ADD
Internet-Draft
Intended status: Standards Track
Expires: October 9, 2020

M. Boucadair
Orange
T. Reddy
McAfee
D. Wing
Citrix
V. Smyslov
ELVIS-PLUS

April 7, 2020

Internet Key Exchange Protocol Version 2 (IKEv2) Configuration for Encrypted DNS
draft-btw-add-ipsecme-ike-00

Abstract

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

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 9, 2020.

Table of Contents

1. Introduction

This document specifies encrypted DNS configuration for an IKE initiator, particularly the Authentication Domain Name (ADN, defined in [RFC8310]) of DNS-over-HTTPS (DoH) [RFC8484] or DNS-over-TLS (DoT) [RFC7858] server using Internet Key Exchange Protocol Version 2 (IKEv2) [RFC7296].

Particularly, this document introduces a new IKEv2 Configuration Payload Attribute Types (Section 4) for the support of encrypted DNS servers (e.g., DoT, DoH).

Sample use cases are discussed in Section 3. The Configuration Payload Attribute Type defined in Section 4 is not specific to these deployments, but can be used in other deployment contexts.

Note that, for many years, typical designs has often considered that the DNS server was usually located inside the protected domain, but could theoretically be located outside of it. With DoH or DoT, 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 makes use of the terms defined in [RFC8499] and [I-D.ietf-dnsop-terminology-ter].

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

Do53 refers to unencrypted DNS.

'DoH/DoT' refers to DNS-over-HTTPS and/or DNS-over-TLS.

3. Sample Deployment Scenarios

3.1. 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 DoT/DoH instead of Do53 provides privacy and integrity protection along the entire path (rather than just to the VPN termination device) and can communicate the DoT/DoH server policies.

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

For non-split-tunnel VPN configurations, the endpoint uses the Enterprise-provided DoT/DoH 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]).

3.2. 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 DoT/DoH server hosted by the VPN service provider can be securely discovered by the endpoint using the IKEv2 Configuration Payload Attribute Type.

3.3. 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.

VPN service providers can also offer publicly accessible DoH/DoT servers. The split-tunnel VPN configuration allows the client to access the DoH/DoT servers hosted by the VPN provider without traversing the tunnel.

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

4. INTERNAL_ENC_DNS Attribute

The INTERNAL_ENC_DNS IKEv2 Configuration Payload Attribute Type is used to configure an encrypted DNS server. The format of this attribute is shown in Figure 1.

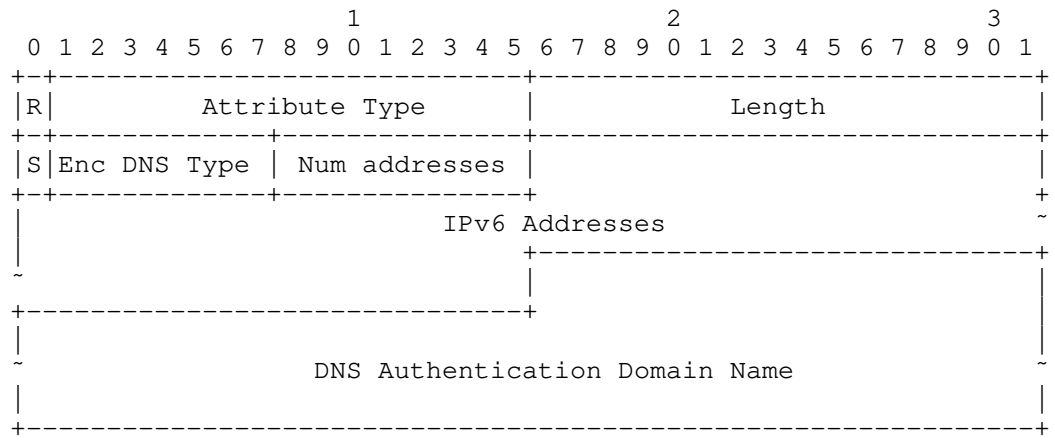


Figure 1: INTERNAL_ENC_DNS Attribute Format

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

- R: Reserved bit; refer to Section 3.15.1 of [RFC7296].

- Attribute Type: MUST be set to TBA (Section 8.1).

- Length: Length of the data in octets. It MUST be set to 1 if the Configuration payload has types CFG_REQUEST or CFG_ACK or to (2 + Length of the ADN + N * 16) if the Configuration payload has types CFG_REPLY or CFG_SET; N being the number of included IP addresses ('Num addresses').

- S: Scope bit. This bit controls whether the DNS queries are sent within the tunnel or outside. If set, this bit instructs the initiator to send encrypted DNS queries outside the tunnel. If the bit is unset, the queries are sent inside the tunnel. The default value of this bit is "0".

- Encrypted DNS Type: Indicates the type of the encrypted DNS server conveyed in this attribute. The following values are defined:

  0: Reserved

  1: DoT

  2: DoH

  See Section 8.2 for future assignment considerations.

- Num addresses: If Length > 1, it indicates the number of enclosed IP addresses.

- IPv6 Address(es): One or more IPv6 addresses to be used to reach the encrypted DNS identified by the name in the DNS Authentication Domain Name.

  IPv4 addresses MUST be encoded using the IPv4-Mapped IPv6 Address format defined in [RFC4291].

- Authentication Domain Name: A fully qualified domain name of the DoT (or DoH) server following the syntax defined in [RFC5890]. The name MUST NOT contain any terminators (e.g., NULL, CR).

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

5. URI Template

DoH servers may support more than one URI Template [RFC8484]. The following sub-sections discuss some candidate solutions for a DoH client to retrieve the list of supported templates by a DoH server. Also, if the resolver hosts several DoH services (e.g., no-filtering, blocking adult content, blocking malware), these services can be discovered as templates.

This section will be updated to reflect the outcome of the discussion in [I-D.btw-add-home].

How a DoH client makes use of the configured DoH services is out of the scope of this document.

6. 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 INTERNAL_ENC_DNS attribute, while responders supply the encrypted DNS configuration in the CFG_REPLY payloads. Concretely:

If the initiator supports encrypted DNS, it includes one or more INTERNAL_ENC_DNS attributes in the CFG_REQUEST with the "Encrypted DNS Type" set to the requested encrypted DNS type (Section 4). For each supported encrypted DNS type the initiator MUST include exactly one INTERNAL_ENC_DNS attribute with the Length field set to 1.

If an INTERNAL_ENC_DNS attribute is included in the CFG_REQUEST, the INTERNAL_ENC_DNS attribute MUST NOT include an ADN and list of IP addresses.

For each INTERNAL_ENC_DNS attribute from the CFG_REQUEST, if the responder supports the corresponding encrypted DNS type, then it MAY send back an INTERNAL_ENC_DNS attribute in the CFG_REPLY with this encrypted DNS type and an appropriate list of IP addresses and ADN. The list of IP addresses MUST NOT be empty.

If the CFG_REQUEST includes an INTERNAL_ENC_DNS attribute but the CFG_REPLY does not include an INTERNAL_ENC_DNS, this is an indication that requested encrypted DNS type(s) is not supported by the responder.

The behavior of the responder if it receives both INTERNAL_ENC_DNS 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 INTERNAL_ENC_DNS attribute in the reply, it should not include any of INTERNAL_IP4_DNS/ INTERNAL_IP6_DNS attributes.

The DNS client establishes a DoH/DoT session with the address(es) conveyed in INTERNAL_ENC_DNS and uses the mechanism discussed in Section 8 of [RFC8310] to authenticate the DNS server certificate using the authentication domain name conveyed in INTERNAL_ENC_DNS.

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 INTERNAL_ENC_DNS 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 INTERNAL_ENC_DNS attribute.

7. 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 DoH/DoT 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.

In most deployment scenarios, the initiator expects that it is using the DoH/DoT server hosted by a specific organization or enterprise. The DNS client can validate the signatory (i.e., cryptographically attested by the organization hosting the DoH/DoT server) using, for example, [I-D.reddy-add-server-policy-selection], and the user can review human-readable privacy policy information of the DNS server and assess whether the DNS server performs DNS-based content filtering. This helps to protect from a compromised IKE server advertising a malicious DoH/DoT server.

The initiator may trust the DoH/DoT 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). In addition, the initiator may prefer IKEv2-supplied DoH/DoT servers if they provide additional features (e.g., malware filtering) compared to the pre-configured DNS servers and meets the privacy preserving data policy requirements of the user.

If the DoH/DoT server that was discovered by means of IKEv2 does not meet the privacy preserving data policy and filtering requirements of the user, the user can instruct the DNS client to take appropriate actions. For example, the action can be to use the local DoH/DoT server only to access internal-only DNS names and use another DNS server (that addresses his/her expectations) for public domains. Such actions and their handling is out of scope.

If IKEv2 is being negotiated with an anonymous or unknown endpoint (such as for Opportunistic Security [RFC7435]), the initiator MUST NOT use INTERNAL_ENC_DNS servers 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].

8. IANA Considerations

8.1. Configuration Payload Attribute Type

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
TBA	INTERNAL_ENC_DNS	YES	1 or more	RFC XXXX

8.2. Encrypted DNS Types

This document requests IANA to create a new registry called "Encrypted DNS Types" under "Internet Key Exchange Version 2 (IKEv2) Parameters" available at https://www.iana.org/assignments/ikev2-parameters/ikev2-parameters.xhtml#ikev2-parameters-21. The initial values of the registry is as follows:

Value	Description	Reference
0	Reserved	RFC XXXX
1	DNS-over-TLS (DoT)	RFC XXXX
2	DNS-over-HTTPs (DoH)	RFC XXXX

New values are assigned on a First Come, First Served (FCFS) basis (Section 4.4 of [RFC8126]).

9. 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.

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <https://www.rfc-editor.org/info/rfc4291>.

[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>.

[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>.

[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>.

[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>.

[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>.

10.2. 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-03 (work in progress), March 2020.

[I-D.btw-add-home]
Boucadair, M., Reddy.K, T., Wing, D., and N. Cook, "DNS-over-HTTPS and DNS-over-TLS Server Discovery and Deployment Considerations for Home and Mobile Networks", draft-btw-add-home-04 (work in progress), March 2020.

[I-D.ietf-dnsop-terminology-ter]
Hoffman, P., "Terminology for DNS Transports and Location", draft-ietf-dnsop-terminology-ter-01 (work in progress), February 2020.

[I-D.reddy-add-server-policy-selection]
Reddy.K, T., Wing, D., Richardson, M., and M. Boucadair, "DNS Server Selection: DNS Server Information with Assertion Token", draft-reddy-add-server-policy-selection-00 (work in progress), March 2020.

[RFC7435] Dukhovni, V., "Opportunistic Security: Some Protection Most of the Time", RFC 7435, DOI 10.17487/RFC7435, December 2014, <https://www.rfc-editor.org/info/rfc7435>.

[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>.

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

ADD
Internet-Draft
Intended status: Standards Track
Expires: August 23, 2021

M. Boucadair
Orange
T. Reddy
McAfee
D. Wing
Citrix
V. Smyslov
ELVIS-PLUS

February 19, 2021

Internet Key Exchange Protocol Version 2 (IKEv2) Configuration for Encrypted DNS
draft-btw-add-ipsecme-ike-02

Abstract

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

Table of Contents

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 DNS-over-HTTPS (DoH) [RFC8484], DNS-over-TLS (DoT) [RFC7858], or DNS-over-QUIC (DoQ) [I-D.ietf-dprive-dnsoquic].

This document introduces new IKEv2 Configuration Payload Attribute Types (Section 4) for the support of DoT, DoH, and DoQ DNS servers. These attributes can be used to provision authentication domain names, a list of IP addresses, alternate port numbers, and a list of DoH URI Template. The use of IKEv2 to provide these template is meant to allow deployments where customized DoH configuration (e.g., per-subscriber or per-site) is required.

Sample use cases are discussed in Section 3. 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 makes use of the terms defined in [RFC8499].

Also, this document makes use 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 as 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 4.

'ENCDNS_IP4_*': refers to an IKEv2 Configuration Payload Attribute Type that carries one or multiple IPv4 addresses of an encrypted DNS server.

'ENCDNS_IP6_*': refers to an IKEv2 Configuration Payload Attribute Type that carries one or multiple IPv6 addresses of an encrypted DNS server.

3. Sample Deployment Scenarios

3.1. 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]).

3.2. 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.

3.3. 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.

4. IKEv2 Configuration Payload Attribute Types for Encrypted DNS

4.1. ENCDNS_IP*_* Configuration Payload Attributes

The ENCDNS_IP*_* IKEv2 Configuration Payload Attribute Types are used to configure a DoT, DoH, or DoQ DNS server. All these attributes share the format shown in Figure 1.

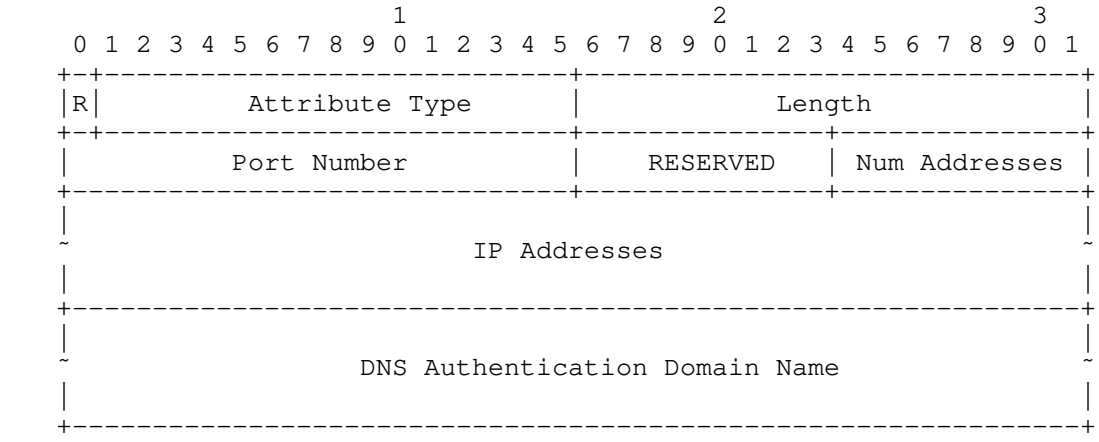


Figure 1: Attributes Format

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

- 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).
- Attribute Type (15 bits) - Identifier for Configuration Attribute Type; is set to one of the TBA1-TBA6 values listed in Section 7.1.
- 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.

  * (2 + Length of the ADN + N * 4) for ENCDNS_IP4_* attributes if the Configuration payload of a has types CFG_REPLY or CFG_SET; N being the number of included IPv4 addresses ('Num addresses').

  * (2 + Length of the ADN + N * 16) 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').

- Port Number (2 octets, unsigned integer) - Indicates the port number to be used for the encrypted DNS. As a reminder, the default port number is 853 for DoT and 443 for DoH.

- RESERVED (1 octet) - These bits are reserved for future use. These bits MUST be set to zero by the sender and MUST be ignored by the receiver.

- Num Addresses (1 octet) - Indicates the number of enclosed IPv4 (for ENCDNS_IP4_* attribute types) or IPv6 (for ENCDNS_IP6_* attribute types) addresses.

- IP Address(es) (variable) - One or more IPv4 or IPv6 addresses to be used to reach the encrypted DNS identified by the name in the DNS Authentication Domain Name.

