Stream: Internet Engineering Task Force (IETF)
RFC: 9333
Category: Informational
Published: January 2023
ISSN: 2070-1721
Authors: D. Migault *Ericsson* T. Guggemos *LMU Munich*

RFC 9333
Minimal IP Encapsulating Security Payload (ESP)

Abstract

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

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

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are candidates for any level of Internet Standard; see Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9333.

Copyright Notice

Table of Contents

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, and limited Traffic Flow Confidentiality (TFC) padding.

Figure 1 describes an ESP packet. Currently, ESP is implemented in the kernel of most major multipurpose Operating Systems (OSes). ESP is usually implemented with all of its features to fit the multipurpose usage of these OSes, at the expense of resources and with no considerations for code size. Constrained devices are likely to have their own implementation of ESP optimized and adapted to their specific use, such as limiting the number of flash writes (for each packet or across wake time), handling frequent wakeup and sleep states, limiting wakeup time, and reducing the use of random generation. With the adoption of IPsec by Internet of Things (IoT) devices with minimal IKEv2 [RFC7815] and ESP Header Compression (EHC) [EHC-DIET-ESP] [EHC-IKEv2], these ESP implementations **MUST** remain interoperable with standard ESP implementations. This document describes the minimal properties an ESP implementation needs to meet to remain interoperable with ESP [RFC4303]. In addition, this document provides advice to implementers for implementing ESP within constrained environments. This document does not update or modify [RFC4303].

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 ESP [RFC4303], while limiting the standard complexity of the implementation.

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ----
 |               Security Parameters Index (SPI)                 | ^Int.
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |Cov-
 |                      Sequence Number                          | |ered
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ | ----
 |                    Payload Data* (variable)                   | |   ^
 ~                                                               ~ |   |
 |                                                               | |Conf.
 +               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |Cov-
 |               |     Padding (0-255 bytes)                     | |ered*
 +-+-+-+-+-+-+-+-+               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |   |
 |                               |  Pad Length   | Next Header   | v   v
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
 |         Integrity Check Value (ICV) (variable)                |
 ~                                                               ~
 |                                                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: ESP Packet Description

2. Requirements 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. Security Parameters Index (SPI)

[RFC4303] defines the SPI as a mandatory 32-bit field.

The SPI has local significance to index the Security Association (SA). As described in Section 4.1 of [RFC4301], nodes supporting only unicast communications can index their SA using only the SPI. Nodes supporting multicast communications also require the use of IP addresses; thus, SA lookup needs to be performed using the longest match.

For nodes supporting only unicast communications, indexing the SA using only the SPI is **RECOMMENDED**. The index may be based on the full 32 bits of the 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 **MUST NOT** be sent over the wire.

[RFC4303] does not require the 32-bit SPI to be randomly generated, although that is the **RECOMMENDED** way to generate SPIs as it provides some privacy and security benefits and avoids correlation between ESP communications. To obtain a usable random 32-bit SPI, the node generates a random 32-bit value and checks it does not fall within the 0-255 range. If the SPI has an acceptable value, it is used to index the inbound session. Otherwise, the generated value is discarded, and the process repeats until a valid value is found.

Some constrained devices are less concerned with the privacy properties associated with randomly generated SPIs. Examples of such devices might include sensors looking to reduce their code complexity. The use of a predictive function to generate the SPI might be preferred over the generation and handling of random values. An implementation of such predictable function could use the combination of a fixed value and the memory address of the Security Association Database (SAD) structure. For every incoming packet, the node will be able to point to the SAD structure directly from the SPI value. This avoids having a separate and additional binding and lookup function for the SPI to its SAD entry for every incoming packet.

3.1. Considerations for SPI Generation

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

The SPI value is only looked up for inbound traffic. The SPI negotiated with IKEv2 [RFC7296] or minimal IKEv2 [RFC7815] by a peer is the value used by the remote peer when it sends traffic. The main advantage of using a rekeying mechanism is to enable a rekey, which is performed by replacing an old SA with a new SA, both indexed with distinct SPIs. The SPI is only used for inbound traffic by the peer, which allows each peer to manage the set of SPIs used for its inbound traffic. The necessary number of SPIs reflects the number of inbound SAs as well as the ability to rekey those SAs. Typically, rekeying an SA is performed by creating a new SA (with a dedicated SPI) before the old SA is deleted. This results in an additional SA and the need to support an additional SPI. Similarly, the privacy concerns associated with the generation of non-random SPIs is also limited to the incoming traffic.

Alternatively, some constrained devices will not implement IKEv2 or minimal IKEv2 and, as such, will not be able to manage a rollover between two distinct SAs. In addition, some of these constrained devices are likely to have a limited number of SAs; for example, they are likely to be indexed over 3 bytes only. One possible way to enable a rekeying mechanism with these devices is to use the SPI where, for example, the first 3 bytes designates the SA while the remaining byte indicates a rekey index. SPI numbers can be used to implement tracking the inbound SAs when rekeying is taking place. When rekeying an SPI, the new SPI could use the SPI bytes to indicate the rekeying index.

