Workgroup:
Light-Weight Implementation Guidance (lwig)
Internet-Draft: draft-ietf-lwig-minimal-esp-09
Published: 6 April 2022
Intended Status: Informational
Expires: 8 October 2022
Authors: D.M. Migault T.G. Guggemos
Ericsson LMU Munich

Minimal IP Encapsulating Security Payload (ESP)

Abstract

This document describes the minimal properties IP Encapsulating Security Payload (ESP) implementation needs to meet to remain interoperable with the standard RFC4303 ESP. Such a minimal version of ESP is not intended to become a replacement of the RFC 4303 ESP. Instead, a minimal implementation is expected to be optimized for constrained environment while remaining interoperable with implementations of RFC 4303 ESP. In addition, this document also provides some considerations to implement minimal ESP in a constrained environment which includes limiting the number of flash writes, handling frequent wakeup / sleep states, limiting wakeup time, or reducing the use of random generation.

This document does not update or modify RFC 4303, but provides a compact description of how to implement the minimal version of the protocol. RFC 4303 remains the authoritative description.

Status of This Memo

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

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

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

This Internet-Draft will expire on 8 October 2022.

Copyright Notice

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

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

Table of Contents

1. Introduction
2. Security Parameter Index (SPI) (32 bit)
  2.1. Considerations over SPI generation
3. Sequence Number(SN) (32 bit)
4. Padding
5. Next Header (8 bit)
6. ICV
7. Cryptographic Suites
8. IANA Considerations
9. Security Considerations
10. Acknowledgment
11. References
  11.1. Normative References
  11.2. Informative References
Authors' Addresses

1. Introduction

ESP [RFC4303] is part of the IPsec protocol suite [RFC4301]. IPsec is used to provide confidentiality, data origin authentication, connectionless integrity, an anti-replay service (a form of partial sequence integrity) and limited traffic flow confidentiality (TFC) padding.

Figure 1 describes an ESP Packet. Currently, ESP is implemented in the kernel of major multipurpose Operating Systems (OS). The ESP and IPsec suite is usually implemented in a complete way to fit multiple purpose usage of these OSes. However, completeness of the IPsec suite as well as multipurpose scope of these OSes is often performed at the expense of resources, or performance. As a result, constrained devices are likely to have their own implementation of ESP optimized and adapted to their specificities such as limiting the number of flash writes (for each packet or across wake time), handling frequent wakeup and sleep state, limiting wakeup time, or reducing the use of random generation. With the adoption of IPsec by IoT devices with minimal IKEv2 [RFC7815] and ESP Header Compression (EHC) with [I-D.mglt-ipsecme-diet-esp] or [I-D.mglt-

ipsecme-ikev2-diet-esp-extension], it becomes crucial that ESP implementation designed for constrained devices remain inter-operable with the standard ESP implementation to avoid a fragmented usage of ESP. This document describes the minimal properties an ESP implementation needs to meet to remain interoperable with [RFC4303] ESP. In addition, this document also provides a set of options to implement these properties under certain constrained environments. This document does not update or modify RFC 4303, but provides a compact description of how to implement the minimal version of the protocol. RFC 4303 remains the authoritative description.

For each field of the ESP packet represented in Figure 1 this document provides recommendations and guidance for minimal implementations. The primary purpose of Minimal ESP is to remain interoperable with other nodes implementing RFC 4303 ESP, while limiting the standard complexity of the implementation.