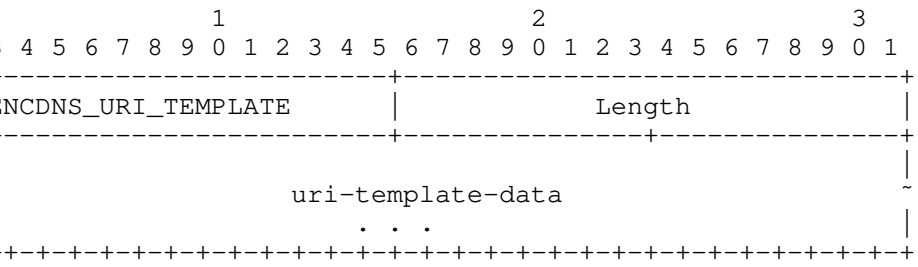
- Authentication Domain Name (variable) - A fully qualified domain name of the DoT, DoH, or DoQ DNS server following the syntax defined in [RFC5890]. The name MUST NOT contain any terminators (e.g., NULL, CR).

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

4.2. ENCDNS_URI_TEMPLATE Configuration Payload Attribute

DoH servers may support more than one URI Template [RFC8484]. Also, if the resolver hosts several DoH services (e.g., no-filtering, blocking adult content, blocking malware), these services can be discovered as templates.

Figure 2 depictes the format of the ENCDNS_URI_TEMPLATE IKEv2 Configuration Payload Attribute Type whihc is used to configure DoH URI Template(s).



Each instance of the uri-template-data is formatted as follows:

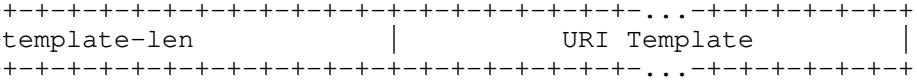


Figure 2: DoH URI Template Attribute Format

The fields of the attribute shown in Figure 2 are as follows:

- 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).
- Attribute Type (15 bits) - Identifier for Configuration Attribute Type; is set to ENCDNS_URI_TEMPLATE (TBA7) (see Section 7.1).
- 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.
- uri-template-data (variable) - XXXX.

  An example of valid URI Template is "XXXX".

How a DoH client makes use of the configured DoH services is out of the scope of this document.

5. 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 multiple 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 more ENCDNS_IP*_* attributes in the CFG_REQUEST with the "Attribute Type" set to the requested encrypted DNS type (Section 4). For each supported encrypted DNS type the initiator MUST include exactly one attribute with the Length field set to 0, so that no data is included for these attributes. If DoH is requested, the initiator includes also ENCDNS_URI_TEMPLATE in the CFG_REQUEST with "Length" set to 0.

For each ENCDNS_IP*_* attribute from the CFG_REQUEST, if the responder supports the corresponding encrypted DNS type, and absent any policy, the responder sends back an ENCDNS_IP*_* attribute in the CFG_REPLY with this encrypted DNS type and an appropriate list of IP addresses, a port number, and an ADN. The list of IP addresses MUST NOT be empty. Multiple instances of the same ENCDNS_IP*_* attribute MAY be returned if distinct ADNs (or port numbers) are to be returned by the responder. If the responder includes ENCDNS_IP4_DOH or ENCDNS_IP6_DOH in the response, it MUST also include ENCDNS_URI_TEMPLATE carrying one or more URI Templates.

If the CFG_REQUEST includes an ENCDNS_IP*_* attribute but the CFG_REPLY does not include an ENCDNS_IP*_* matching the requested encrypted DNS type, this is an indication that requested encrypted DNS type(s) is not supported by the responder or the responder is not configured to provide corresponding server addresses.

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.

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 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.

6. 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].

7. IANA Considerations

7.1. 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_DOT	YES	0 or more	RFC XXXX
TBA2	ENCDNS_IP6_DOT	YES	0 or more	RFC XXXX
TBA3	ENCDNS_IP4_DOH	YES	0 or more	RFC XXXX
TBA4	ENCDNS_IP6_DOH	YES	0 or more	RFC XXXX
TBA5	ENCDNS_IP4_DOQ	YES	0 or more	RFC XXXX
TBA6	ENCDNS_IP6_DOQ	YES	0 or more	RFC XXXX
TBA7	ENCDNS_URI_TEMPLATE	YES	0 or more	RFC XXXX

8. 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 suiggested to return a port number in the attributes.

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>.

[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>.

9.2. 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-dnsoquic]
Huitema, C., Mankin, A., and S. Dickinson, "Specification of DNS over Dedicated QUIC Connections", draft-ietf-dprive-dnsoquic-01 (work in progress), October 2020.

[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>.

[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>.

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: August 26, 2021

D. Fedyk
C. Hopps
LabN Consulting, L.L.C.
February 22, 2021

IP Traffic Flow Security YANG Module
draft-fedyk-ipsecme-yang-iptfs-02

Abstract

This document describes a yang module for the management of IP Traffic Flow Security additions to IKEv2 and IPsec.

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.

Table of Contents

1. Introduction

This document defines a YANG module [RFC7950] for the management of the IP Traffic Flow Security (IP-TFS) extensions as defined in [I-D.ietf-ipsecme-iptfs]. IP-TFS provides enhancements to an IPsec tunnel Security Association to provide improved traffic confidentiality. Traffic confidentiality reduces the ability of traffic analysis to determine identity and correlate observable traffic patterns. IP-TFS offers efficiency when aggregating traffic in fixed size IPsec tunnel packets.

The YANG data model in this document conforms to the Network Management Datastore Architecture defined in [RFC8342].

The only actively published YANG modules for IPsec are found in [I-D.ietf-i2nsf-sdn-ipsec-flow-protection]. This document uses these models as a general IPsec model that can be augmented. The models in [I-D.ietf-i2nsf-sdn-ipsec-flow-protection] provide for an ike and an ikeless model.

1.1. Terminology & Concepts

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

2. Overview

This document defines configuration and operational parameters of IP traffic flow security (IP-TFS). IP-TFS, defined in [I-D.ietf-ipsecme-iptfs], defines a security association for tunnel mode IPsec with characteristics that improve traffic confidentiality and reduce bandwidth efficiency loss. These documents assume familiarity with IP security concepts described in [RFC4301].

IP-TFS uses tunnel mode to improve confidentiality by hiding inner packet identifiable information, packet size and packet timing. IP-TFS provides a general capability allowing aggregation of multiple packets in uniform size outer tunnel ipsec packets. It maintains the outer packet size by utilizing combinations of aggregating, padding and fragmentating inner packets to fll out the IPsec outer tunnel packet. Zero byte padding is used to fill the packet when no data is available to send.

This document specifies an extensible configuration model for IP-TFS. This version utilizes the capabilities of IP-TFS to configure fixed size IP-TFS Packets that are transmitted at a constant rate. This model is structured to allow for different types of operation through future augmentation.

IP-TFS YANG augments IPsec YANG model from [I-D.ietf-i2nsf-sdn-ipsec-flow-protection]. IP-TFS makes use of IPsec tunnel mode and adds a small number configuration items to tunnel mode IPsec. As defined in [I-D.ietf-ipsecme-iptfs], any SA configured to use IP-TFS supports only IP-TFS packets i.e. no mixed IPsec modes.

The behavior for IP-TFS is controlled by the source. The self-describing format of an IP-TFS packets allows a sending side to adjust the packet-size and timing independently from any receiver. Both directions are also independent, e.g. IP-TFS may be run only in one direction. This means that counters, which are created here for both directions may be 0 or not updated in the case of an SA that uses IP-TFS only in on direction.

Cases where IP-TFS statistics are active for one direction:

- SA one direction - IP-TFS enabled

- SA both directions - IP-TFS only enabled in one direction

Case where IP-TFS statistics are for both directions:

- SA both directions - IP-TFS enable for both directions

The data model uses following constructs for configuration and management:

- Configuration

- Operational State

This YANG module supports configuration of fixed size and fixed rate packets, and elements that may be augmented to support future configuration. The protocol specification [I-D.ietf-ipsecme-iptfs], goes beyond this simple fixed mode of operation by defining a general format for any type of scheme. In this document the outer IPsec packets can be sent with fixed or variable size (without padding). The configuration allows the fixed packet size to be determined by the path MTU. The fixed packet size can also be configured if a value lower than the path MTU is desired.

Other configuration items include:

- Congestion Control. A congestion control setting to allow IP-TFS to reduce the packet rate when congestion is detected.

- Fixed Rate configuration. The IP-TFS tunnel rate can be configured taking into account either layer 2 overhead or layer 3 overhead. Layer 3 overhead is the IP data rate and layer 2 overhead is the rate of bits on the link. The combination of packet size and rate determines the nominal maximum bandwidth and the transmission interval when fixed size packets are used.

- User packet Fragmentation Control. While fragmentation is recommended for improved efficiency, a configuration is provided if users wish to observe the effect no-fragmentation on their data flows.

The YANG operational data allows the readout of the configured parameters as well as the per SA statistics and error counters for IP-TFS. Per SA IPsec packet statistics are provided as a feature and per SA IP-TFS specific statistics as another feature. Both sets of statistics augment the IPsec YANG models with counters that allow observation of IP-TFS packet efficiency.

Draft [I-D.ietf-i2nsf-sdn-ipsec-flow-protection] has a mature set of IPsec YANG management objects.

IP-TFS YANG augments:

- Yang catalog entry for ietf-i2nsf-ike@2020-10-30.yang

- Yang catalog entry for ietf-i2nsf-ikeless@20202-10-30.yang

The Security Policy database entry and Security Association entry for an IPsec Tunnel can be augmented with IP-TFS.

3. YANG Management

3.1. YANG Tree

The following is the YANG tree diagram ([RFC8340]) for the IP-TFS extensions.

```
module: ietf-ipsecme-iptfs
  augment /nsfike:ipsec-ike/nsfike:conn-entry/nsfike:spd
            /nsfike:spd-entry/nsfike:ipsec-policy-config
            /nsfike:processing-info/nsfike:ipsec-sa-cfg:
    +--rw traffic-flow-security
       +--rw congestion-control?     boolean
       +--rw packet-size
       |  +--rw use-path-mtu-discovery?   boolean
       |  +--rw outer-packet-size?        uint16
       +--rw (tunnel-rate)?
       |  +--:(l2-fixed-rate)
       |  |  +--rw l2-fixed-rate?    uint64
       |  +--:(l3-fixed-rate)
       |     +--rw l3-fixed-rate?    uint64
       +--rw dont-fragment?          boolean
       +--rw max-aggregation-time?   decimal64
  augment /nsfike:ipsec-ike/nsfike:conn-entry/nsfike:child-sa-info:
    +--ro traffic-flow-security
       +--ro congestion-control?     boolean
       +--ro packet-size
       |  +--ro use-path-mtu-discovery?   boolean
       |  +--ro outer-packet-size?        uint16
       +--ro (tunnel-rate)?
       |  +--:(l2-fixed-rate)
       |  |  +--ro l2-fixed-rate?    uint64
       |  +--:(l3-fixed-rate)
       |     +--ro l3-fixed-rate?    uint64
       +--ro dont-fragment?          boolean
```

```
        +--ro max-aggregation-time?   decimal64
     augment /nsfikels:ipsec-ikeless/nsfikels:spd/nsfikels:spd-entry
               /nsfikels:ipsec-policy-config/nsfikels:processing-info
               /nsfikels:ipsec-sa-cfg:
       +--rw traffic-flow-security
          +--rw congestion-control?     boolean
          +--rw packet-size
          |  +--rw use-path-mtu-discovery?   boolean
          |  +--rw outer-packet-size?        uint16
          +--rw (tunnel-rate)?
          |  +--:(l2-fixed-rate)
          |  |  +--rw l2-fixed-rate?    uint64
          |  +--:(l3-fixed-rate)
          |     +--rw l3-fixed-rate?    uint64
          +--rw dont-fragment?          boolean
          +--rw max-aggregation-time?   decimal64
     augment /nsfikels:ipsec-ikeless/nsfikels:sad/nsfikels:sad-entry:
       +--ro traffic-flow-security
          +--ro congestion-control?     boolean
          +--ro packet-size
          |  +--ro use-path-mtu-discovery?   boolean
          |  +--ro outer-packet-size?        uint16
          +--ro (tunnel-rate)?
          |  +--:(l2-fixed-rate)
          |  |  +--ro l2-fixed-rate?    uint64
          |  +--:(l3-fixed-rate)
          |     +--ro l3-fixed-rate?    uint64
          +--ro dont-fragment?          boolean
          +--ro max-aggregation-time?   decimal64
     augment /nsfike:ipsec-ike/nsfike:conn-entry/nsfike:child-sa-info:
       +--ro ipsec-stats {ipsec-stats}?
       |  +--ro tx-pkts?        uint64
       |  +--ro tx-octets?      uint64
       |  +--ro tx-drop-pkts?   uint64
       |  +--ro rx-pkts?        uint64
       |  +--ro rx-octets?      uint64
       |  +--ro rx-drop-pkts?   uint64
       +--ro iptfs-inner-pkt-stats {iptfs-stats}?
       |  +--ro tx-pkts?              uint64
       |  +--ro tx-octets?            uint64
       |  +--ro rx-pkts?              uint64
       |  +--ro rx-octets?            uint64
       |  +--ro rx-incomplete-pkts?   uint64
       +--ro iptfs-outer-pkt-stats {iptfs-stats}?
          +--ro tx-all-pad-pkts?       uint64
          +--ro tx-all-pad-octets?     uint64
          +--ro tx-extra-pad-pkts?     uint64
          +--ro tx-extra-pad-octets?   uint64
```

```
          +--ro rx-all-pad-pkts?       uint64
          +--ro rx-all-pad-octets?     uint64
          +--ro rx-extra-pad-pkts?     uint64
          +--ro rx-extra-pad-octets?   uint64
          +--ro rx-errored-pkts?       uint64
          +--ro rx-missed-pkts?        uint64
    augment /nsfikels:ipsec-ikeless/nsfikels:sad/nsfikels:sad-entry:
      +--rw ipsec-stats {ipsec-stats}?
      |  +--ro tx-pkts?        uint64
      |  +--ro tx-octets?      uint64
      |  +--ro tx-drop-pkts?   uint64
      |  +--ro rx-pkts?        uint64
      |  +--ro rx-octets?      uint64
      |  +--ro rx-drop-pkts?   uint64
      +--ro iptfs-inner-pkt-stats {iptfs-stats}?
      |  +--ro tx-pkts?              uint64
      |  +--ro tx-octets?            uint64
      |  +--ro rx-pkts?              uint64
      |  +--ro rx-octets?            uint64
      |  +--ro rx-incomplete-pkts?   uint64
      +--ro iptfs-outer-pkt-stats {iptfs-stats}?
         +--ro tx-all-pad-pkts?       uint64
         +--ro tx-all-pad-octets?     uint64
         +--ro tx-extra-pad-pkts?     uint64
         +--ro tx-extra-pad-octets?   uint64
         +--ro rx-all-pad-pkts?       uint64
         +--ro rx-all-pad-octets?     uint64
         +--ro rx-extra-pad-pkts?     uint64
         +--ro rx-extra-pad-octets?   uint64
         +--ro rx-errored-pkts?       uint64
         +--ro rx-missed-pkts?        uint64
```