The use of a small, limited set of SPI numbers across communications comes with privacy and security concerns. Some specific values or subsets of SPI values could reveal the model or manufacturer of the node implementing ESP or reveal a state such as "not yet rekeyed" or "rekeyed 10 times". If a constrained host uses a very limited number of applications, eventually a single one, the SPI itself could indicate what kind of traffic is transmitted (e.g., the kind of application typically running). This could also be correlated with encrypted data size to further leak information to an observer on the network. In addition, use of specific hardcoded SPI numbers could reveal a manufacturer or device version. If updated devices use different SPI numbers, an attacker could locate vulnerable devices by their use of specific SPI numbers.

A privacy analysis should consider at least the type of information as well as the traffic pattern before deciding whether non-random SPIs are safe to use. Typically, temperature and wind sensors that are used outdoors do not leak privacy-sensitive information, and most of their traffic is expected to be outbound traffic. When used indoors, a sensor that reports an encrypted status of a door (closed or opened) every minute might not leak sensitive information outside the local network. In these examples, the privacy aspect of the information itself might be limited. Being able to determine the version of the sensor to potentially take control of it may also have some limited security consequences. Of course, this depends on the context in which these sensors are being used. If the risks associated to privacy and security are acceptable, a non-randomized SPI can be used.

4. Sequence Number (SN)

The Sequence Number (SN) in [RFC4303] is a mandatory 32-bit 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 the following: if packet B is sent after packet A, then the SN of packet B is higher than the SN of packet A.

Some constrained devices may establish communication with specific devices where it is known whether or not the peer implements anti-replay protection. As per [RFC4303], the sender **MUST** still implement a strictly increasing function to generate the SN.

It is **RECOMMENDED** that multipurpose ESP implementations increment a counter for each packet sent. However, a constrained device may avoid maintaining this context and use another source that is known to always increase. Typically, constrained devices use 802.15.4 Time Slotted Channel Hopping (TSCH). This communication is heavily dependent on time. A constrained device can take advantage of this clock mechanism to generate the SN. A lot of IoT devices are in a sleep state most of the time and wake up only to perform a specific operation before going back to sleep. These devices have separate hardware that allows them to wake up after a certain timeout and typically also have timers that start running when the device is booted up, so they might have a concept of time with certain granularity. This requires devices to store any information in stable storage that can be restored across sleeps (e.g., flash memory). Storing information associated with the SA (such as the SN) requires some read and write operations on stable storage after each packet is sent as opposed to an SPI number or cryptographic keys that are only written to stable storage at the creation of the SA. Write operations wear out the flash storage. Write operations also slow down the system significantly, as writing to flash is much slower than reading from flash. While these devices have internal clocks or timers that might not be very accurate, they are good enough to guarantee that each time the device wakes up from sleep, the time is greater than what it was before the device went to sleep. Using time for the SN would guarantee a strictly increasing function and avoid storing any additional values or context related to the SN on flash. In addition to the time value, a RAM-based counter can be used to ensure that the serial numbers are still increasing and unique if the device sends multiple packets over an SA within one wakeup period.

For inbound traffic, it is **RECOMMENDED** that receivers implement anti-replay protection. The size of the window should depend on the network characteristic to deliver packets out of order. In an environment where out-of-order packets are not possible, the window size can be set to one. An ESP implementation may choose to not implement anti-replay protection. An implementation of anti-replay protection may require the device to write the received SN for every packet to stable storage. This will have the same issues as discussed earlier with the SN. Some constrained device implementations may choose to not implement the optional anti-replay protection. A typical example is an IoT device such as a temperature sensor that sends a temperature measurement every 60 seconds and receives an acknowledgment from the receiver. In a case like this, 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 an ESP anti-replay mechanism adapted to a specific application. Typically, when the sending peer is using an SN based on time, anti-replay may be implemented by discarding any packets that present an SN whose value is too much in the past. 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. Receiving peers could accept any SN as long as it is higher than the previously received SN. Another mechanism could be used where only the received time on the device is used to consider a packet to be valid, without looking at the SN at all.

The SN can be represented as a 32-bit number or as a 64-bit number, known as an "Extended Sequence Number (ESN)". As per [RFC4303], support of ESN is not mandatory, and its use is negotiated via IKEv2 [RFC7296]. An ESN is used for high-speed links to ensure there can be more than 2^{32} packets before the SA needs to be rekeyed to prevent the SN from rolling over. This assumes the SN is incremented by 1 for each packet. When the SN is incremented differently --

such as when time is used -- rekeying needs to happen based on how the SN is incremented to prevent the SN from rolling over. 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 not always predictable, and large margins should be used, especially as nodes could be online for much more time than expected. Even for constrained devices, it is **RECOMMENDED** to implement some rekeying mechanisms (see Section 10).