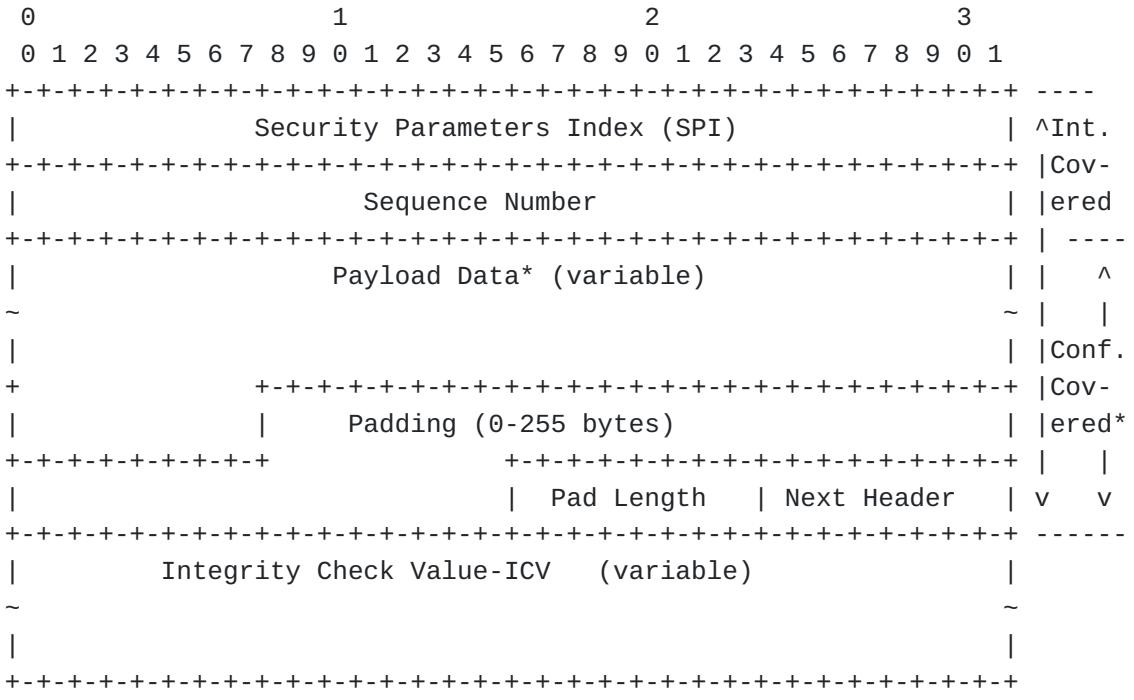


Figure 1: ESP Packet Description

2. Security Parameter Index (SPI) (32 bit)

According to the [RFC4303], the SPI is a mandatory 32 bits field and is not allowed to be removed.

The SPI has a local significance to index the Security Association (SA). From [RFC4301] section 4.1, nodes supporting only unicast communications can index their SA only using the SPI. On the other hand, nodes supporting multicast communications must also use the IP

addresses and thus SA lookup needs to be performed using the longest match.

For nodes supporting only unicast communications, this document recommends indexing SA with the SPI only. The index may be based on the full 32 bits of SPI or a subset of these bits. The node may require a combination of the SPI as well as other parameters (like the IP address) to index the SA.

Values 0-255 must not be used. As per section 2.1 of [RFC4303], values 1-255 are reserved and 0 is only allowed to be used internally and it must not be sent on the wire.

[RFC4303] does not require the SPI to be randomly generated over 32 bits. However, this is the recommended way to generate SPIs as it provides some privacy benefits and avoids, for example, correlation between ESP communications. To randomly generate a 32 bit SPI, the node generates a random 32 bit value and checks it does not fall in the 0-255 range. If the SPI has an acceptable value, it is used to index the inbound session, otherwise the SPI is re-generated until an acceptable value is found.

However, some constrained nodes may be less concerned by the privacy properties associated to SPIs randomly generated. Examples of such nodes might include sensors looking to reduce their code complexity, in which case the use of a predictive function to generate the SPI might be preferred over the generation and handling of random values. An example of such predictable function may consider the combination of a fixed value and the memory address of the SAD structure. For every incoming packet, the node will be able to point the SAD structure directly from the SPI value. This avoids having a separate and additional binding between SPI and SAD entries that is involved for every incoming packet.

2.1. Considerations over SPI generation

SPI that are not randomly generated over 32 bits may lead to privacy and security concerns. As a result, the use of alternative designs requires careful security and privacy reviews. This section provides some considerations upon the adoption of alternative designs.

Note that SPI value is used only for inbound traffic, as such the SPI negotiated with IKEv2 [RFC7296] or [RFC7815] by a peer, is the value used by the remote peer when it sends traffic. As SPI is only used for inbound traffic by the peer, this allows each peer to manage the set of SPIs used for its inbound traffic. Similarly, the privacy concerns associated with the generation of non random SPI is also limited to the incoming traffic.

When alternate designs are considered, it is likely that the number of possible SPIs will be limited. This limit should both consider the number of inbound SAs - possibly per IP addresses - as well as the ability for the node to rekey. SPI can typically be used to implement a key update with the SPI indicating the key is being used. For example, a SPI might be encoded with the Security Association Database (SAD) entry on a subset of bytes (for example 3 bytes), while the remaining byte indicates the rekey index.