3.2. YANG Module

The following is the YANG module for managing the IP-TFS extensions.

```
<CODE BEGINS> file "ietf-ipsecme-iptfs@2021-02-22.yang"
module ietf-ipsecme-iptfs {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-ipsecme-iptfs";
  prefix iptfs;

  import ietf-i2nsf-ike {
    prefix nsfike;
  }
  import ietf-i2nsf-ikeless {
    prefix nsfikels;
  }
```

```
  organization
    "IETF IPSECME Working Group (IPSECME)";
  contact
    "WG Web:  <https://tools.ietf.org/wg/ipsecme/>
     WG List: <mailto:ipsecme@ietf.org>

     Author: Don Fedyk
             <mailto:dfedyk@labn.net>

     Author: Christian Hopps
             <mailto:chopps@chopps.org>";

  // RFC Ed.: replace XXXX with actual RFC number and
  // remove this note.

  description
    "This module defines the configuration and operational state for
     managing the IP Traffic Flow Security functionality [RFC XXXX].

     Copyright (c) 2020 IETF Trust and the persons identified as
     authors of the code. All rights reserved.

     Redistribution and use in source and binary forms, with or
     without modification, is permitted pursuant to, and subject to
     the license terms contained in, the Simplified BSD License set
     forth in Section 4.c of the IETF Trust's Legal Provisions
     Relating to IETF Documents
     (https://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC XXXX
     (https://tools.ietf.org/html/rfcXXXX); see the RFC itself for
     full legal notices.";

  revision 2021-02-22 {
    description
      "Initial Revision";
    reference
      "RFC XXXX: IP Traffic Flow Security YANG Module";
  }

  feature ipsec-stats {
    description
      "This feature indicates the device supports
       per SA IPsec statistics";
  }

  feature iptfs-stats {
    description
```

```
      "This feature indicates the device supports
       per SA IP Traffic Flow Security statistics";
  }

  /*--------------------*/
  /*   groupings        */
  /*--------------------*/

  grouping ipsec-tx-stat-grouping {
    description
      "IPsec outbound statistics";
    leaf tx-pkts {
      type uint64;
      config false;
      description
        "Outbound Packet count";
    }
    leaf tx-octets {
      type uint64;
      config false;
      description
        "Outbound Packet bytes";
    }
    leaf tx-drop-pkts {
      type uint64;
      config false;
      description
        "Outbound dropped packets count";
    }
  }

  grouping ipsec-rx-stat-grouping {
    description
      "IPsec inbound statistics";
    leaf rx-pkts {
      type uint64;
      config false;
      description
        "Inbound Packet count";
    }
    leaf rx-octets {
      type uint64;
      config false;
      description
        "Inbound Packet bytes";
    }
    leaf rx-drop-pkts {
      type uint64;
```

```
      config false;
      description
        "Inbound dropped packets count";
    }
  }

  grouping iptfs-inner-tx-stat-grouping {
    description
      "IP-TFS outbound inner packet statistics";
    leaf tx-pkts {
      type uint64;
      config false;
      description
        "Total number of IP-TFS inner packets sent. This
         count is whole packets only.  A fragmented packet
         counts as one packet";
      reference
        "draft-ietf-ipsecme-iptfs";
    }
    leaf tx-octets {
      type uint64;
      config false;
      description
        "Total number of IP-TFS inner octets sent.  This is
         inner packet octets only.  Does not count padding.";
      reference
        "draft-ietf-ipsecme-iptfs";
    }
  }

  grouping iptfs-outer-tx-stat-grouping {
    description
      "IP-TFS outbound inner packet statistics";
    leaf tx-all-pad-pkts {
      type uint64;
      config false;
      description
        "Total number of transmitted IP-TFS packets that
         were all padding with no inner packet data.";
      reference
        "draft-ietf-ipsecme-iptfs section 2.2.3";
    }
    leaf tx-all-pad-octets {
      type uint64;
      config false;
      description
        "Total number transmitted octets of padding added to
         IP-TFS packets with no inner packet data.";
```

```
      reference
        "draft-ietf-ipsecme-iptfs section 2.2.3";
    }
    leaf tx-extra-pad-pkts {
      type uint64;
      config false;
      description
        "Total number of transmitted outer IP-TFS packets
         that included some padding.";
      reference
        "draft-ietf-ipsecme-iptfs section 2.2.3.1";
    }
    leaf tx-extra-pad-octets {
      type uint64;
      config false;
      description
        "Total number of transmitted octets of padding added
         to outer IP-TFS packets with data.";
      reference
        "draft-ietf-ipsecme-iptfs section 2.2.3.1";
    }
  }

  grouping iptfs-inner-rx-stat-grouping {
    description
      "IP-TFS inner packet inbound statistics";
    leaf rx-pkts {
      type uint64;
      config false;
      description
        "Total number of IP-TFS inner packets received.";
      reference
        "draft-ietf-ipsecme-iptfs section 2.2";
    }
    leaf rx-octets {
      type uint64;
      config false;
      description
        "Total number of IP-TFS inner octets received.  Does
         not include padding or overhead";
      reference
        "draft-ietf-ipsecme-iptfs section 2.2";
    }
    leaf rx-incomplete-pkts {
      type uint64;
      config false;
      description
        "Total number of IP-TFS inner packets that were
```

```
          incomplete.  Usually this is due to fragments not
          received. Also, this may be due to misordering or
          errors in received outer packets.";
        reference
          "draft-ietf-ipsecme-iptfs";
      }
    }

    grouping iptfs-outer-rx-stat-grouping {
      description
        "IP-TFS outer packet inbound statistics";
      leaf rx-all-pad-pkts {
        type uint64;
        config false;
        description
          "Total number of received IP-TFS packets that were
           all padding with no inner packet data.";
        reference
          "draft-ietf-ipsecme-iptfs section 2.2.3";
      }
      leaf rx-all-pad-octets {
        type uint64;
        config false;
        description
          "Total number received octets of padding added to
           IP-TFS packets with no inner packet data.";
        reference
          "draft-ietf-ipsecme-iptfs section 2.2.3";
      }
      leaf rx-extra-pad-pkts {
        type uint64;
        config false;
        description
          "Total number of received outer IP-TFS packets that
           included some padding.";
        reference
          "draft-ietf-ipsecme-iptfs section 2.2.3.1";
      }
      leaf rx-extra-pad-octets {
        type uint64;
        config false;
        description
          "Total number of received octets of padding added to
           outer IP-TFS packets with data.";
        reference
          "draft-ietf-ipsecme-iptfs section 2.2.3.1";
      }
      leaf rx-errored-pkts {
```

```
      type uint64;
      config false;
      description
        "Total number of IP-TFS outer packets dropped due to
         errors.";
      reference
        "draft-ietf-ipsecme-iptfs";
    }
    leaf rx-missed-pkts {
      type uint64;
      config false;
      description
        "Total number of IP-TFS outer packets missing
         indicated by missing sequence number.";
      reference
        "draft-ietf-ipsecme-iptfs";
    }
  }

  grouping iptfs-config {
    description
      "This is the grouping for iptfs configuration";
    container traffic-flow-security {
      // config true; want this so we can refine?
      description
        "Configure the IPSec TFS in Security
         Association Database (SAD)";
      leaf congestion-control {
        type boolean;
        default "true";
        description
          "Congestion Control 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.";
        reference
          "draft-ietf-ipsecme-iptfs section 2.5.2";
      }
      container packet-size {
        description
          "Packet size is either auto-discovered or manually
           configured.";
        leaf use-path-mtu-discovery {
          type boolean;
          default "true";
          description
            "Utilize path mtu discovery to determine maximum IP-TFS
```

```
            packet size. If the packet size is explicitly
            configured, then it will only be adjusted downward
            if use-path-mtu-discovery is set.";
        reference
          "draft-ietf-ipsecme-iptfs section 4.2";
      }
      leaf outer-packet-size {
        type uint16;
        description
          "The size of the outer encapsulating tunnel packet (i.e.,
           the IP packet containing the ESP payload).";
        reference
          "draft-ietf-ipsecme-iptfs section 4.2";
      }
    }
    choice tunnel-rate {
      description
        "TFS bit rate may be specified at layer 2 wire
         rate or layer 3 packet rate";
      leaf l2-fixed-rate {
        type uint64;
        description
          "Target bandwidth/bit rate in bps for iptfs tunnel. This
           fixed rate is the nominal timing for the fixed size packet.
           If congestion control is enabled the rate may be adjusted
           down (or up if unset).";
        reference
          "draft-ietf-ipsecme-iptfs section 4.1";
      }
      leaf l3-fixed-rate {
        type uint64;
        description
          "Target bandwidth/bit rate in bps for iptfs tunnel. This
           fixed rate is the nominal timing for the fixed size packet.
           If congestion control is enabled the rate may be adjusted
           down (or up if unset).";
        reference
          "draft-ietf-ipsecme-iptfs section 4.1";
      }
    }
    leaf dont-fragment {
      type boolean;
      default "false";
      description
        "Disable packet fragmentation across consecutive iptfs
         tunnel packets";
      reference
        "draft-ietf-ipsecme-iptfs section 2.2.4 and 6.4.1";
```

```
      }
      leaf max-aggregation-time {
        type decimal64 {
          fraction-digits 6;
        }
        units "milliseconds";
        description
          "Maximum Aggregation Time in  Milliseconds
           or fractional milliseconds down to 1 nanosecond";
      }
    }
  }

  /*
   * IP-TFS ike configuration
   */

  augment "/nsfike:ipsec-ike/nsfike:conn-entry/nsfike:spd/"
        + "nsfike:spd-entry/"
        + "nsfike:ipsec-policy-config/"
        + "nsfike:processing-info/"
        + "nsfike:ipsec-sa-cfg" {
    description
      "IP-TFS configuration for this policy.";
    uses iptfs-config;
  }

  augment "/nsfike:ipsec-ike/nsfike:conn-entry/"
        + "nsfike:child-sa-info" {
    description
      "IP-TFS configured on this SA.";
    uses iptfs-config {
      refine "traffic-flow-security" {
        config false;
      }
    }
  }

  /*
   * IP-TFS ikeless configuration
   */

  augment "/nsfikels:ipsec-ikeless/nsfikels:spd/"
        + "nsfikels:spd-entry/"
        + "nsfikels:ipsec-policy-config/"
        + "nsfikels:processing-info/"
        + "nsfikels:ipsec-sa-cfg" {
    description
```

```
      "IP-TFS configuration for this policy.";
    uses iptfs-config;
  }

  augment "/nsfikels:ipsec-ikeless/nsfikels:sad/"
        + "nsfikels:sad-entry" {
    description
      "IP-TFS configured on this SA.";
    uses iptfs-config {
      refine "traffic-flow-security" {
        config false;
      }
    }
  }

  /*
   * packet counters
   */

  augment "/nsfike:ipsec-ike/nsfike:conn-entry/"
        + "nsfike:child-sa-info" {
    description
      "Per SA Counters";
    container ipsec-stats {
      if-feature "ipsec-stats";
      config false;
      description
        "IPsec per SA packet counters.";
      uses ipsec-tx-stat-grouping {
        //when "direction = 'outbound'";
      }
      uses ipsec-rx-stat-grouping {
        //when "direction = 'inbound'";
      }
    }
    container iptfs-inner-pkt-stats {
      if-feature "iptfs-stats";
      config false;
      description
        "IPTFS per SA inner packet counters.";
      uses iptfs-inner-tx-stat-grouping {
        //when "direction = 'outbound'";
      }
      uses iptfs-inner-rx-stat-grouping {
        //when "direction = 'inbound'";
      }
    }
    container iptfs-outer-pkt-stats {
```

```
      if-feature "iptfs-stats";
      config false;
      description
        "IPTFS per SA outer packets counters.";
      uses iptfs-outer-tx-stat-grouping {
        //when "direction = 'outbound'";
      }
      uses iptfs-outer-rx-stat-grouping {
        //when "direction = 'inbound'";
      }
    }
  }

  /*
   * packet counters
   */

  augment "/nsfikels:ipsec-ikeless/nsfikels:sad/"
        + "nsfikels:sad-entry" {
    description
      "Per SA Counters";
    container ipsec-stats {
      if-feature "ipsec-stats";
      description
        "IPsec per SA packet counters.";
      uses ipsec-tx-stat-grouping {
        //when "direction = 'outbound'";
      }
      uses ipsec-rx-stat-grouping {
        //when "direction = 'inbound'";
      }
    }
    container iptfs-inner-pkt-stats {
      if-feature "iptfs-stats";
      config false;
      description
        "IPTFS per SA inner packet counters.";
      uses iptfs-inner-tx-stat-grouping {
        //when "direction = 'outbound'";
      }
      uses iptfs-inner-rx-stat-grouping {
        //when "direction = 'inbound'";
      }
    }
    container iptfs-outer-pkt-stats {
      if-feature "iptfs-stats";
      config false;
      description
```

```
        "IPTFS per SA outer packets counters.";
      uses iptfs-outer-tx-stat-grouping {
        //when "direction = 'outbound'";
      }
      uses iptfs-outer-rx-stat-grouping {
        //when "direction = 'inbound'";
      }
    }

  }
}
<CODE ENDS>
```

4. IANA Considerations

4.1. Updates to the IETF XML Registry

This document registers a URI in the "IETF XML Registry" [RFC3688]. Following the format in [RFC3688], the following registration has been made:

URI:
urn:ietf:params:xml:ns:yang:ietf-ipsecme-iptfs

Registrant Contact:
The IESG.

XML:
N/A; the requested URI is an XML namespace.

4.2. Updates to the YANG Module Names Registry

This document registers one YANG module in the "YANG Module Names" registry [RFC6020]. Following the format in [RFC6020], the following registration has been made:

name:
ietf-ipsecme-iptfs

namespace:
urn:ietf:params:xml:ns:yang:ietf-ipsecme-iptfs

prefix:
iptfs

reference:
RFC XXXX (RFC Ed.: replace XXXX with actual RFC number and remove this note.)

5. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

The YANG module defined in this document can enable, disable and modify the behavior of IP traffic flow security, for the implications regarding these types of changes consult the [I-D.ietf-ipsecme-iptfs] which defines the functionality.

6. Acknowledgements

The authors would like to thank Eric Kinzie for his feedback on the YANG model.

7. References

7.1. Normative References

[I-D.ietf-i2nsf-sdn-ipsec-flow-protection]
Marin-Lopez, R., Lopez-Millan, G., and F. Pereniguez-Garcia, "Software-Defined Networking (SDN)-based IPsec Flow Protection", draft-ietf-i2nsf-sdn-ipsec-flow-protection-12 (work in progress), October 2020.

[I-D.ietf-ipsecme-iptfs]
Hopps, C., "IP-TFS: IP Traffic Flow Security Using Aggregation and Fragmentation", draft-ietf-ipsecme-iptfs-06 (work in progress), January 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>.

[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>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[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>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

7.2. Informative References

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

Appendix A. Examples

The following examples show configuration and operational data for the ikeless case in xml and ike case in json. Also, the operational statistics for the ikeless case are shown using xml.