5. Padding

Padding is required to keep the 32-bit alignment of ESP. It is also required for some encryption transforms that need a specific block size of input, such as ENCR_AES_CBC. ESP specifies padding in the Pad Length byte, followed by up to 255 bytes of padding.

Checking the padding structure is not mandatory, so constrained devices may omit these checks on received ESP packets. For outgoing ESP packets, padding must be applied as required by ESP.

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

ESP [RFC4303] additionally provides Traffic Flow Confidentiality (TFC) as a way to perform padding to hide traffic characteristics. TFC is not mandatory and is negotiated with the SA management protocol, such as IKEv2. TFC has been widely implemented, but it is not widely deployed for ESP traffic. It is **NOT RECOMMENDED** to implement TFC for minimal ESP.

As a consequence, communication protection that relies on TFC would be more sensitive to traffic patterns without TFC. This can leak application information as well as the manufacturer or model of the device used to a passive monitoring attacker. Such information can be used, for example, by an attacker if a vulnerability is known for the specific device or application. In addition, some applications (such as health applications) could leak important privacy-oriented information.

Constrained devices that have a limited battery lifetime may prefer to avoid sending extra padding bytes. In most cases, the payload carried by these devices is quite small, and the standard padding mechanism can be used as an alternative to TFC. Alternatively, any information leak based on the size -- or presence -- of the packet can also be addressed at the application level before the packet is encrypted with ESP. If application packets vary between 1 to 30 bytes, the application could always send 32-byte responses to ensure all traffic sent is of identical length. To prevent leaking information that a sensor changed state, such as "temperature changed" or "door opened", an application could send this information at regular time intervals, rather than when a specific event is happening, even if the sensor state did not change.

6. Next Header and "Dummy" Packets

ESP [RFC4303] defines the Next Header as a mandatory 8-bit field in the packet. The Next Header, only visible after decryption, specifies the data contained in the payload. In addition, the Next Header may carry an indication on how to process the packet [BEET-ESP]. The Next Header can point to a "dummy" packet, i.e., a packet with the Next Header value set to 59, meaning "no next header". The data following "no next header" is unstructured "dummy" data. (Note that this document uses the term "dummy" for consistency with [RFC4303].)

The ability to generate, receive, and ignore "dummy" packets is required by [RFC4303]. An implementation can omit ever generating and sending "dummy" packets. For interoperability, a minimal ESP implementation **MUST** be able to process and discard "dummy" packets without indicating an error.

In constrained environments, sending "dummy" packets may have too much impact on the device lifetime, in which case, "dummy" packets should not be generated and sent. On the other hand, constrained devices running specific applications that would leak too much information by not generating and sending "dummy" packets may implement this functionality or even implement something similar at the application layer. Note also that similarly to padding and TFC that can be used to hide some traffic characteristics (see Section 5), "dummy" packets may also reveal some patterns that can be used to identify the application. For example, an application may send "dummy" data to hide a traffic pattern. Suppose such an application sends a 1-byte data when a change occurs. This results in sending a packet notifying a change has occurred. "Dummy" packets may be used to prevent such information from being leaked by sending a 1-byte packet every second when the information is not changed. After an upgrade, the data becomes 2 bytes. At that point, the "dummy" packets do not hide anything, and having 1 byte regularly versus 2 bytes makes even the identification of the application version easier to identify. This generally makes the use of "dummy" packets more appropriate on high-speed links.

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

Specific processing indications have not been standardized yet [BEET-ESP] and are expected to result from an agreement between the peers. As a result, they **SHOULD NOT** be part of a minimal implementation of ESP.

7. ICV

The ICV depends on the cryptographic suite used. As detailed in [RFC8221], authentication or authenticated encryption is **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.

8. Cryptographic Suites

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 to select an appropriate cryptographic suite. The list is not expected to be exhaustive and may also evolve over time.