The use of a smaller number of SPIs across communications comes with privacy and security concerns. Typically, some specific values or subset of SPI values may reveal the models or manufacturer of the node implementing ESP. This may raise some privacy issues as an observer is likely to be able to determine the constrained devices of the network. In some cases, these nodes may host a very limited number of applications - typically a single application - in which case the SPI would provide some information related to the application of the user. In addition, the device or application may be associated with some vulnerabilities, in which case specific SPI values may be used by an attacker to discover vulnerabilities.

While the use of randomly generated SPIs may reduce the leakage or privacy of security related information by ESP itself, these pieces of information may also be leaked otherwise. As a result, a privacy analysis should consider at least the type of information as well the traffic pattern before determining a non random SPI can be used. Typically, temperature sensors, wind sensors, used outdoors may not leak privacy sensitive information and most of its traffic is expected to be outbound traffic. When used indoors, a sensor that reports every minute an encrypted status of the door (closed or opened) may leak truly little privacy sensitive information outside the local network. In both cases, if the state of the sensor doesn't leak privacy info, a randomized SPI is not necessary.

3. Sequence Number(SN) (32 bit)

According to [RFC4303], the Sequence Number (SN) is a mandatory 32 bits field in the packet.

The SN is set by the sender so the receiver can implement anti-replay protection. The SN is derived from any strictly increasing function that guarantees: if packet B is sent after packet A, then SN of packet B is strictly greater than the SN of packet A.

Some constrained devices may establish communication with specific devices, like a specific gateway, or nodes similar to them. As a result, the sender may know whereas the receiver implements anti-replay protection or not. Even though the sender may know the

receiver does not implement anti-replay protection, the sender must implement an always increasing function to generate the SN.

Usually, SN is generated by incrementing a counter for each packet sent. A constrained device may avoid maintaining this context and use another source that is known to always increase. Typically, constrained nodes using 802.15.4 Time Slotted Channel Hopping (TSCH), whose communication is heavily dependent on time, can take advantage of their clock to generate the SN. A lot of IoT devices are in a sleep state most of the time wake up and are only awake to perform a specific operation before going back to sleep. They do have separate hardware that allows them to wake up after a certain timeout, and most likely also timers that start running when the device was booted up, so they might have a concept of time with certain granularity. This requires to store any information in a stable storage - such as flash memory - that can be restored across sleeps. Storing information associated with the SA such as SN requires some read and writing operation on a stable storage after each packet is sent as opposed to SPI or keys that are only written at the creation of the SA. Such operations are likely to wear out the flash, and slow down the system greatly, as writing to flash is not as fast as reading. Their internal clocks/timers might not be very accurate, but they should be enough to know that each time they wake up their time is greater than what it was last time they woke up. Using time for SN would guarantee a strictly increasing function and avoid storing any additional values or context related to the SN. Of course, one should only consider use of a clock to generate SNs if the application will inherently ensure that no two packets with a given SA are sent with the same time value. Note however that standard receivers are generally configured with incrementing counters and, if not appropriately configured, the use of a significantly larger SN difference may result in the packet out of the receiver's windows and that packet being discarded.

For inbound traffic, this document recommends that any receiver provides anti-replay protection, and the size of the window depends on the ability of the network to deliver packets out of order. As a result, in an environment where out of order packets is not possible the window size can be set to one. However, while recommended, there are no requirements to implement an anti-replay protection mechanism implemented by IPsec. Similarly to the SN the implementation of anti replay protection may require the device to write the received SN for every packet, which may in some cases come with the same drawbacks as those exposed for SN. As a result, some implementations may drop a non required anti replay protection especially when the necessary resource involved overcomes the benefit of the mechanism. These resources need also to balance that absence of anti-replay mechanism, may lead to unnecessary integrity check operations that might be significantly more expensive as well. A typical example

might consider an IoT device such as a temperature sensor that is sending a temperature every 60 seconds, and that receives an acknowledgment from the receiver. In such cases, the ability to spoof and replay an acknowledgement is of limited interest and might not justify the implementation of an anti replay mechanism. Receiving peers may also use ESP anti-replay mechanism adapted to a specific application. Typically, when the sending peer is using SN based on time, anti-replay may be implemented by discarding any packets that present a SN whose value is too much in the past. Note that such mechanisms may consider clock drifting in various ways in addition to acceptable delay induced by the network to avoid the anti replay windows rejecting legitimate packets. When a packet is received at a regular time interval, some variant of time based mechanisms may not even use the value of the SN, but instead only consider the receiving time of the packet.