A.1. Example XML Configuration

This example illustrates configuration for IP-TFS in the ikeless case. Note that since this augments the ipsec ikeless schema only minimal ikeless configuration to satisfy the schema has been populated.

```
<i:ipsec-ikeless
  xmlns:i="urn:ietf:params:xml:ns:yang:ietf-i2nsf-ikeless"
  xmlns:tfs="urn:ietf:params:xml:ns:yang:ietf-ipsecme-iptfs">
  <i:spd>
    <i:spd-entry>
      <i:name>protect-policy-1</i:name>
      <i:direction>outbound</i:direction>
      <i:ipsec-policy-config>
        <i:traffic-selector>
          <i:local-prefix>1.1.1.1/32</i:local-prefix>
          <i:remote-prefix>2.2.2.2/32</i:remote-prefix>
        </i:traffic-selector>
        <i:processing-info>
          <i:action>protect</i:action>
          <i:ipsec-sa-cfg>
            <tfs:traffic-flow-security>
             <tfs:congestion-control>true</tfs:congestion-control>
              <tfs:packet-size>
                <tfs:use-path-mtu-discovery
                   >true</tfs:use-path-mtu-discovery>
              </tfs:packet-size>
              <tfs:l2-fixed-rate>1000000000</tfs:l2-fixed-rate>
              <tfs:max-aggregation-time
                 >0.1</tfs:max-aggregation-time>
            </tfs:traffic-flow-security>
          </i:ipsec-sa-cfg>
        </i:processing-info>
      </i:ipsec-policy-config>
    </i:spd-entry>
  </i:spd>
</i:ipsec-ikeless>
```

Figure 1: Example IP-TFS XML configuration

A.2. Example XML Operational Data

This example illustrates operational data for IP-TFS in the ikeless case. Note that since this augments the ipsec ikeless schema only minimal ikeless configuration to satisfy the schema has been populated.

```
 <i:ipsec-ikeless
   xmlns:i="urn:ietf:params:xml:ns:yang:ietf-i2nsf-ikeless"
   xmlns:tfs="urn:ietf:params:xml:ns:yang:ietf-ipsecme-iptfs">
   <i:sad>
     <i:sad-entry>
       <i:name>sad-1</i:name>
       <i:ipsec-sa-config>
         <i:spi>1</i:spi>
         <i:traffic-selector>
           <i:local-prefix>1.1.1.1/32</i:local-prefix>
           <i:remote-prefix>2.2.2.2/32</i:remote-prefix>
         </i:traffic-selector>
       </i:ipsec-sa-config>
       <tfs:traffic-flow-security>
        <tfs:congestion-control>true</tfs:congestion-control>
         <tfs:packet-size>
           <tfs:use-path-mtu-discovery>true</tfs:use-path-mtu-discovery>
         </tfs:packet-size>
         <tfs:l2-fixed-rate>1000000000</tfs:l2-fixed-rate>
         <tfs:max-aggregation-time>0.100</tfs::max-aggregation-time>
       </tfs:traffic-flow-security>
     </i:sad-entry>
   </i:sad>
 </i:ipsec-ikeless>
```

Figure 2: Example IP-TFS XML Operational data

A.3. Example JSON Configuration

This example illustrates config data for IP-TFS in the ike case. Note that since this augments the ipsec ike schema only minimal ike configuration to satisfy the schema has been populated.

```
{
  "ietf-i2nsf-ike:ipsec-ike": {
    "ietf-i2nsf-ike:conn-entry": [
      {
        "name": "my-peer-connection",
        "ike-sa-encr-alg": [
          {
            "id": 1,
            "algorithm-type": 12,
            "key-length": 128
          }
          ],
          "local": {
            "local-pad-entry-name": "local-1"
          },
```

```
            "remote": {
              "remote-pad-entry-name": "remote-1"
            },
            "ietf-i2nsf-ike:spd": {
            "spd-entry": [
              {
                "name": "protect-policy-1",
                "ipsec-policy-config": {
                  "traffic-selector": {
                    "local-prefix": "1.1.1.1/32",
                    "remote-prefix": "2.2.2.2/32"
                  },
                  "processing-info": {
                    "action": "protect",
                    "ipsec-sa-cfg": {
                      "ietf-ipsecme-iptfs:traffic-flow-security": {
                        "congestion-control": "true",
                        "l2-fixed-rate": 1000000000,
                        "packet-size": {
                          "use-path-mtu-discovery": "true"
                        },
                        "max-aggregation-time": "0.1"
                      }
                    }
                  }
                }
              }
            ]
          }
        }
      ]
    }
  }
```

Figure 3: Example IP-TFS JSON configuration

A.4. Example JSON Operational Data

This example illustrates operational data for IP-TFS in the ike case. Note that since this augments the ipsec ike tree only minimal ike configuration to satisfy the schema has been populated.

```
{
  "ietf-i2nsf-ike:ipsec-ike": {
    "ietf-i2nsf-ike:conn-entry": [
      {
        "name": "my-peer-connection",
        "ike-sa-encr-alg": [
        {
          "id": 1,
          "algorithm-type": 12,
          "key-length": 128
        }
        ],
        "local": {
          "local-pad-entry-name": "local-1"
        },
        "remote": {
          "remote-pad-entry-name": "remote-1"
        },
        "ietf-i2nsf-ike:child-sa-info": {
          "ietf-ipsecme-iptfs:traffic-flow-security": {
            "congestion-control": "true",
            "l2-fixed-rate": 1000000000,
            "packet-size": {
              "use-path-mtu-discovery": "true"
            },
            "max-aggregation-time": "0.1"
          }
        }
      }
    ]
  }
}
```

Figure 4: Example IP-TFS JSON Operational data

A.5. Example JSON Operational Statistics

This example shows the json formated statistics for IP-TFS. Note a unidirectional IP-TFS transmit side is illustrated, with arbitray numbers for transmit.

```
{
  "ietf-i2nsf-ikeless:ipsec-ikeless": {
    "sad": {
      "sad-entry": [
        {
          "name": "sad-1",
          "ipsec-sa-config": {
```

```
            "spi": 1,
            "traffic-selector": {
              "local-prefix": "1.1.1.1/32",
              "remote-prefix": "2.2.2.2/32"
            }
          },
          "ietf-ipsecme-iptfs:ipsec-stats": {
            "tx-pkts": "300",
            "tx-octets": "80000",
            "tx-drop-pkts": "2",
            "rx-pkts": "0",
            "rx-octets": "0",
            "rx-drop-pkts": "0"
          },
          "ietf-ipsecme-iptfs:iptfs-inner-pkt-stats": {
            "tx-pkts": "250",
            "tx-octets": "75000",
            "rx-pkts": "0",
            "rx-octets": "0",
            "rx-incomplete-pkts": "0"
          },
          "ietf-ipsecme-iptfs:iptfs-outer-pkt-stats": {
            "tx-all-pad-pkts": "40",
            "tx-all-pad-octets": "40000",
            "tx-extra-pad-pkts": "200",
            "tx-extra-pad-octets": "30000",
            "rx-all-pad-pkts": "0",
            "rx-all-pad-octets": "0",
            "rx-extra-pad-pkts": "0",
            "rx-extra-pad-octets": "0",
            "rx-errored-pkts": "0",
            "rx-missed-pkts": "0"
          },
          "ipsec-sa-state": {
            "sa-lifetime-current": {
              "time": 80000,
              "bytes": 4000606,
              "packets": 1000,
              "idle": 5
            }
          }
        }
      ]
    }
  }
}
```


Figure 5: Example IP-TFS JSON Statistics

Authors' Addresses

Don Fedyk
LabN Consulting, L.L.C.

Email: dfedyk@labn.net

Christian Hopps
LabN Consulting, L.L.C.

Email: chopps@chopps.org

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 13, 2021

V. Smyslov
ELVIS-PLUS

September 9, 2020

Announcing Supported Authentication Methods in IKEv2
draft-smyslov-ipsecme-ikev2-auth-announce-02

Abstract

This specification defines a mechanism that allows the Internet Key Exchange version 2 (IKEv2) implementations to indicate the list of supported authentication methods to their peers while establishing IKEv2 Security Association (SA). This mechanism improves interoperability when IKEv2 partners are configured with multiple different credentials to authenticate each other.

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 March 13, 2021.

Copyright Notice

Table of Contents

1. Introduction

The Internet Key Exchange version 2 (IKEv2) protocol, defined in [RFC7296], performs authenticated key exchange in IPsec. IKEv2, unlike its predecessor IKEv1, defined in [RFC2409], doesn't include a mechanism to negotiate an authentication method that the peers would use to authenticate each other. It is assumed that each peer selects whatever authentication method it thinks is appropriate, depending on authentication credentials it has.

This approach generally works well when there is no ambiguity in selecting authentication credentials. The problem may arise when there are several credentials of different type configured on one peer, while only some of them are supported on the other peer. Another problem situation is when a single credential may be used to produce different types of authentication tokens (e.g. signatures of different formats). Emerging post-quantum signature algorithms may bring additional challenges for implementations, especially if so called hybrid schemes are used (e.g. see [I-D.ounsworth-pq-composite-sigs]).

This specification defines an extension to the IKEv2 protocol that allows peers to announce their supported authentication methods, thus decreasing risks of SA establishment failure in situations when there are several ways for the peers to authenticate themselves.

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. Protocol Details

The idea is that each party sends a list of authentication methods it supports to its peer. In addition, the sending party may optionally specify that some of the authentication methods are only to be used with particular trust anchors. Upon receiving this information the peer may take it into account while selecting an algorithm for its authentication if several methods are available.

3.1. Exchanges

If the responder is willing to use this extension, it includes a new notification SUPPORTED_AUTH_METHODS in a response message of the IKE_SA_INIT exchange. This notification contains a list of authentication methods supported by the responder.

Figure 1: IKE_SA_INIT Exchange

If the initiator doesn't support this extension, it will ignore the received notification as an unknown status notify. Otherwise, it MAY send the SUPPORTED_AUTH_METHODS notification in the IKE_AUTH request message, with a list of authentication methods supported by the initiator.

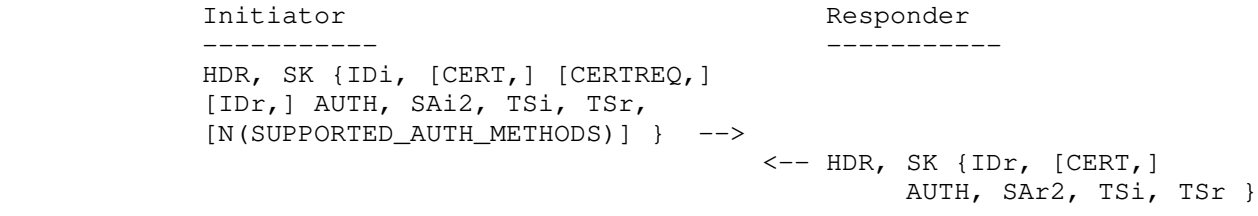


Figure 2: IKE_AUTH Exchange

Since the responder sends the SUPPORTED_AUTH_METHODS notification in the IKE_SA_INIT exchange, it must take care that the size of the response message wouldn't grow too much so that IP fragmentation takes place. If the following conditions are met:

- o the SUPPORTED_AUTH_METHODS notification to be included is so large, that the responder suspects that IP fragmentation of the resulting IKE_SA_INIT response message may happen;

- o both peers support the IKE_INTERMEDIATE exchange, defined in [I-D.ietf-ipsecme-ikev2-intermediate] (i.e. the responder has received and is going to send the INTERMEDIATE_EXCHANGE_SUPPORTED notification);

then the responder may choose not to send actual list of the supported authentication methods in the IKE_SA_INIT exchange and instead ask the initiator to start the IKE_INTERMEDIATE exchange for the list to be sent in. In this case the responder includes SUPPORTED_AUTH_METHODS notification containing no data in the IKE_SA_INIT response.

If the initiator receives the empty SUPPORTED_AUTH_METHODS notification in the IKE_SA_INIT exchange, it means that the responder is going to send the list of the supported authentication methods in the IKE_INTERMEDIATE exchange. If this exchange is to be initiated anyway for some other reason, then the responder MUST use it to send the SUPPORTED_AUTH_METHODS notification. Otherwise, the initiator MAY start the IKE_INTERMEDIATE exchange just for this sole purpose by sending an empty request message.

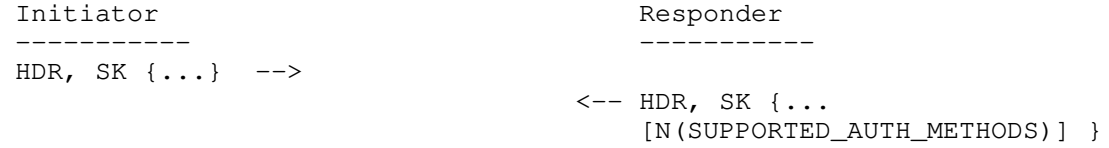


Figure 3: IKE_INTERMEDIATE Exchange

Note, that sending the SUPPORTED_AUTH_METHODS notification and using information obtained from it is optional for both the initiator and the responder.

3.2. SUPPORTED_AUTH_METHODS Notify

The format of the SUPPORTED_AUTH_METHODS notification is shown below.

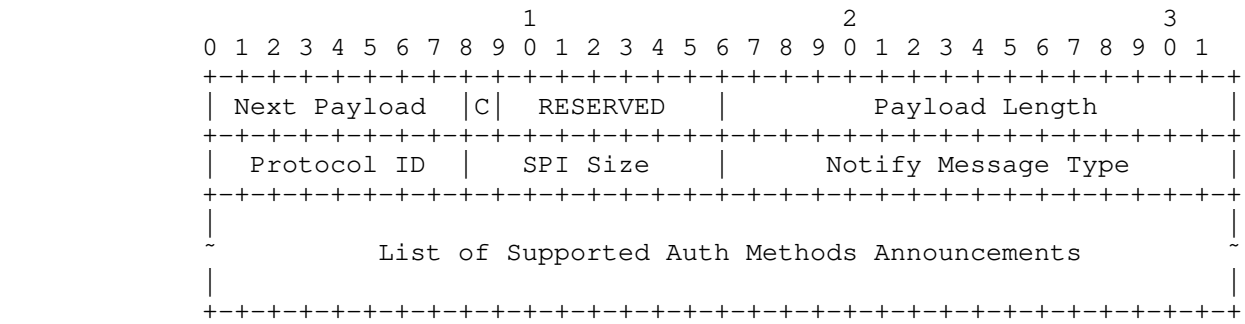


Figure 4: SUPPORTED_AUTH_METHODS Notify

The Notify payload format is defined in Section 3.10 of [RFC7296]. When a Notify payload of type SUPPORTED_AUTH_METHODS is sent, the Protocol ID field is set to 0, the SPI Size is set to 0, meaning there is no SPI field, and the Notify Message Type is set to <TBA by IANA>.

The Notification Data field contains the list of supported authentication methods announcements. Each individual announcement is a variable-size data blob, which format depends on the announced authentication method. The blob always starts with an octet containing the length of the blob followed by an octet containing the authentication method. Authentication methods are represented as values from the "IKEv2 Authentication Method" registry defined in [IKEV2-IANA]. The meaning of the remaining octets of the blob, if any, depends on the authentication method and is defined below. Note, that for the currently defined authentication methods the length octet fully defines both the format and the semantics of the blob.

If more authentication methods are defined in future, the corresponding documents must describe the semantics of the announcements for these methods. Implementations MUST skip announcements which semantics they don’t understand.

3.2.1. 2-octet Announcement

If the announcement contains an authentication method that is not concerned with public key cryptography, then the following format is used.

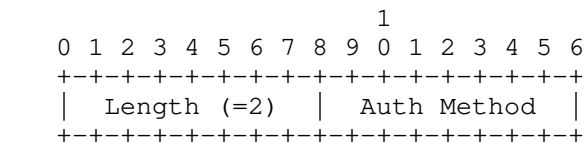


Figure 5: Supported Authentication Method

- Length - Length of the blob, must be 2 for this case.

- Auth Method - Announced authentication method.

This format is applicable for the authentication methods "Shared Key Message Integrity Code" (2) and "NULL Authentication" (13). Note, that authentication method "Generic Secure Password Authentication Method" (12) would also fall in this category, however it is negotiated separately (see [RFC6467] and for this reason there is no point to announce it via this mechanism.