1. Security: Security is the criteria that should be considered first for the selection of encryption algorithm transforms. 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 vulnerable or weak (see [RFC8221] for outdated ciphers). ESP can be used to authenticate only (ENCR_NULL) or to encrypt the communication. In the latter case, Authenticated Encryption with Associated Data (AEAD) is **RECOMMENDED** [RFC8221].
2. Resilience to Nonce Reuse: Some transforms, including AES-GCM, are vulnerable 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 sleep states or reboot. This causes an issue for devices that are configured using static keys (called "manual keying"), and manual keying should not be used with these encryption algorithms. When the key is likely to be reused across reboots, algorithms that are resistant to nonce misuse (for example, AES-SIV [RFC5297], AES-GCM-SIV [RFC8452], and Deoxys-II [DeoxysII]) are **RECOMMENDED**. Note, however, that none of these are currently defined for use with ESP.
3. Interoperability: Constrained devices usually only implement one or very few different encryption algorithm transforms. [RFC8221] takes the life cycle of encryption algorithm transforms and device manufacturing into consideration in its recommendations for mandatory-to-implement (MTI) algorithms.
4. Power Consumption and Cipher Suite Complexity: Complexity of the encryption algorithm transform and the energy cost associated with it are especially important considerations for devices that have limited resources or are battery powered. The battery life might determine the lifetime of the entire device. When choosing a cryptographic function, reusing specific libraries or taking advantage of hardware acceleration provided by the device should be considered. For example, if the device benefits from AES hardware modules and uses ENCR_AES_CTR, it may prefer AUTH_AES-XCBC for its authentication. In addition, some devices may embed radio modules with hardware acceleration for AES-CCM, in which case, this transform may be preferred.
5. Power Consumption and Bandwidth Consumption: Reducing the payload sent may significantly reduce the energy consumption of the device. Encryption algorithm transforms with low overhead are strongly preferred. To reduce the overall payload size, one may, for example:
    - Use counter-based ciphers without fixed block length (e.g., AES-CTR or ChaCha20-Poly1305).

- Use ciphers capable of using implicit Initialization Vectors (IVs) [RFC8750].
- Use ciphers recommended for IoT [RFC8221].
- Avoid padding by sending payload data that are aligned to the cipher block length -- 2 bytes for the ESP trailer.

9. IANA Considerations

This document has no IANA actions.

10. Security Considerations

The security considerations in [RFC4303] apply to this document as well. In addition, this document provides security recommendations and guidance for the implementation choices for each ESP 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.

It is **RECOMMENDED** 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 preventing a session key being used beyond its lifetime. When such mechanisms cannot be implemented, such as when the session key is provisioned, the device **MUST** ensure that keys are not used beyond their lifetime and that the key remains used in compliance with all security requirements across reboots (e.g., conditions on counters and nonces remain valid).

When a device generates its own key or when random values such as nonces are generated, the random generation **MUST** follow [RFC4086]. In addition, [SP-800-90A-Rev-1] provides guidance on how to build random generators based on deterministic random functions.

11. Privacy Considerations

Preventing the leakage of privacy-sensitive information is a hard problem to solve and usually results in balancing the information potentially being leaked to the cost associated with the counter measures. This problem is not inherent to the minimal ESP described in this document and also concerns the use of ESP in general.

This document targets minimal implementations of ESP and, as such, describes a minimalistic way to implement ESP. In some cases, this may result in potentially revealing privacy-sensitive pieces of information. This document describes these privacy implications so the implementer can make the appropriate decisions given the specificities of a given environment and deployment.

The main risk associated with privacy is the ability to identify an application or a device by analyzing the traffic, which is designated as "traffic shaping". As discussed in Section 3, the use in a very specific context of non-randomly generated SPIs might ease the determination of the device or the application in some cases. Similarly, padding provides limited capabilities to obfuscate the traffic compared to those provided by TFC. Such consequences on privacy are detailed in Section 5.

12. References

12.1. 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>.

12.2. Informative References

[BEET-ESP] 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://datatracker.ietf.org/doc/html/draft-nikander-esp-beet-mode-09>.

[DeoxysII] Jean, J., Nikolić, I., Peyrin, T., and Y. Seurin, "Deoxys v1.41", October 2016, <https://competitions.cr.yp.to/round3/deoxysv141.pdf>.

[EHC-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-08, 13 May 2022, <https://datatracker.ietf.org/doc/html/draft-mglt-ipsecme-diet-esp-08>.

[EHC-IKEv2] 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-02, 13 May 2022, <https://datatracker.ietf.org/doc/html/draft-mglt-ipsecme-ikev2-diet-esp-extension-02>.

[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] Barker, E. and J. Kelsey, "Recommendation for Random Number Generation Using Deterministic Random Bit Generators", NIST SP 800-90A Rev 1, DOI 10.6028/NIST.SP.800-90Ar1, June 2015, <https://csrc.nist.gov/publications/detail/sp/800-90a/rev-1/final>.

Acknowledgments

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 also provided multiple clarifications and examples of ESP deployment 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. The authors would also like to thank Mohit Sethi for his support as the LWIG Working Group Chair.

Authors' Addresses

Daniel Migault
Ericsson
8275 Rte Transcanadienne
Saint-Laurent QC H4S 0B6
Canada
Email: daniel.migault@ericsson.com

Tobias Guggemos
LMU Munich
MNM-Team
Oettingenstr. 67
80538 Munich
Germany
Email: guggemos@mnm-team.org