SN can be encoded over 32 bits or 64 bits - known as Extended Sequence Number (ESN). As per [RFC4303], the support of ESN is not mandatory. The determination of the use of ESN is based on the largest possible value a SN can take over a session. When SN is incremented for each packet, the number of packets sent over the lifetime of a session may be considered. However, when the SN is incremented differently - such as when time is used - the maximum value SN needs to be considered instead. Note that the limit of messages being sent is primarily determined by the security associated with the key rather than the SN. The security of all data protected under a given key decreases slightly with each message and a node must ensure the limit is not reached - even though the SN would permit it. Estimation of the maximum number of packets to be sent by a node is always challenging and as such should be considered cautiously as nodes could be online for much more time than expected. Even for constrained devices, this document recommends implementing some rekey mechanisms (see Section 9).

4. Padding

The purpose of padding is to respect the 32 bit alignment of ESP or block size expected by an encryption transform - such as AES-CBC for example. ESP must have at least one padding byte Pad Length that indicates the padding length. ESP padding bytes are generated by a succession of unsigned bytes starting with 1, 2, 3 with the last byte set to Pad Length, where Pad Length designates the length of the padding bytes.

Checking the padding structure is not mandatory, so the constrained device may omit such checks, however, in order to interoperate with existing ESP implementations, it must build the padding bytes as recommended by ESP.

In some situation the padding bytes may take a fixed value. This would typically be the case when the Data Payload is of fixed size.

ESP [RFC4303] also provides Traffic Flow Confidentiality (TFC) as a way to perform padding to hide traffic characteristics, which differs from respecting a 32 bit alignment. TFC is not mandatory and must be negotiated with the SA management protocol. TFC has not yet being widely adopted for standard ESP traffic. One possible reason is that it requires to shape the traffic according to one traffic pattern that needs to be maintained. This is likely to require extra processing as well as providing a "well recognized" traffic shape which could end up being counterproductive. As such, this document does not recommend that minimal ESP implementation supports TFC.

As a result, TFC cannot be enabled with minimal ESP, and communication protection that were relying on TFC will be more sensitive to traffic shaping. This could expose the application as well as the devices used to a passive monitoring attacker. Such information could be used by the attacker in case a vulnerability is disclosed on the specific device. In addition, some application use - such as health applications - may also reveal important privacy oriented information.

Some constrained nodes that have limited battery lifetime may also prefer avoiding sending extra padding bytes. However, the same nodes may also be very specific to an application and device. As a result, they are also likely to be the main target for traffic shaping. In most cases, the payload carried by these nodes is quite small, and the standard padding mechanism may also be used as an alternative to TFC, with a sufficient tradeoff between the required energy to send additional payload and the exposure to traffic shaping attacks. In addition, the information leaked by the traffic shaping may also be addressed by the application level. For example, it is preferred to have a sensor sending some information at regular time interval, rather than when a specific event is happening. Typically, a sensor monitoring the temperature, or a door is expected to send regularly the information - i.e. the temperature of the room or whether the door is closed or open) instead of only sending the information when the temperature has raised or when the door is being opened.

5. Next Header (8 bit)

According to [RFC4303], the Next Header is a mandatory 8 bits field in the packet. Next header specifies the data contained in the payload as well as dummy packet, i.e. packets with the Next Header with a value 59 meaning "no next header". In addition, the Next Header may also carry an indication on how to process the packet [I-D.nikander-esp-beet-mode].

The ability to generate and receive dummy packets is required by [RFC4303]. For interoperability, a minimal ESP implementation must discard dummy packets without indicating an error. Note that such recommendation only applies for nodes receiving packets, and that nodes designed to only send data might not implement this capability.

As the generation of dummy packets is subject to local management and based on a per-SA basis, a minimal ESP implementation may not generate such dummy packet. More especially, in constrained environment sending dummy packets may have too much impact on the device lifetime, and so may be avoided. On the other hand, constrained nodes may be dedicated to specific applications, in which case, traffic pattern may expose the application or the type of node. For these nodes, not sending dummy packet may have some privacy implication that needs to be measured. However, for the same reasons exposed in Section 4 traffic shaping at the IPsec layer may also introduce some traffic pattern, and on constrained devices the application is probably the most appropriated layer to limit the risk of leaking information by traffic shaping.

In some cases, devices are dedicated to a single application or a single transport protocol, in which case, the Next Header has a fixed value.