3.2.2. 3-octet Announcement

If the announcement contains an authentication method that is concerned with public key cryptography, then the following format is used. This format allows to link the announcement with a particular trust anchor from the Certificate Request payload.

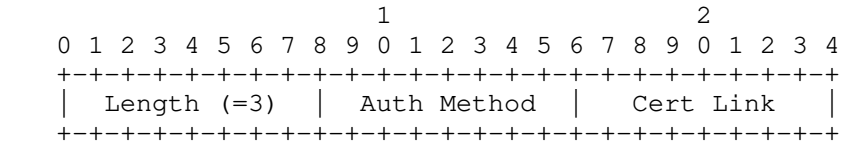


Figure 6: Supported Authentication Method

- Length - Length of the blob, must be 3 for this case.

- Auth Method - Announced authentication method.

- Cert Link - Link this announcement to a particular CA.

If the Cert Link field contains non-zero value N, it means that the announced authentication method is intended to be used only with the N-th trust anchor (CA certificate) from the Certificate Request payload(s) sent by this peer. If it is zero, then this authentication method may be used with any of CAs, that are not linked to any other announcement. If multiple CERTREQ payloads were sent, the CAs from all of them are treated as a single list for the purpose of the linking. If no Certificate Request payload were

receives, the content of this field MUST be ignored and treated as zero.

This format is applicable for the authentication methods "RSA Digital Signature" (1), "DSS Digital Signature" (3), "ECDSA with SHA-256 on the P-256 curve" (9), "ECDSA with SHA-384 on the P-384 curve" (10) and "ECDSA with SHA-512 on the P-512 curve" (11). Note however, that these authentication methods are currently superseded by the "Digital Signature" (14) authentication method, which has a different announcement format, described below.

3.2.3. Multi-octet Announcement

The following format is currently used only with the "Digital Signature" (14) authentication method.

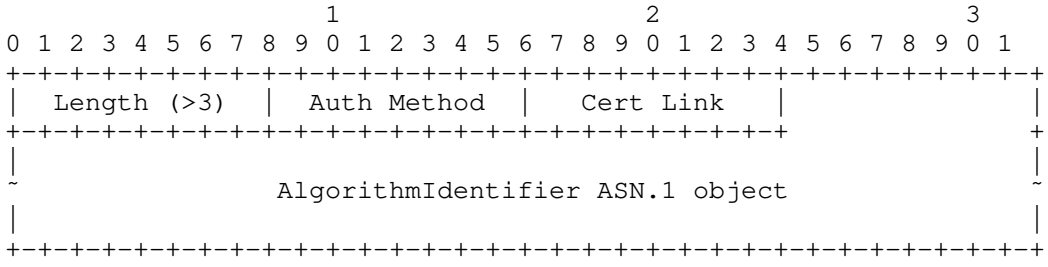


Figure 7: Supported Authentication Method

- Length - Length of the blob, must be greater than 3 for this case.
- Auth Method - Announced authentication method, currently may only be 14 ("Digital Signature").
- Cert Link - Link this announcement to a particular CA; see Section 3.2.2 for details.
- AlgorithmIdentifier ASN.1 object - DER-encoded ASN.1 object AlgorithmIdentifier.

The "Digital Signature" authentication method, defined in [RFC7427], supersedes previously defined signature authentication methods. In this case the real authentication algorithm is identified via AlgorithmIdentifier ASN.1 object. Appendix A in [RFC7427] contains examples of Commonly Used ASN.1 Objects.

4. Security Considerations

Security considerations for IKEv2 protocol are discussed in [RFC7296]. It is assumed that this extension of the IKEv2 doesn't add new vulnerabilities to the protocol.

5. IANA Considerations

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

<TBA> SUPPORTED_AUTH_METHODS

6. References

6.1. 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>.

[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>.

[RFC7427] Kivinen, T. and J. Snyder, "Signature Authentication in the Internet Key Exchange Version 2 (IKEv2)", RFC 7427, DOI 10.17487/RFC7427, January 2015, <https://www.rfc-editor.org/info/rfc7427>.

[I-D.ietf-ipsecme-ikev2-intermediate] Smyslov, V., "Intermediate Exchange in the IKEv2 Protocol", draft-ietf-ipsecme-ikev2-intermediate-04 (work in progress), June 2020.

[IKEV2-IANA] IANA, "Internet Key Exchange Version 2 (IKEv2) Parameters", <http://www.iana.org/assignments/ikev2-parameters/ikev2-parameters.xhtml#ikev2-parameters-7>.

6.2. Informative References

[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>.

[RFC6467] Kivinen, T., "Secure Password Framework for Internet Key Exchange Version 2 (IKEv2)", RFC 6467, DOI 10.17487/RFC6467, December 2011, <https://www.rfc-editor.org/info/rfc6467>.

[I-D.ounsworth-pq-composite-sigs] Ounsworth, M. and M. Pala, "Composite Keys and Signatures For Use In Internet PKI", draft-ounsworth-pq-composite-sigs-03 (work in progress), July 2020.

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
Obsoletes: 8229 (if approved)
Intended status: Standards Track
Expires: May 2, 2021

V. Smyslov
ELVIS-PLUS
T. Pauly
Apple Inc.

October 29, 2020

TCP Encapsulation of IKE and IPsec Packets
draft-smyslov-ipsecme-rfc8229bis-02

Abstract

This document describes a method to transport Internet Key Exchange Protocol (IKE) and IPsec packets over a TCP connection for traversing network middleboxes that may block IKE negotiation over UDP. This method, referred to as "TCP encapsulation", involves sending both IKE packets for Security Association establishment and Encapsulating Security Payload (ESP) packets over a TCP connection. This method is intended to be used as a fallback option when IKE cannot be negotiated over UDP.

TCP encapsulation for IKE and IPsec was defined in [RFC8229]. This document updates specification for TCP encapsulation by including additional calarifications obtained during implementation and deployment of this method. This documents makes RFC8229 obsolete.

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 2, 2021.

Table of Contents

1. Introduction

The Internet Key Exchange Protocol version 2 (IKEv2) [RFC7296] is a protocol for establishing IPsec Security Associations (SAs), using IKE messages over UDP for control traffic, and using Encapsulating Security Payload (ESP) [RFC4303] messages for encrypted data traffic. Many network middleboxes that filter traffic on public hotspots block all UDP traffic, including IKE and IPsec, but allow TCP connections through because they appear to be web traffic. Devices on these networks that need to use IPsec (to access private enterprise networks, to route Voice over IP calls to carrier networks, or because of security policies) are unable to establish IPsec SAs. This document defines a method for encapsulating IKE control messages as well as IPsec data messages within a TCP connection.

Using TCP as a transport for IPsec packets adds a third option to the list of traditional IPsec transports:

1. Direct. Currently, IKE negotiations begin over UDP port 500. If no Network Address Translation (NAT) device is detected between the Initiator and the Responder, then subsequent IKE packets are sent over UDP port 500, and IPsec data packets are sent using ESP.

2. UDP Encapsulation [RFC3948]. If a NAT is detected between the Initiator and the Responder, then subsequent IKE packets are sent over UDP port 4500 with four bytes of zero at the start of the UDP payload, and ESP packets are sent out over UDP port 4500. Some peers default to using UDP encapsulation even when no NAT is detected on the path, as some middleboxes do not support IP protocols other than TCP and UDP.

3. TCP Encapsulation. If the other two methods are not available or appropriate, IKE negotiation packets as well as ESP packets can be sent over a single TCP connection to the peer.

Direct use of ESP or UDP encapsulation should be preferred by IKE implementations due to performance concerns when using TCP encapsulation (Section 10). Most implementations should use TCP

encapsulation only on networks where negotiation over UDP has been attempted without receiving responses from the peer or if a network is known to not support UDP.

1.1. Prior Work and Motivation

Encapsulating IKE connections within TCP streams is a common approach to solve the problem of UDP packets being blocked by network middleboxes. The specific goals of this document are as follows:

- To promote interoperability by defining a standard method of framing IKE and ESP messages within TCP streams.
- To be compatible with the current IKEv2 standard without requiring modifications or extensions.
- To use IKE over UDP by default to avoid the overhead of other alternatives that always rely on TCP or Transport Layer Security (TLS) [RFC5246][RFC8446].

Some previous alternatives include:

Cellular Network Access
: Interworking Wireless LAN (IWLAN) uses IKEv2 to create secure connections to cellular carrier networks for making voice calls and accessing other network services over Wi-Fi networks. 3GPP has recommended that IKEv2 and ESP packets be sent within a TLS connection to be able to establish connections on restrictive networks.

ISAKMP over TCP
: Various non-standard extensions to the Internet Security Association and Key Management Protocol (ISAKMP) have been deployed that send IPsec traffic over TCP or TCP-like packets.

Secure Sockets Layer (SSL) VPNs
: Many proprietary VPN solutions use a combination of TLS and IPsec in order to provide reliability. These often run on TCP port 443.

IKEv2 over TCP
: IKEv2 over TCP as described in [I-D.ietf-ipsecme-ike-tcp] is used to avoid UDP fragmentation.

2. Terminology and Notation

This document distinguishes between the IKE peer that initiates TCP connections to be used for TCP encapsulation and the roles of Initiator and Responder for particular IKE messages. During the

course of IKE exchanges, the role of IKE Initiator and Responder may swap for a given SA (as with IKE SA rekeys), while the Initiator of the TCP connection is still responsible for tearing down the TCP connection and re-establishing it if necessary. For this reason, this document will use the term "TCP Originator" to indicate the IKE peer that initiates TCP connections. The peer that receives TCP connections will be referred to as the "TCP Responder". If an IKE SA is rekeyed one or more times, the TCP Originator MUST remain the peer that originally initiated the first IKE SA.

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. Configuration

One of the main reasons to use TCP encapsulation is that UDP traffic may be entirely blocked on a network. Because of this, support for TCP encapsulation is not specifically negotiated in the IKE exchange. Instead, support for TCP encapsulation must be pre-configured on both the TCP Originator and the TCP Responder.

Implementations MUST support TCP encapsulation on TCP port 4500, which is reserved for IPsec NAT traversal.

Beyond a flag indicating support for TCP encapsulation, the configuration for each peer can include the following optional parameters:

- Alternate TCP ports on which the specific TCP Responder listens for incoming connections. Note that the TCP Originator may initiate TCP connections to the TCP Responder from any local port.

- An extra framing protocol to use on top of TCP to further encapsulate the stream of IKE and IPsec packets. See Appendix B for a detailed discussion.

Since TCP encapsulation of IKE and IPsec packets adds overhead and has potential performance trade-offs compared to direct or UDP-encapsulated SAs (as described in Section 10), implementations SHOULD prefer ESP direct or UDP-encapsulated SAs over TCP-encapsulated SAs when possible.

4. TCP-Encapsulated Header Formats

Like UDP encapsulation, TCP encapsulation uses the first four bytes of a message to differentiate IKE and ESP messages. TCP encapsulation also adds a 16-bit Length field that precedes every message to define the boundaries of messages within a stream. The value in this field is equal to the length of the original message plus the length of the field itself, in octets. If the first 32 bits of the message are zeros (a non-ESP marker), then the contents comprise an IKE message. Otherwise, the contents comprise an ESP message. Authentication Header (AH) messages are not supported for TCP encapsulation.

Although a TCP stream may be able to send very long messages, implementations SHOULD limit message lengths to typical UDP datagram ESP payload lengths. The maximum message length is used as the effective MTU for connections that are being encrypted using ESP, so the maximum message length will influence characteristics of inner connections, such as the TCP Maximum Segment Size (MSS). Additionally, since TCP headers are longer than UDP headers, and TCP encapsulation adds a 16-bit Length field, some very long ESP and IKE messages that could be sent over UDP cannot be encapsulated in TCP, because their total length after encapsulation would exceed 65535 and thus could not be represented in Length field.

Note that this method of encapsulation will also work for placing IKE and ESP messages within any protocol that presents a stream abstraction, beyond TCP.

4.1. TCP-Encapsulated IKE Header Format

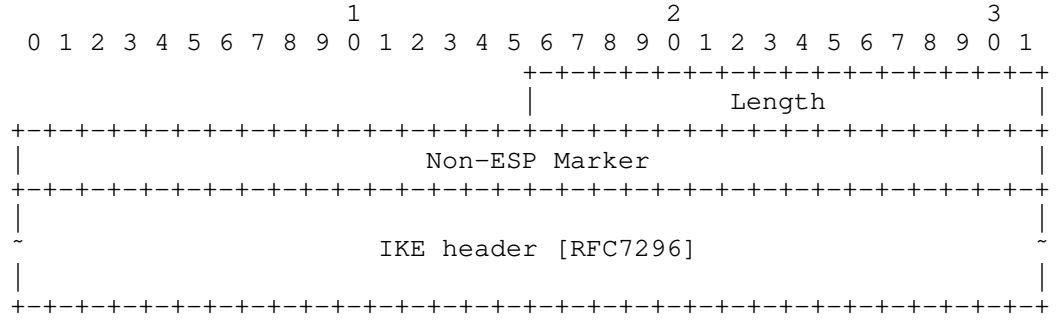


Figure 1

The IKE header is preceded by a 16-bit Length field in network byte order that specifies the length of the IKE message (including the non-ESP marker) within the TCP stream. As with IKE over UDP port

4500, a zeroed 32-bit non-ESP marker is inserted before the start of the IKE header in order to differentiate the traffic from ESP traffic between the same addresses and ports.

- Length (2 octets, unsigned integer) - Length of the IKE packet, including the Length field and non-ESP marker. The value in the Length field MUST NOT be 0 or 1. The receiver MUST treat these values as fatal errors and MUST close TCP connection.

4.2. TCP-Encapsulated ESP Header Format

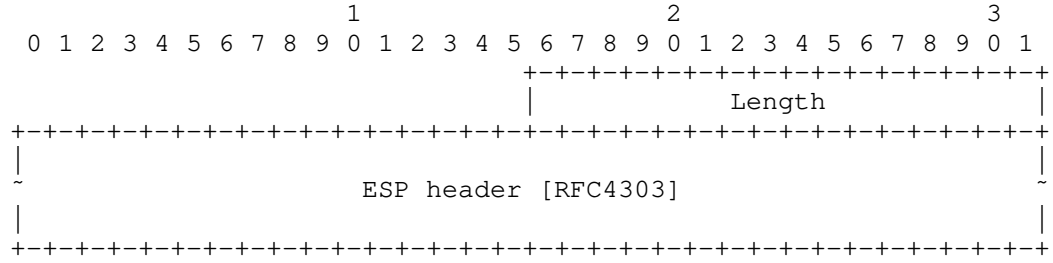


Figure 2

The ESP header is preceded by a 16-bit Length field in network byte order that specifies the length of the ESP packet within the TCP stream.

The Security Parameter Index (SPI) field [RFC7296] in the ESP header MUST NOT be a zero value.

- Length (2 octets, unsigned integer) - Length of the ESP packet, including the Length field. The value in the Length field MUST NOT be 0 or 1. The receiver MUST treat these values as fatal errors and MUST close TCP connection.

5. TCP-Encapsulated Stream Prefix

Each stream of bytes used for IKE and IPsec encapsulation MUST begin with a fixed sequence of six bytes as a magic value, containing the characters "IKETCP" as ASCII values. This value is intended to identify and validate that the TCP connection is being used for TCP encapsulation as defined in this document, to avoid conflicts with the prevalence of previous non-standard protocols that used TCP port 4500. This value is only sent once, by the TCP Originator only, at the beginning of any stream of IKE and ESP messages.

