) Message Fragmentation", RFC 7383, DOI 10.17487/RFC7383, November 2014, <<https://www.rfc-editor.org/info/rfc7383>>.

- [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>>.

8.2. Informative References

- [BSI] Federal Office for Information Security, "Cryptographic Mechanisms: Recommendations and Key Lengths", 2020, <<https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TG02102/BSI-TR-02102-1.pdf>>.
- [CM] Classic McEliece submission team, "Classic McEliece: NIST post-quantum cryptography standardization finalist", 2020, <<https://classic.mceliece.org/>>.
- [FIPS-202] National Institute of Standards and Technology, "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions", 2015, <<https://doi.org/10.6028/NIST.FIPS.202>>.
- [McEliece] McEliece, R., "A Public-key Cryptosystem based on Algebraic Coding Theory", DSN Progress Report 42-44, 1978.
- [NIST] National Institute of Standards and Technology, "Post-Quantum Cryptography Standardization", <<https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization>>.
- [RFC8019] Nir, Y. and V. Smyslov, "Protecting Internet Key Exchange Protocol Version 2 (IKEv2) Implementations from Distributed Denial-of-Service Attacks", RFC 8019, DOI 10.17487/RFC8019, November 2016, <<https://www.rfc-editor.org/info/rfc8019>>.

Appendix A. Alternative Approaches

A.1. Hash and URL

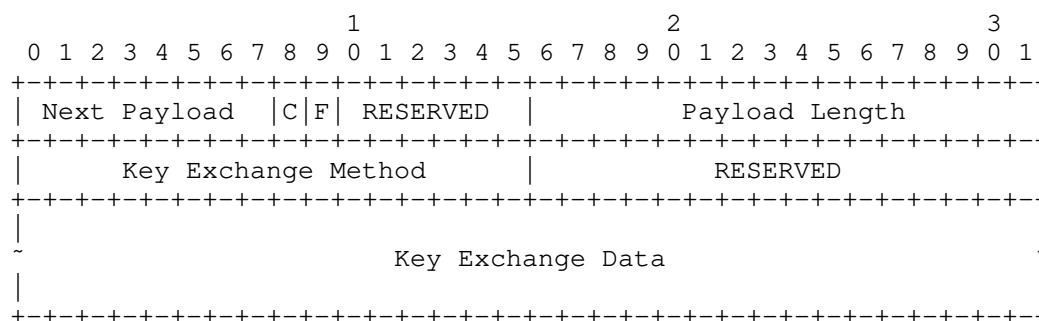
[RFC7296] defines a mechanism whereby an authentication payload such as a certificate can be encoded using a hash value and a URL. A peer utilizes HTTP_CERT_LOOKUP_SUPPORTED Notify payload to indicate that X.509 certificates are not transported in-band, instead the other peer shall fetch the certificates from the given URL and verify its integrity from the hash value. In this way, the peer needs to send 20 octets plus a variable length URL only over the wire, instead of a

few kilobytes of payload, which is useful in the event IKEv2 message fragmentation is not available.

Large public keys can be transported by reusing the same technique and this can be done in two ways, as described below.

A.1.1. Key Exchange Payload

The Key Exchange Data field of IKEv2 Key Exchange Payload contains a single format, which is a blob that is only meaningful to the specified key exchange method. In order to support hash and URL data, an encoding format needs to be specified on the header.



The reserved bit-field F above specifies the encoding format. If it is 0, the Key Exchange Data is a blob as specified in RFC7296. On the other hand if it is 1, the Key Exchange Data is in the form of hash and URL format, whereby the hash value is the SHA3-256 digest [FIPS-202] of the replaced value truncated to 20 octets and the URL value is a variable length URL (in either http or https schema) that resolves to the DER-encoded of the replaced value itself.

Because the hash and URL value is transported in a Key Exchange Payload, it is possible to support the use-case of a single post-quantum key-establishment with large public-key. This payload will be sent as part of IKE_SA_INIT exchange and it will not require IKE_INTERMEDIATE exchanges.