Specific processing indications have not been standardized yet [I-D.nikander-esp-beet-mode] and is expected to result from an agreement between the peers. As a result, it should not be part of a minimal implementation of ESP.

6. ICV

The ICV depends on the cryptographic suite used. Currently, [RFC8221] only recommends cryptographic suites with an ICV which makes the ICV a mandatory field.

As detailed in [RFC8221] authentication or authenticated encryption are recommended and as such the ICV field must be present with a size different from zero. Its length is defined by the security recommendations only.

7. Cryptographic Suites

The cryptographic suites implemented are an important component of ESP. The recommended algorithms to use are expected to evolve over time and implementers should follow the recommendations provided by [RFC8221] and updates.

This section lists some of the criteria that may be considered. The list is not expected to be exhaustive and may also evolve overtime. As a result, the list is provided as informational:

1. Security: Security is the criteria that should be considered first for the selection of encryption algorithm transform. The security of encryption algorithm transforms is expected to evolve over time, and it is of primary importance to follow up-to-date security guidance and recommendations. The chosen encryption algorithm must not be known vulnerable or weak (see [RFC8221] for outdated ciphers). ESP can be used to authenticate only or to encrypt the communication. In the latter case, authenticated encryption must always be considered [RFC8221].

2. Resilience to nonce re-use: Some transforms -including AES-GCM - are very sensitive to nonce collision with a given key. While the generation of the nonce may prevent such collision during a session, the mechanisms are unlikely to provide such protection across reboot. This causes an issue for devices that are configured with a key. When the key is likely to be re-used across reboots, this document recommends considering algorithms that are nonce misuse resistant such as, for example, AES-SIV [RFC5297], AES-GCM-SIV [RFC8452] or Deoxys-II [DeoxysII]. Note however that currently none of them has yet been defined for ESP.

3. Interoperability: Interoperability considers the encryption algorithm transforms shared with the other nodes. Note that it is not because an encryption algorithm transform is widely deployed that it is secured. As a result, security should not be weakened for interoperability. [RFC8221] and successors consider the life cycle of encryption algorithm transforms sufficiently long to provide interoperability. Constrained devices may have limited interoperability requirements which makes possible to reduces the number of encryption algorithm transforms to implement.

4. Power Consumption and Cipher Suite Complexity: Complexity of the encryption algorithm transform or the energy associated with it are especially considered when devices have limited resources or are using some batteries, in which case the battery determines the life of the device. The choice of a cryptographic function may consider re-using specific libraries or to take advantage of hardware acceleration provided by the device. For example, if the device benefits from AES hardware modules and uses AES-CTR, it may prefer AUTH_AES-XCBC for its authentication. In addition, some devices may also embed radio

modules with hardware acceleration for AES-CCM, in which case, this mode may be preferred.

5. Power Consumption and Bandwidth Consumption: Similarly to the encryption algorithm transform complexity, reducing the payload sent, may significantly reduce the energy consumption of the device. As a result, encryption algorithm transforms with low overhead may be considered. To reduce the overall payload size one may, for example:

    1. Use of counter-based ciphers without fixed block length (e.g. AES-CTR, or ChaCha20-Poly1305).

    2. Use of ciphers with capability of using implicit IVs [RFC8750].

    3. Use of ciphers recommended for IoT [RFC8221].

    4. Avoid Padding by sending payload data which are aligned to the cipher block length - 2 for the ESP trailer.

8. IANA Considerations

There are no IANA consideration for this document.

9. Security Considerations

Security considerations are those of [RFC4303]. In addition, this document provided security recommendations and guidance over the implementation choices for each field.

The security of a communication provided by ESP is closely related to the security associated with the management of that key. This usually includes mechanisms to prevent a nonce from repeating, for example. When a node is provisioned with a session key that is used across reboot, the implementer must ensure that the mechanisms put in place remain valid across reboot as well.

This document recommends to use ESP in conjunction with key management protocols such as for example IKEv2 [RFC7296] or minimal IKEv2 [RFC7815]. Such mechanisms are responsible for negotiating fresh session keys as well as prevent a session key being use beyond its lifetime. When such mechanisms cannot be implemented and the session key is, for example, provisioned, the nodes must ensure that keys are not used beyond their lifetime and that the appropriate use of the key remains across reboots - e.g. conditions on counters and nonces remains valid.