If other framing protocols are used within TCP to further encapsulate or encrypt the stream of IKE and ESP messages, the stream prefix must

be at the start of the TCP Originator's IKE and ESP message stream within the added protocol layer (Appendix B). Although some framing protocols do support negotiating inner protocols, the stream prefix should always be used in order for implementations to be as generic as possible and not rely on other framing protocols on top of TCP.

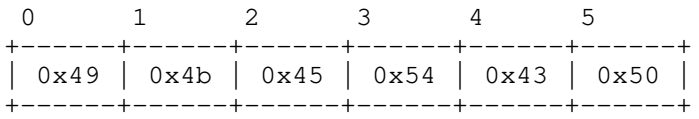


Figure 3

6. Applicability

TCP encapsulation is applicable only when it has been configured to be used with specific IKE peers. If a Responder is configured to use TCP encapsulation, it MUST listen on the configured port(s) in case any peers will initiate new IKE sessions. Initiators MAY use TCP encapsulation for any IKE session to a peer that is configured to support TCP encapsulation, although it is recommended that Initiators should only use TCP encapsulation when traffic over UDP is blocked.

Since the support of TCP encapsulation is a configured property, not a negotiated one, it is recommended that if there are multiple IKE endpoints representing a single peer (such as multiple machines with different IP addresses when connecting by Fully Qualified Domain Name, or endpoints used with IKE redirection), all of the endpoints equally support TCP encapsulation.

If TCP encapsulation is being used for a specific IKE SA, all messages for that IKE SA and its Child SAs MUST be sent over a TCP connection until the SA is deleted or IKEv2 Mobility and Multihoming (MOBIKE) is used to change the SA endpoints and/or the encapsulation protocol. See Section 8.1 for more details on using MOBIKE to transition between encapsulation modes.

6.1. Recommended Fallback from UDP

Since UDP is the preferred method of transport for IKE messages, implementations that use TCP encapsulation should have an algorithm for deciding when to use TCP after determining that UDP is unusable. If an Initiator implementation has no prior knowledge about the network it is on and the status of UDP on that network, it SHOULD always attempt to negotiate IKE over UDP first. IKEv2 defines how to use retransmission timers with IKE messages and, specifically, IKE_SA_INIT messages [RFC7296]. Generally, this means that the implementation will define a frequency of retransmission and the

maximum number of retransmissions allowed before marking the IKE SA as failed. An implementation can attempt negotiation over TCP once it has hit the maximum retransmissions over UDP, or slightly before to reduce connection setup delays. It is recommended that the initial message over UDP be retransmitted at least once before falling back to TCP, unless the Initiator knows beforehand that the network is likely to block UDP.

When switching from UDP to TCP, a new IKE_SA_INIT exchange MUST be initiated with new Initiator's SPI and with recalculated content of NAT_DETECTION_SOURCE_IP notification.

7. Using TCP Encapsulation

7.1. Connection Establishment and Teardown

When the IKE Initiator uses TCP encapsulation, it will initiate a TCP connection to the Responder using the configured TCP port. The first bytes sent on the stream MUST be the stream prefix value (Section 5). After this prefix, encapsulated IKE messages will negotiate the IKE SA and initial Child SA [RFC7296]. After this point, both encapsulated IKE (Figure 1) and ESP (Figure 2) messages will be sent over the TCP connection. The TCP Responder MUST wait for the entire stream prefix to be received on the stream before trying to parse out any IKE or ESP messages. The stream prefix is sent only once, and only by the TCP Originator.

In order to close an IKE session, either the Initiator or Responder SHOULD gracefully tear down IKE SAs with DELETE payloads. Once the SA has been deleted, the TCP Originator SHOULD close the TCP connection if it does not intend to use the connection for another IKE session to the TCP Responder. If the connection is left idle and the TCP Responder needs to clean up resources, the TCP Responder MAY close the TCP connection.

An unexpected FIN or a TCP Reset on the TCP connection may indicate a loss of connectivity, an attack, or some other error. If a DELETE payload has not been sent, both sides SHOULD maintain the state for their SAs for the standard lifetime or timeout period. The TCP Originator is responsible for re-establishing the TCP connection if it is torn down for any unexpected reason. Since new TCP connections may use different ports due to NAT mappings or local port allocations changing, the TCP Responder MUST allow packets for existing SAs to be received from new source ports.

A peer MUST discard a partially received message due to a broken connection.

Whenever the TCP Originator opens a new TCP connection to be used for an existing IKE SA, it MUST send the stream prefix first, before any IKE or ESP messages. This follows the same behavior as the initial TCP connection.

If a TCP connection is being used to resume a previous IKE session, the TCP Responder can recognize the session using either the IKE SPI from an encapsulated IKE message or the ESP SPI from an encapsulated ESP message. If the session had been fully established previously, it is suggested that the TCP Originator send an UPDATE_SA_ADDRESSES message if MOBIKE is supported, or an informational message (a keep-alive) otherwise.

The TCP Responder MUST NOT accept any messages for the existing IKE session on a new incoming connection, unless that connection begins with the stream prefix. If either the TCP Originator or TCP Responder detects corruption on a connection that was started with a valid stream prefix, it SHOULD close the TCP connection. The connection can be determined to be corrupted if there are too many subsequent messages that cannot be parsed as valid IKE messages or ESP messages with known SPIs, or if the authentication check for an ESP message with a known SPI fails. Implementations SHOULD NOT tear down a connection if only a single ESP message has an unknown SPI, since the SPI databases may be momentarily out of sync. If there is instead a syntax issue within an IKE message, an implementation MUST send the INVALID_SYNTAX notify payload and tear down the IKE SA as usual, rather than tearing down the TCP connection directly.

A TCP Originator SHOULD only open one TCP connection per IKE SA, over which it sends all of the corresponding IKE and ESP messages. This helps ensure that any firewall or NAT mappings allocated for the TCP connection apply to all of the traffic associated with the IKE SA equally.

Similarly, a TCP Responder SHOULD at any given time send packets for an IKE SA and its Child SAs over only one TCP connection. It SHOULD choose the TCP connection on which it last received a valid and decryptable IKE or ESP message. In order to be considered valid for choosing a TCP connection, an IKE message must be successfully decrypted and authenticated, not be a retransmission of a previously received message, and be within the expected window for IKE message IDs. Similarly, an ESP message must pass authentication checks and be decrypted, and must not be a replay of a previous message.

Since a connection may be broken and a new connection re-established by the TCP Originator without the TCP Responder being aware, a TCP Responder SHOULD accept receiving IKE and ESP messages on both old and new connections until the old connection is closed by the TCP

Originator. A TCP Responder MAY close a TCP connection that it perceives as idle and extraneous (one previously used for IKE and ESP messages that has been replaced by a new connection).

Multiple IKE SAs MUST NOT share a single TCP connection, unless one is a rekey of an existing IKE SA, in which case there will temporarily be two IKE SAs on the same TCP connection.

7.2. Retransmissions

Section 2.1 of [RFC7296] describes how IKEv2 deals with the unreliability of the UDP protocol. In brief, the exchange Initiator is responsible for retransmissions and must retransmit requests message until response message is received. If no reply is received after several retransmissions, the SA is deleted. The Responder never initiates retransmission, but must send a response message again in case it receives a retransmitted request.

When IKEv2 uses a reliable transport protocol, like TCP, the retransmission rules are as follows:

- the exchange Initiator SHOULD NOT retransmit request message; if no response is received within some reasonable period of time, the IKE SA is deleted.

- if a TCP connection is broken and reestablished while the exchange Initiator is waiting for a response, the Initiator MUST retransmit its request and continue to wait for a response.

- the exchange Responder does not change its behavior, but acts as described in Section 2.1 of [RFC7296].

7.3. Cookies and Puzzles

IKEv2 provides a DoS attack protection mechanism through Cookies, which is described in Section 2.6 of [RFC7296]. [RFC8019] extends this mechanism for protection against DDoS attacks by means of Client Puzzles. Both mechanisms allow the Responder to avoid keeping state until the Initiator proves its IP address is legitimate (and after solving a puzzle if required).

The connection-oriented nature of TCP and transport brings additional considerations for using these mechanisms. In general, Cookies provide less value in case of TCP encapsulation, since by the time a Responder receives the IKE_SA_INIT request, the TCP session has already been established and the Initiator's IP address has been verified. Moreover, a TCP Responder creates state once a SYN packet is received (unless SYN Cookies described in [RFC4987] are employed),

which eliminates some of the benefits of IKEv2 Cookies. When using TCP encapsulation, it adds little value to send Cookie requests without Puzzles unless the Responder is concerned with the possibility of TCP Sequence Number attacks (see [RFC6528] for details). Puzzles, on the other hand, still remain useful (and their use requires using Cookies).

The following considerations are applicable for using Cookie and Puzzle mechanisms in case of TCP encapsulation:

- the exchange Responder SHOULD NOT request a Cookie, with the exception of Puzzles or for rare cases like preventing TCP Sequence Number attacks.

- if the Responder chooses to send Cookie request (possibly along with Puzzle request), then the TCP connection that the IKE_SA_INIT request message was received over SHOULD be closed, so that the Responder remains stateless at least until the Cookie (or Puzzle Solution) is returned. Note that if this TCP connection is closed, the Responder MUST NOT include the Initiator's TCP port into the Cookie calculation (*), since the Cookie will be returned over a new TCP connection with a different port.

- the exchange Initiator acts as described in Section 2.6 of [RFC7296] and Section 7 of [RFC8019], i.e. using TCP encapsulation doesn't change the Initiator's behavior.

(*) Examples of Cookie calculation methods are given in Section 2.6 of [RFC7296] and in Section 7.1.1.3 of [RFC8019] and they don't include transport protocol ports. However these examples are given for illustrative purposes, since Cookie generation algorithm is a local matter and some implementations might include port numbers, that won't work with TCP encapsulation.

7.4. Error Handling in IKE_SA_INIT

Section 2.21.1 of [RFC7296] describes how error notifications are handled in the IKE_SA_INIT exchange. In particular, it is advised that the Initiator should not act immediately after receiving error notification and should instead wait some time for valid response, since the IKE_SA_INIT messages are completely unauthenticated. This advice does not apply equally in case of TCP encapsulation. If the Initiator receives a response message over TCP, then either this message is genuine and was sent by the peer, or the TCP session was hijacked and the message is forged. In this latter case, no genuine messages from the Responder will be received.

Thus, in case of TCP encapsulation, an Initiator SHOULD NOT wait for additional messages in case it receives error notification from the Responder in the IKE_SA_INIT exchange.

7.5. NAT Detection Payloads

When negotiating over UDP port 500, IKE_SA_INIT packets include NAT_DETECTION_SOURCE_IP and NAT_DETECTION_DESTINATION_IP payloads to determine if UDP encapsulation of IPsec packets should be used. These payloads contain SHA-1 digests of the SPIs, IP addresses, and ports as defined in [RFC7296]. IKE_SA_INIT packets sent on a TCP connection SHOULD include these payloads with the same content as when sending over UDP and SHOULD use the applicable TCP ports when creating and checking the SHA-1 digests.

If a NAT is detected due to the SHA-1 digests not matching the expected values, no change should be made for encapsulation of subsequent IKE or ESP packets, since TCP encapsulation inherently supports NAT traversal. Implementations MAY use the information that a NAT is present to influence keep-alive timer values.

If a NAT is detected, implementations need to handle transport mode TCP and UDP packet checksum fixup as defined for UDP encapsulation in [RFC3948].

7.6. Keep-Alives and Dead Peer Detection

Encapsulating IKE and IPsec inside of a TCP connection can impact the strategy that implementations use to detect peer liveness and to maintain middlebox port mappings. Peer liveness should be checked using IKE informational packets [RFC7296].

In general, TCP port mappings are maintained by NATs longer than UDP port mappings, so IPsec ESP NAT keep-alives [RFC3948] SHOULD NOT be sent when using TCP encapsulation. Any implementation using TCP encapsulation MUST silently drop incoming NAT keep-alive packets and not treat them as errors. NAT keep-alive packets over a TCP-encapsulated IPsec connection will be sent as an ESP message with a one-octet-long payload with the value 0xFF.

Note that, depending on the configuration of TCP and TLS on the connection, TCP keep-alives [RFC1122] and TLS keep-alives [RFC6520] may be used. These MUST NOT be used as indications of IKE peer liveness.

7.7. Implications of TCP Encapsulation on IPsec SA Processing

Using TCP encapsulation affects some aspects of IPsec SA processing.

1. Section 8.1 of [RFC4301] requires all tunnel mode IPsec SAs to be able to copy the Don't Fragment (DF) bit from inner IP header to the outer (tunnel) one. With TCP encapsulation this is generally not possible, because TCP/IP stack manages DF bit in the outer IP header, and usually the stack ensures that the DF bit is set for TCP packets to avoid IP fragmentation.

2. The other feature that is less applicable with TCP encapsulation is an ability to split traffic of different QoS classes into different IPsec SAs, created by a single IKE SA. In this case the Differentiated Services Code Point (DSCP) field is usually copied from the inner IP header to the outer (tunnel) one, ensuring that IPsec traffic of each SA receives the corresponding level of service. With TCP encapsulation all IPsec SAs created by a single IKE SA will share a single TCP connection and thus will receive the same level of service (see Section 10.3). If this functionality is needed, implementations should create several IKE SAs over TCP and assign a corresponding DSCP value to each of them.

8. Interaction with IKEv2 Extensions

8.1. MOBIKE Protocol

MOBIKE protocol, that allows IKEv2 SA to migrate between IP addresses, is defined in [RFC4555], and [RFC4621] further clarifies the details of the protocol. When an IKE session that has negotiated MOBIKE is transitioning between networks, the Initiator of the transition may switch between using TCP encapsulation, UDP encapsulation, or no encapsulation. Implementations that implement both MOBIKE and TCP encapsulation MUST support dynamically enabling and disabling TCP encapsulation as interfaces change.

When a MOBIKE-enabled Initiator changes networks, the INFORMATIONAL exchange with the UPDATE_SA_ADDRESSES notification SHOULD be initiated first over UDP before attempting over TCP. If there is a response to the request sent over UDP, then the ESP packets should be sent directly over IP or over UDP port 4500 (depending on if a NAT was detected), regardless of if a connection on a previous network was using TCP encapsulation. If no response is received within a certain period of time after several retransmissions, the Initiator ought to change its transport for this exchange from UDP to TCP and resend the request message. New INFORMATIONAL exchange MUST NOT be started in this situation. If the Responder only responds to the

request sent over TCP, then the ESP packets should be sent over the TCP connection, regardless of if a connection on a previous network did not use TCP encapsulation.

Since switching from UDP to TCP happens can occur during a single INFORMATIONAL message exchange, the content of the NAT_DETECTION_SOURCE_IP notification will in most cases be incorrect (since UDP and TCP source ports will most likely be different), and the peer may incorrectly detect the presence of a NAT. This should not cause functional issues since all messages will be encapsulated in TCP anyway, and TCP encapsulation does not change based on the presence of NATs.

MOBIKE protocol defined the NO_NATS_ALLOWED notification that can be used to detect the presence of NAT between peer and to refuse to communicate in this situation. In case of TCP the NO_NATS_ALLOWED notification SHOULD be ignored because TCP generally has no problems with NAT boxes.