While using hash and URL method to transport large key-establishment data requires minimal modification to IKEv2 protocol, there are disadvantages from deployment point of view that may make this method impractical. Firstly, an IKE peer that originates a hash and URL value will also need to implement additional infrastructure so that it can serve HTTP requests in order to allow the actual key-establishment data to be fetched. While this may not be an issue for Internet facing peers, in the context of road-warrior or remote-access cases, the hash and URL value is initiated by an IKE peer that

is usually a device sitting behind a network address translation (NAT) device and as such, it may not be able to run a publicly reachable HTTP server infrastructure on the same device. An possible solution for these cases is to publish the key-establishment data to a separate server, which is not practical as one cannot expect an IKE initiator to always have deployed an HTTP server. Lastly, IKE peers are predominantly deployed at the network edge where strict firewall rules are generally enforced. The need to open up another port to serve HTTP requests may cause either technical or policy complication that render this approach unacceptable.

The hash and URL approach is vulnerable to (distributed) denial of service attacks as an unauthenticated rogue peer may trick a legitimate peer to fetch a large amount of random meaningless data from a remote server. Implementations SHOULD NOT blindly download all of the data in the given URL. Because a legitimate key-establishment payload should be DER-encoded, they SHOULD download the first few octets to determine the length of the ASN.1 structure representing these octets, then only continue to download the remaining decoded number of octets if the length is expected for the chosen key-establishment algorithm. It should be noted that the content of the data to be downloaded may be under attacker's control and therefore even if the length is as expected, the content may be meaningless bit that is of no use for key-establishment.

A.1.2. Certificate Payload

An alternative is to re-purpose Certificate Payload to carry the hash and URL value of the post-quantum key-establishment data. At the time of writing, the IANA registry defines two hash and URL encoding values, namely X.509 certificate and X.509 certificate bundle. In order to use this payload, a new encoding value for key establishment data will be required.

Because a Certificate Payload is part of IKE_AUTH message, unlike the previous approach, the hash and URL value of the key-establishment data shall be transported via IKE_INTERMEDIATE message. As such, it will not be able to support a single post-quantum key-establishment with a large public-key case. Furthermore, it is semantically incorrect to re-purpose Certificate Payload, which is intended to carry authentication data, to transport key-establishment data.

A.2. Incremental Transfer and Confirmation

As stated in Section 4 of [RFC7383], if any single fragment is lost, the receiving peer will not be able to reassemble the original large key-establishment payload. The above bulk transfer is susceptible to this issue. There is another way to transfer these payload chunks

that is less susceptible to this, but at the cost of higher latency. Instead of transferring in a bulk, each Key Exchange payload chunk must be acknowledged prior to sending the subsequent chunk. As before, the large key-establishment payload is split over several Key Exchange payload chunks where each of them share the same Key Exchange Method value. Each chunk is then sent to the peer using the IKE_INTERMEDIATE message, and each one must be acknowledged by the receiving peer before the subsequent chunk can be sent.

Initiator	Responder

HDR, SAi1, KEi1, Ni, N(IKEV2_FRAGMENTATION_SUPPORTED)*, N(INTERMEDIATE_EXCHANGE_SUPPORTED) --->	
	HDR, SAr1, KEr1, Nr, N(IKEV2_FRAGMENTATION_SUPPORTED)*, <--- N(INTERMEDIATE_EXCHANGE_SUPPORTED)
HDR, SK{KEi2.1, ...} --->	
	<--- HDR, SK{}
HDR, SK{KEi2.2, ...} --->	
	<--- HDR, SK{}
HDR, SK{KEi2.3, ...} --->	
	<--- HDR, SK{KEr2, ...}
HDR, SK{}	---
*: optional	

In order to support key-encapsulation mechanism, the receiving peer has to wait until the entire chunks are received before it can respond with its own Key Exchange payload, which may not be large.

Authors' Addresses

CJ Tjhai
Post-Quantum
UK

Email: cjt@post-quantum.com

Tobias Heider
genua GmbH
DE

Email: me@tobhe.de

Valery Smyslov
ELVIS-PLUS
PO Box 81
Moscow (Zelenograd) 124460
RU

Phone: +7 495 276 0211
Email: svan@elvis.ru