When a node generates its key or when random value such as nonces are generated, the random generation must follow [RFC4086]. In

addition, [SP-800-90A-Rev-1] provides appropriated guidance to build random generators based on deterministic random functions.

10. Acknowledgment

The authors would like to thank Daniel Palomares, Scott Fluhrer, Tero Kivinen, Valery Smyslov, Yoav Nir, Michael Richardson, Thomas Peyrin, Eric Thormarker, Nancy Cam-Winget and Bob Briscoe for their valuable comments. In particular Scott Fluhrer suggested including the rekey index in the SPI. Tero Kivinen provided also multiple clarifications and examples of deployment ESP within constrained devices with their associated optimizations. Thomas Peyrin Eric Thormarker and Scott Fluhrer suggested and clarified the use of transform resilient to nonce misuse.

11. References

11.1. 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>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/info/rfc4086>.

[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>.

[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>.

[RFC7815] Kivinen, T., "Minimal Internet Key Exchange Version 2 (IKEv2) Initiator Implementation", RFC 7815, DOI 10.17487/RFC7815, March 2016, <https://www.rfc-editor.org/info/rfc7815>.

[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>.

[RFC8221] Wouters, P., Migault, D., Mattsson, J., Nir, Y., and T. Kivinen, "Cryptographic Algorithm Implementation Requirements and Usage Guidance for Encapsulating Security Payload (ESP) and Authentication Header (AH)", RFC 8221, DOI 10.17487/RFC8221, October 2017, <https://www.rfc-editor.org/info/rfc8221>.

[RFC8750] Migault, D., Guggemos, T., and Y. Nir, "Implicit Initialization Vector (IV) for Counter-Based Ciphers in Encapsulating Security Payload (ESP)", RFC 8750, DOI 10.17487/RFC8750, March 2020, <https://www.rfc-editor.org/info/rfc8750>.

11.2. Informative References

[DeoxysII] Jeremy, J. J., Ivica, I. N., Thomas, T. P., and Y. S. Yannick, "Deoxys v1.41", October 2016, <https://competitions.cr.yp.to/round3/deoxysv141.pdf>.

[I-D.mglt-ipsecme-diet-esp] Migault, D., Guggemos, T., Bormann, C., and D. Schinazi, "ESP Header Compression and Diet-ESP", Work in Progress, Internet-Draft, draft-mglt-ipsecme-diet-esp-07, 11 March 2019, <https://www.ietf.org/archive/id/draft-mglt-ipsecme-diet-esp-07.txt>.

[I-D.mglt-ipsecme-ikev2-diet-esp-extension] Migault, D., Guggemos, T., and D. Schinazi, "Internet Key Exchange version 2 (IKEv2) extension for the ESP Header Compression (EHC) Strategy", Work in Progress, Internet-Draft, draft-mglt-ipsecme-ikev2-diet-esp-extension-01, 26 June 2018, <https://www.ietf.org/archive/id/draft-mglt-ipsecme-ikev2-diet-esp-extension-01.txt>.

[I-D.nikander-esp-beet-mode] Nikander, P. and J. Melen, "A Bound End-to-End Tunnel (BEET) mode for ESP", Work in Progress, Internet-Draft, draft-nikander-esp-beet-mode-09, 5 August 2008, <https://www.ietf.org/archive/id/draft-nikander-esp-beet-mode-09.txt>.

[RFC5297] Harkins, D., "Synthetic Initialization Vector (SIV) Authenticated Encryption Using the Advanced Encryption Standard (AES)", RFC 5297, DOI 10.17487/RFC5297, October 2008, <https://www.rfc-editor.org/info/rfc5297>.

[RFC8452] Gueron, S., Langley, A., and Y. Lindell, "AES-GCM-SIV: Nonce Misuse-Resistant Authenticated Encryption", RFC 8452, DOI 10.17487/RFC8452, April 2019, <https://www.rfc-editor.org/info/rfc8452>.

[SP-800-90A-Rev-1]
Elain, E. B. and J. K. Kelsey, "Recommendation for Random Number Generation Using Deterministic Random Bit Generators", <https://csrc.nist.gov/publications/detail/sp/800-90a/rev-1/final>.

Authors' Addresses

Daniel Migault
Ericsson
8400 boulevard Decarie
Montreal, QC H4P 2N2
Canada

Email: daniel.migault@ericsson.com

Tobias Guggemos
LMU Munich
MNM-Team
Oettingenstr. 67
80538 Munich
Germany

Email: guggemos@mnm-team.org