Section 3.7 of [RFC4555] describes an additional optional step in the process of changing IP addresses called Return Routability Check. It is performed by the responder in order to be sure that the new initiator's address is in fact routable. In case of TCP encapsulation this check has little value, since TCP handshake proves routability of the TCP Originator's address. So, in case of TCP encapsulation the Return Routability Check SHOULD NOT be performed.

8.2. IKE Redirect

A redirect mechanism for IKEv2 is defined in [RFC5685]. This mechanism allows security gateways to redirect clients to another gateway either during IKE SA establishment or after session setup. If a client is connecting to a security gateway using TCP and then is redirected to another security gateway, the client needs to reset its transport selection. In other words, the client MUST again try first UDP and then fall back to TCP while establishing a new IKE SA, regardless of the transport of the SA the redirect notification was received over (unless the client's configuration instructs it to instantly use TCP for the gateway it is redirected to).

8.3. IKEv2 Session Resumption

Session resumption for IKEv2 is defined in [RFC5723]. Once an IKE SA is established, the server creates a resumption ticket where information about this SA is stored, and transfers this ticket to the client. The ticket may be later used to resume the IKE SA after it is deleted. In the event of resumption the client presents the ticket in a new exchange, called IKE_SESSION_RESUME. Some parameters

in the new SA are retrieved from the ticket and others are re-negotiated (more details are given in Section 5 of [RFC5723]). If TCP encapsulation was used in an old SA, then the client SHOULD resume this SA using TCP, without first trying to connect over UDP.

8.4. IKEv2 Protocol Support for High Availability

[RFC6311] defines a support for High Availability in IKEv2. In case of cluster failover, a new active node must immediately initiate a special INFORMATION exchange containing the IKEV2_MESSAGE_ID_SYNC notification, which instructs the client to skip some number of Message IDs that might not be synchronized yet between nodes at the time of failover.

Synchronizing states when using TCP encapsulation is much harder than when using UDP; doing so requires access to TCP/IP stack internals, which is not always available from an IKE/IPsec implementation. If a cluster implementation doesn't synchronize TCP states between nodes, then after failover event the new active node will not have any TCP connection with the client, so the node cannot initiate the INFORMATIONAL exchange as required by [RFC6311]. Since the cluster usually acts as TCP Responder, the new active node cannot re-establish TCP connection, since only the TCP Originator can do it. For the client, the cluster failover event may remain undetected for long time if it has no IKE or ESP traffic to send. Once the client sends an ESP or IKEv2 packet, the cluster node will reply with TCP RST and the client (as TCP Originator) will reestablish the TCP connection so that the node will be able to initiate the INFORMATIONAL exchange informing the client about the cluster failover.

This document makes the following recommendation: if support for High Availability in IKEv2 is negotiated and TCP transport is used, a client that is a TCP Originator SHOULD periodically send IKEv2 messages (e.g. by initiating liveness check exchange) whenever there is no IKEv2 or ESP traffic. This differs from the recommendations given in Section 2.4 of [RFC7296] in the following: the liveness check should be periodically performed even if the client has nothing to send over ESP. The frequency of sending such messages should be high enough to allow quick detection and restoring of broken TCP connection.

8.5. IKEv2 Fragmentation

IKE message fragmentation [RFC7383] is not required when using TCP encapsulation, since a TCP stream already handles the fragmentation of its contents across packets. Since fragmentation is redundant in this case, implementations might choose to not negotiate IKE

fragmentation. Even if fragmentation is negotiated, an implementation SHOULD NOT send fragments when going over a TCP connection, although it MUST support receiving fragments.

If an implementation supports both MOBIKE and IKE fragmentation, it SHOULD negotiate IKE fragmentation over a TCP-encapsulated session in case the session switches to UDP encapsulation on another network.

9. Middlebox Considerations

Many security networking devices, such as firewalls or intrusion prevention systems, network optimization/acceleration devices, and NAT devices, keep the state of sessions that traverse through them.

These devices commonly track the transport-layer and/or application-layer data to drop traffic that is anomalous or malicious in nature. While many of these devices will be more likely to pass TCP-encapsulated traffic as opposed to UDP-encapsulated traffic, some may still block or interfere with TCP-encapsulated IKE and IPsec traffic.

A network device that monitors the transport layer will track the state of TCP sessions, such as TCP sequence numbers. TCP encapsulation of IKE should therefore use standard TCP behaviors to avoid being dropped by middleboxes.

10. Performance Considerations

Several aspects of TCP encapsulation for IKE and IPsec packets may negatively impact the performance of connections within a tunnel-mode IPsec SA. Implementations should be aware of these performance impacts and take these into consideration when determining when to use TCP encapsulation. Implementations SHOULD favor using direct ESP or UDP encapsulation over TCP encapsulation whenever possible.

10.1. TCP-in-TCP

If the outer connection between IKE peers is over TCP, inner TCP connections may suffer negative effects from using TCP within TCP. Running TCP within TCP is discouraged, since the TCP algorithms generally assume that they are running over an unreliable datagram layer.

If the outer (tunnel) TCP connection experiences packet loss, this loss will be hidden from any inner TCP connections, since the outer connection will retransmit to account for the losses. Since the outer TCP connection will deliver the inner messages in order, any messages after a lost packet may have to wait until the loss is recovered. This means that loss on the outer connection will be

interpreted only as delay by inner connections. The burstiness of inner traffic can increase, since a large number of inner packets may be delivered across the tunnel at once. The inner TCP connection may interpret a long period of delay as a transmission problem, triggering a retransmission timeout, which will cause spurious retransmissions. The sending rate of the inner connection may be unnecessarily reduced if the retransmissions are not detected as spurious in time.

The inner TCP connection's round-trip-time estimation will be affected by the burstiness of the outer TCP connection if there are long delays when packets are retransmitted by the outer TCP connection. This will make the congestion control loop of the inner TCP traffic less reactive, potentially permanently leading to a lower sending rate than the outer TCP would allow for.

TCP-in-TCP can also lead to increased buffering, or bufferbloat. This can occur when the window size of the outer TCP connection is reduced and becomes smaller than the window sizes of the inner TCP connections. This can lead to packets backing up in the outer TCP connection's send buffers. In order to limit this effect, the outer TCP connection should have limits on its send buffer size and on the rate at which it reduces its window size.

Note that any negative effects will be shared between all flows going through the outer TCP connection. This is of particular concern for any latency-sensitive or real-time applications using the tunnel. If such traffic is using a TCP-encapsulated IPsec connection, it is recommended that the number of inner connections sharing the tunnel be limited as much as possible.

10.2. Added Reliability for Unreliable Protocols

Since ESP is an unreliable protocol, transmitting ESP packets over a TCP connection will change the fundamental behavior of the packets. Some application-level protocols that prefer packet loss to delay (such as Voice over IP or other real-time protocols) may be negatively impacted if their packets are retransmitted by the TCP connection due to packet loss.

10.3. Quality-of-Service Markings

Quality-of-Service (QoS) markings, such as the Differentiated Services Code Point (DSCP) and Traffic Class, should be used with care on TCP connections used for encapsulation. Individual packets SHOULD NOT use different markings than the rest of the connection, since packets with different priorities may be routed differently and cause unnecessary delays in the connection.

10.4. Maximum Segment Size

A TCP connection used for IKE encapsulation SHOULD negotiate its MSS in order to avoid unnecessary fragmentation of packets.

10.5. Tunneling ECN in TCP

Since there is not a one-to-one relationship between outer IP packets and inner ESP/IP messages when using TCP encapsulation, the markings for Explicit Congestion Notification (ECN) [RFC3168] cannot be simply mapped. However, any ECN Congestion Experienced (CE) marking on inner headers should be preserved through the tunnel.

Implementations SHOULD follow the ECN compatibility mode for tunnel ingress as described in [RFC6040]. In compatibility mode, the outer tunnel TCP connection marks its packet headers as not ECN-capable. If upon egress, the arriving outer header is marked with CE, the implementation will drop the inner packet, since there is not a distinct inner packet header onto which to translate the ECN markings.

11. Security Considerations

IKE Responders that support TCP encapsulation may become vulnerable to new Denial-of-Service (DoS) attacks that are specific to TCP, such as SYN-flooding attacks. TCP Responders should be aware of this additional attack surface.

TCP Responders should be careful to ensure that (1) the stream prefix "IKETCP" uniquely identifies incoming streams as streams that use the TCP encapsulation protocol and (2) they are not running any other protocols on the same listening port (to avoid potential conflicts).

Attackers may be able to disrupt the TCP connection by sending spurious TCP Reset packets. Therefore, implementations SHOULD make sure that IKE session state persists even if the underlying TCP connection is torn down.

If MOBIKE is being used, all of the security considerations outlined for MOBIKE apply [RFC4555].

Similarly to MOBIKE, TCP encapsulation requires a TCP Responder to handle changes to source address and port due to network or connection disruption. The successful delivery of valid IKE or ESP messages over a new TCP connection is used by the TCP Responder to determine where to send subsequent responses. If an attacker is able to send packets on a new TCP connection that pass the validation checks of the TCP Responder, it can influence which path future

packets will take. For this reason, the validation of messages on the TCP Responder must include decryption, authentication, and replay checks.

Since TCP provides reliable, in-order delivery of ESP messages, the ESP anti-replay window size SHOULD be set to 1. See [RFC4303] for a complete description of the ESP anti-replay window. This increases the protection of implementations against replay attacks.

12. IANA Considerations

TCP port 4500 is already allocated to IPsec for NAT traversal. This port SHOULD be used for TCP-encapsulated IKE and ESP as described in this document.

This document updates the reference for TCP port 4500 from RFC 8229 to itself:

Keyword	Decimal	Description	Reference
ipsec-nat-t	4500/tcp	IPsec NAT-Traversal	[RFCXXXX]

Figure 4

13. References

13.1. 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>.

[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>.

[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>.

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

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

[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>.

[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>.

[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>.

13.2. Informative References

[I-D.ietf-ipsecme-ike-tcp] Nir, Y., "A TCP transport for the Internet Key Exchange", draft-ietf-ipsecme-ike-tcp-01 (work in progress), December 2012.

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <https://www.rfc-editor.org/info/rfc1122>.

[RFC2817] Khare, R. and S. Lawrence, "Upgrading to TLS Within HTTP/1.1", RFC 2817, DOI 10.17487/RFC2817, May 2000, <https://www.rfc-editor.org/info/rfc2817>.

[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>.

[RFC4555] Eronen, P., "IKEv2 Mobility and Multihoming Protocol (MOBIKE)", RFC 4555, DOI 10.17487/RFC4555, June 2006, <https://www.rfc-editor.org/info/rfc4555>.

[RFC4621] Kivinen, T. and H. Tschofenig, "Design of the IKEv2 Mobility and Multihoming (MOBIKE) Protocol", RFC 4621, DOI 10.17487/RFC4621, August 2006, <https://www.rfc-editor.org/info/rfc4621>.

[RFC4987] Eddy, W., "TCP SYN Flooding Attacks and Common Mitigations", RFC 4987, DOI 10.17487/RFC4987, August 2007, <https://www.rfc-editor.org/info/rfc4987>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[RFC5685] Devarapalli, V. and K. Weniger, "Redirect Mechanism for the Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 5685, DOI 10.17487/RFC5685, November 2009, <https://www.rfc-editor.org/info/rfc5685>.

[RFC5723] Sheffer, Y. and H. Tschofenig, "Internet Key Exchange Protocol Version 2 (IKEv2) Session Resumption", RFC 5723, DOI 10.17487/RFC5723, January 2010, <https://www.rfc-editor.org/info/rfc5723>.

[RFC6311] Singh, R., Ed., Kalyani, G., Nir, Y., Sheffer, Y., and D. Zhang, "Protocol Support for High Availability of IKEv2/ IPsec", RFC 6311, DOI 10.17487/RFC6311, July 2011, <https://www.rfc-editor.org/info/rfc6311>.

[RFC6520] Seggelmann, R., Tuexen, M., and M. Williams, "Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension", RFC 6520, DOI 10.17487/RFC6520, February 2012, <https://www.rfc-editor.org/info/rfc6520>.

[RFC6528] Gont, F. and S. Bellovin, "Defending against Sequence Number Attacks", RFC 6528, DOI 10.17487/RFC6528, February 2012, <https://www.rfc-editor.org/info/rfc6528>.

[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>.

[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>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

Appendix A. Using TCP Encapsulation with TLS

This section provides recommendations on how to use TLS in addition to TCP encapsulation.

When using TCP encapsulation, implementations may choose to use TLS 1.2 [RFC5246] or TLS 1.3 [RFC8446] on the TCP connection to be able to traverse middleboxes, which may otherwise block the traffic.

If a web proxy is applied to the ports used for the TCP connection and TLS is being used, the TCP Originator can send an HTTP CONNECT message to establish an SA through the proxy [RFC2817].

The use of TLS should be configurable on the peers, and may be used as the default when using TCP encapsulation or may be used as a fallback when basic TCP encapsulation fails. The TCP Responder may expect to read encapsulated IKE and ESP packets directly from the TCP connection, or it may expect to read them from a stream of TLS data packets. The TCP Originator should be pre-configured to use TLS or not when communicating with a given port on the TCP Responder.

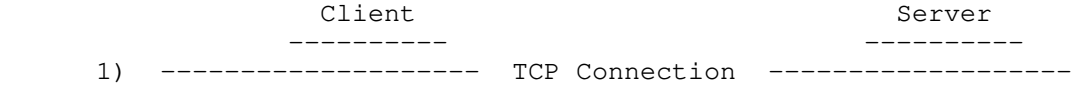
When new TCP connections are re-established due to a broken connection, TLS must be renegotiated. TLS session resumption is recommended to improve efficiency in this case.

The security of the IKE session is entirely derived from the IKE negotiation and key establishment and not from the TLS session (which in this context is only used for encapsulation purposes); therefore, when TLS is used on the TCP connection, both the TCP Originator and the TCP Responder SHOULD allow the NULL cipher to be selected for performance reasons. Note, that TLS 1.3 only supports AEAD algorithms and at the time of writing this document there was no recommended cipher suite for TLS 1.3 with the NULL cipher.

Implementations should be aware that the use of TLS introduces another layer of overhead requiring more bytes to transmit a given IKE and IPsec packet. For this reason, direct ESP, UDP encapsulation, or TCP encapsulation without TLS should be preferred in situations in which TLS is not required in order to traverse middleboxes.

Appendix B. Example Exchanges of TCP Encapsulation with TLS 1.3

B.1. Establishing an IKE Session



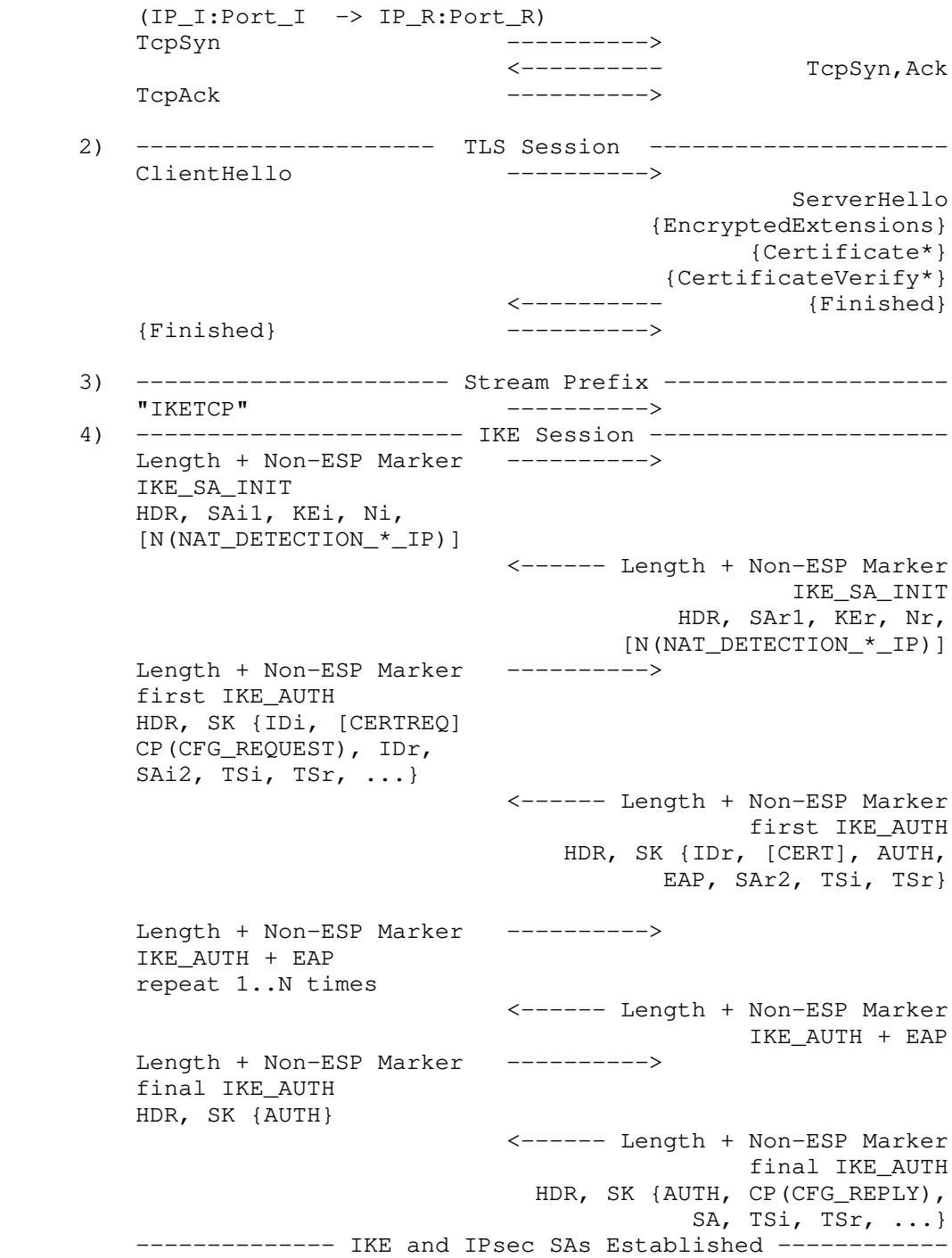

(IP_I:Port_I -> IP_R:Port_R)
TcpSyn
TcpSyn,Ack
TcpAck
2) TLS Session
ClientHello
ServerHello
{EncryptedExtensions}
{Certificate*}
{CertificateVerify*}
{Finished}
{Finished}
3) Stream Prefix
"IKETCP"
4) IKE Session
Length + Non-ESP Marker
IKE_SA_INIT
HDR, SAi1, KEi, Ni,
[N(NAT_DETECTION_*_IP)]
Length + Non-ESP Marker
IKE_SA_INIT
HDR, SAr1, KEr, Nr,
[N(NAT_DETECTION_*_IP)]
Length + Non-ESP Marker
first IKE_AUTH
HDR, SK {IDi, [CERTREQ]
CP(CFG_REQUEST), IDr,
SAi2, TSi, TSr, ...}
Length + Non-ESP Marker
first IKE_AUTH
HDR, SK {IDr, [CERT], AUTH,
EAP, SAr2, TSi, TSr}
Length + Non-ESP Marker
IKE_AUTH + EAP
repeat 1..N times
Length + Non-ESP Marker
IKE_AUTH + EAP
Length + Non-ESP Marker
final IKE_AUTH
HDR, SK {AUTH}
Length + Non-ESP Marker
final IKE_AUTH
HDR, SK {AUTH, CP(CFG_REPLY),
SA, TSi, TSr, ...}
IKE and IPsec SAs Established

Figure 5

1. The client establishes a TCP connection with the server on port 4500 or on an alternate pre-configured port that the server is listening on.

2. If configured to use TLS, the client initiates a TLS handshake. During the TLS handshake, the server SHOULD NOT request the client's certificate, since authentication is handled as part of IKE negotiation.

3. The client sends the stream prefix for TCP-encapsulated IKE (Section 5) traffic to signal the beginning of IKE negotiation.

4. The client and server establish an IKE connection. This example shows EAP-based authentication, although any authentication type may be used.

B.2. Deleting an IKE Session

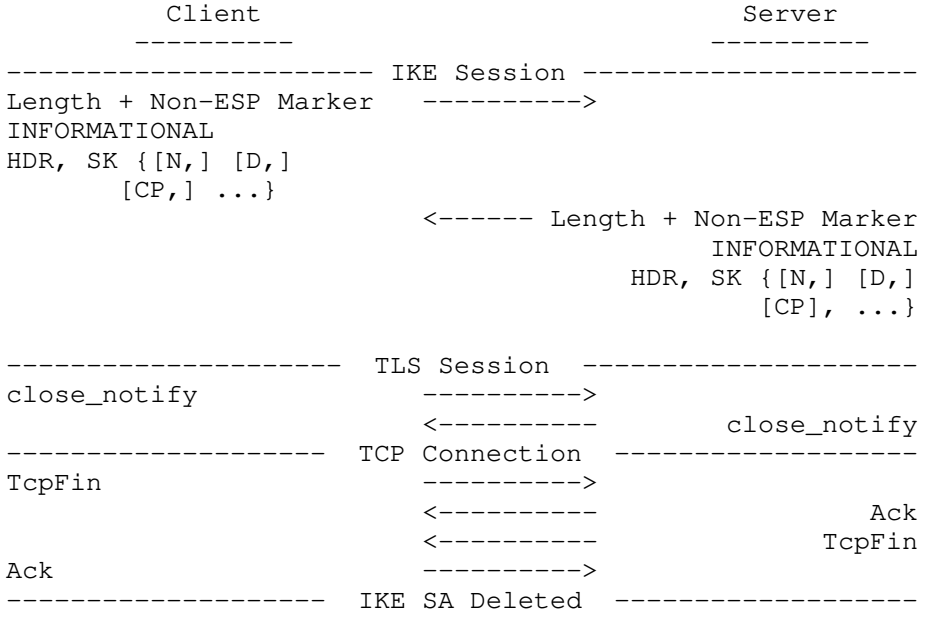


Figure 6

1. The client and server exchange informational messages to notify IKE SA deletion.

2. The client and server negotiate TLS session deletion using TLS CLOSE_NOTIFY.

3. The TCP connection is torn down.

The deletion of the IKE SA should lead to the disposal of the underlying TLS and TCP state.

B.3. Re-establishing an IKE Session

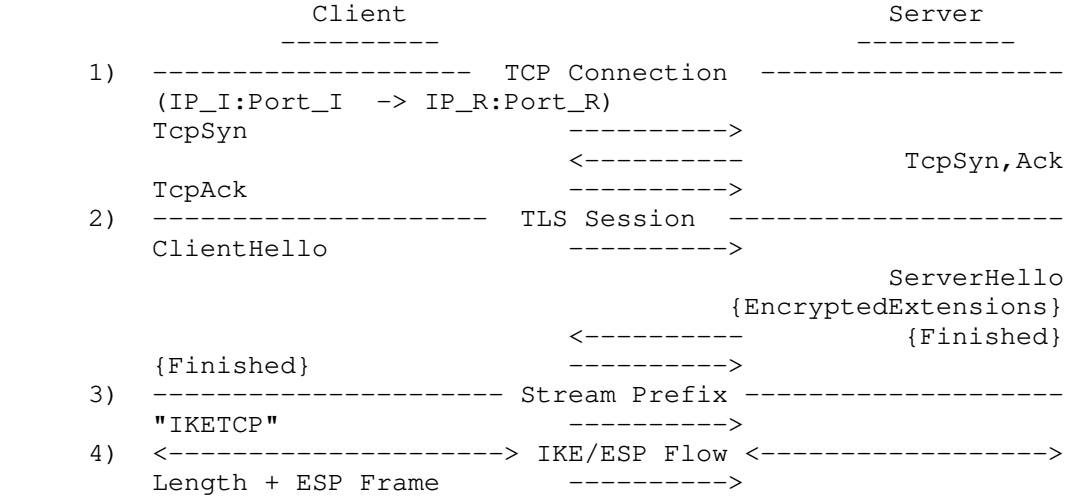


Figure 7

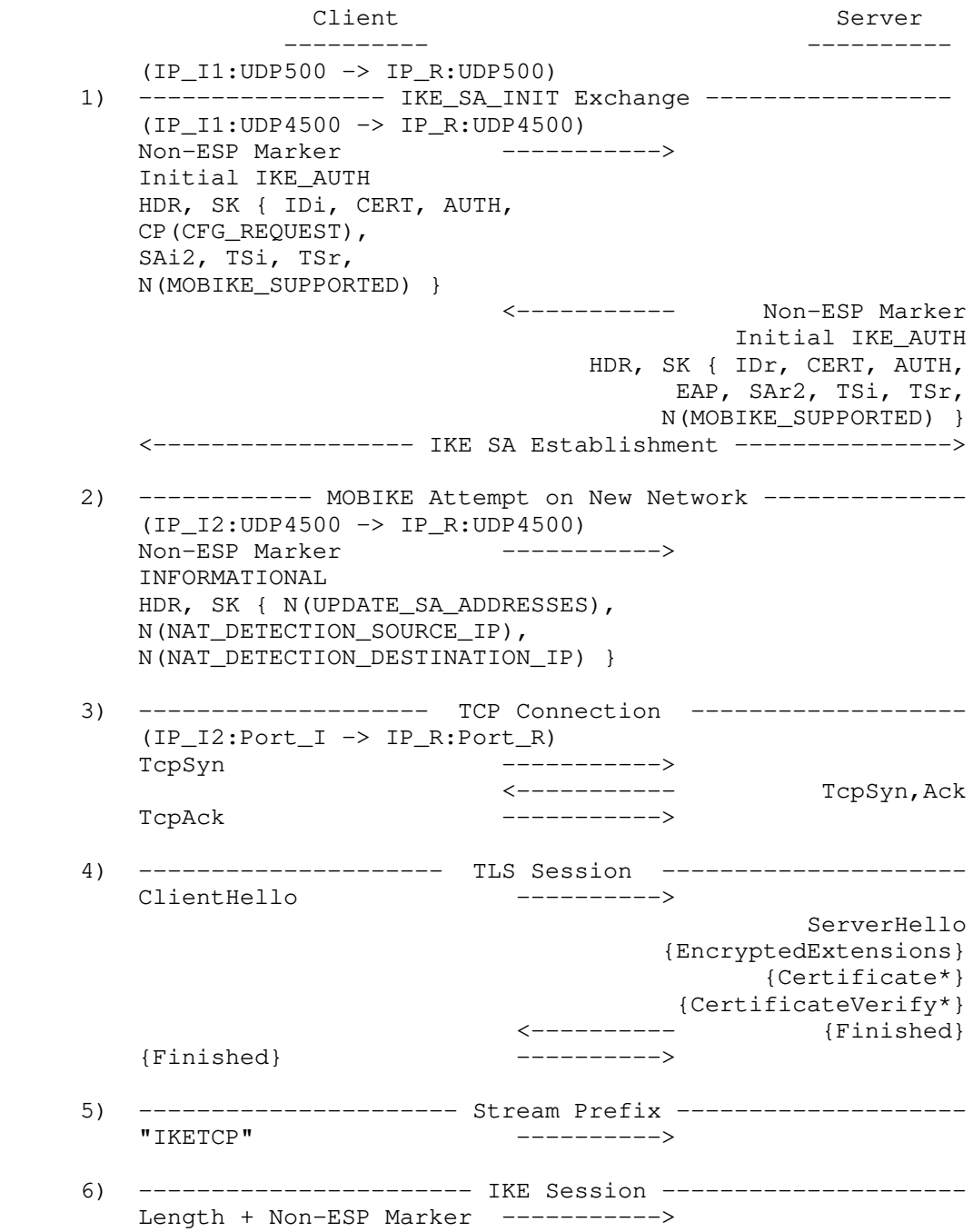
1. If a previous TCP connection was broken (for example, due to a TCP Reset), the client is responsible for re-initiating the TCP connection. The TCP Originator's address and port (IP_I and Port_I) may be different from the previous connection's address and port.

2. The client SHOULD attempt TLS session resumption if it has previously established a session with the server.

3. After TCP and TLS are complete, the client sends the stream prefix for TCP-encapsulated IKE traffic (Section 5).

4. The IKE and ESP packet flow can resume. If MOBIKE is being used, the Initiator SHOULD send an UPDATE_SA_ADDRESSES message.

B.4. Using MOBIKE between UDP and TCP Encapsulation



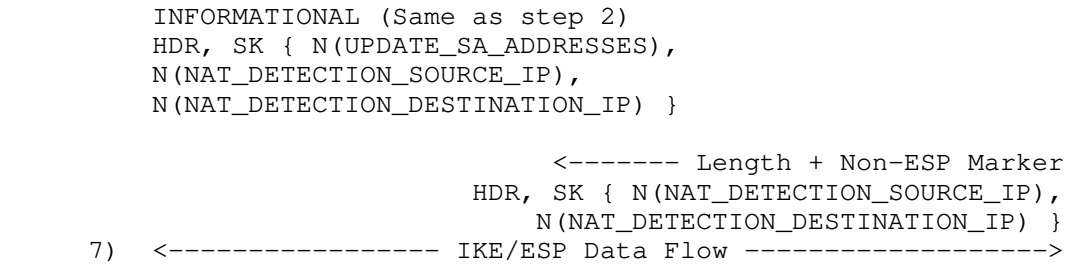


Figure 8

1. During the IKE_SA_INIT exchange, the client and server exchange MOBIKE_SUPPORTED notify payloads to indicate support for MOBIKE.

2. The client changes its point of attachment to the network and receives a new IP address. The client attempts to re-establish the IKE session using the UPDATE_SA_ADDRESSES notify payload, but the server does not respond because the network blocks UDP traffic.

3. The client brings up a TCP connection to the server in order to use TCP encapsulation.

4. The client initiates a TLS handshake with the server.

5. The client sends the stream prefix for TCP-encapsulated IKE traffic (Section 5).

6. The client sends the UPDATE_SA_ADDRESSES notify payload on the TCP-encapsulated connection. Note that this IKE message is the same as the one sent over UDP in step 2; it should have the same message ID and contents.

7. The IKE and ESP packet flow can resume.

Acknowledgments

The following people provided valuable feedback and advices while preparing RFC8229: Stuart Cheshire, Delziel Fernandes, Yoav Nir, Christoph Paasch, Yaron Sheffer, David Schinazi, Graham Bartlett, Byju Pularikkal, March Wu, Kingwel Xie, Valery Smyslov, Jun Hu, and Tero Kivinen. Special thanks to Eric Kinnear for his implementation work.

The authors would like to thank Tero Kivinen for his valuable comments while preparing this document.

Authors' Addresses

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

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

Tommy Pauly
Apple Inc.
1 Infinite Loop
Cupertino, California 95014
United States of America

Email: tpauly@apple.com