Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 19, 2017

N. Madden
ForgeRock

June 17, 2017

Synthetic IV (SIV) encryption modes for JWE
draft-madden-jose-siv-mode-00

Abstract

This document defines how to use Synthetic Initialization Vector (SIV) encryption and key-wrapping modes with JSON Web Encryption (JWE), and registers identifiers for SIV-based key-wrapping and content encryption algorithms. SIV provides either deterministic authenticated encryption and key-wrapping, or nonce-based misuse-resistant authenticated encryption depending on usage.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://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 December 19, 2017.

Table of Contents

1. Introduction

This specification registers cryptographic algorithms and identifiers to be used with JSON Web Encryption (JWE) [RFC7516] for key-wrapping and content encryption based on Synthetic Initialization Vectors (SIV, or "Synthetic IV") [RFC5297]. SIV mode takes as input a key, the JWE Protected Header, an optional nonce (IV), and the plaintext payload, and produces a ciphertext having the same length as the plaintext and an authentication tag that also serves as the synthetic initialization vector.

This extends [RFC7518].

1.1. Requirements Terminology

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

1.2. Motivation

The motivations from [RFC5297] apply here.

Compared to the existing JWE AES Key Wrap algorithm [RFC7516] (Section 4.4), SIV provides a provable security bound.

For Content Encryption with a nonce, SIV is at least as efficient as AES_CBC_HMAC_SHA2, and typically less efficient than AES GCM for long messages. However, while the security of AES GCM collapses catastrophically if a key-nonce pair is reused, in SIV an attacker would only learn whether the same plaintext (and the same associated data) has been encrypted with the same key and nonce. This property, known as nonce misuse resistance (NMR), provides a measure of safety in the face of programming errors or poor quality nonce generation, such as misconfigured or interfered with random data generators, or accidental reuse due to logic errors in deterministic nonce generation algorithms (for instance, reusing nonces after a server restart).

Where the content or associated data of a JWE is known to contain a non-repeating value or key (such as a unique JWT ID), then the nonce can be omitted, resulting in a more compact serialisation. The nonce can also be omitted if deterministic authenticated encryption is desired.

For constrained devices, the abstract SIV scheme can be instantiated with AES in CTR mode for confidentiality, and CMAC [RFC4493] for authentication. In this instantiation the mode requires only an AES encryption circuit, providing similar benefits (and comparable performance) to AES CCM mode [RFC3610], but with the added robustness of nonce misuse resistance. The NMR property is particularly attractive for devices that have limited access to high-quality sources of entropy.

Finally, SIV allows a single construction to be used for both authenticated content encryption and key wrapping, and the construction itself is simple to describe and implement correctly from standard building blocks.

1.3. Notational Conventions

BASE64URL(OCTETS) denotes the base64url encoding of OCTETS, per Section 2 of [RFC7515].

UTF8(STRING) denotes the octets of the UTF-8 [RFC3629] representation of STRING, where STRING is a sequence of zero or more Unicode [UNICODE] characters.

ASCII(STRING) denotes the octets of the ASCII [RFC20] representation of STRING, where STRING is a sequence of zero or more ASCII characters.

The concatenation of two values A and B is denoted as A || B.

1.4. Terminology

These terms defined by the JSON Web Signature (JWS) [RFC7515] specification are incorporated into this specification: "Base64url Encoding"

These terms defined by the JSON Web Encryption (JWE) [RFC7516] specification are incorporated into this specification: "JSON Web Encryption (JWE)", "Additional Authenticated Data (AAD)", "Authentication Tag", "Content Encryption Key (CEK)", "JWE Authentication Tag", "JWE Ciphertext", "JWE Encrypted Key", "JWE Initialization Vector", "JWE Protected Header", and "Key Wrapping".

These terms defined by the Internet Security Glossary, Version 2 [RFC4949] are incorporated into this specification: "Ciphertext", "Message Authentication Code (MAC)", and "Plaintext".

2. Algorithms

2.1. Generic SIV Construction

This section defines a family of authenticated encryption algorithms built using a combination of AES in Counter (CTR) mode and either CMAC or HMAC-SHA2 operations. The presentation here is based on the abstract SIV scheme in Section 4 of [SIV]. The generic construction is parameterised by the size of the key and the instantiation of the MAC algorithm. We use MAC(K, M) to denote the application of the MAC algorithm to the given message M using the given key K. We use AES-CTR(K, IV, M) to denote the application of AES in CTR mode to the message M, using the key K and Initialization Vector IV.

Rather than adopting the S2V construction of [RFC5297] for providing multiple Additional Authentication Data (AAD) blocks to the MAC, we

instead adopt a simpler method based on the base64url-encoded compact serialisation of the JWE Protected Header and IV separated by dots, and the unencoded plaintext octets. This encoding uniquely determines the components of the AAD while being simpler, and uses encoded components that are already produced if the Compact Serialization is being used. As stated in Section 5 of [SIV], the motivation for the S2V construction is efficiency rather than security, and any unambiguous encoding will suffice. It is expected that a simpler construction will aid adoption of these safer encryption modes in situations where performance is not of paramount importance.

[Note to reviewers: there is an I-D defining an AES-GCM-SIV mode currently in progress (https://tools.ietf.org/html/draft-irtf-cfrg-gcmsiv-05). This is a much more high-performance SIV mode than the ones defined in this document. I have left it out of this specification because it is more complex to implement and still in draft form. A further I-D/RFC could be proposed to also add that mode for in this same framework, but I believe the modes defined in the present I-D will be useful for many years to come, especially on constrained devices.]

For the CMAC-based algorithms, we only define modes for an overall 128-bit security level. That is, the expected effort for an attacker to either produce an authentication tag forgery, recover either the encryption or MAC keys, or to compromise the privacy of a any SIV-encrypted JWE, is on the order of 2^128 operations. For the HMAC-based algorithms we define modes at overall 128-bit, 192-bit and 256-bit security levels. The reason for this is that AES-CMAC is only capable of producing a maximum authentication tag of 128 bits and so cannot provide more than 128 bits of protection against authentication tag forgery.

2.1.1. Encryption

The authenticated encryption algorithm takes as input for octet strings: a secret key K, a plaintext P, additional authenticated data AAD (computed as per Steps 13-14 of Section 5.1 of [RFC7516]), and an optional initialization vector IV. It produces the ciphertext value E and an authentication tag T as outputs. The data in the plaintext are encrypted, and the additional authenticated data are authenticated, but not encrypted.

Encryption is performed using the following steps:

1. The secondary keys MAC_KEY and ENC_KEY are generated from the input key K as follows. Each of these two keys is an octet string.

MAC_KEY consists of the initial MAC_KEY_LEN octets of K, in order.

ENC_KEY consists of the final ENC_KEY_LEN octets of K, in order.

The number of octets in the input key K MUST be the sum of MAC_KEY_LEN and ENC_KEY_LEN.

2. If a nonce is to be used, then the IV SHOULD be a 128-bit value generated randomly or pseudorandomly.

3. A message Authentication Tag T is computed as:

   1. T = MAC(MAC_KEY, ASCII(AAD || '.' || BASE64URL(IV) || '.') || plaintext).

   If no IV (nonce) is being used, then an empty octet sequence MUST be used instead.

4. The Synthetic IV, SIV, is set to the first 16 octets of T, in order.

5. The plaintext is encrypted using AES-CTR with ENC_KEY as the key and SIV as the IV. We denote the ciphertext output of this step as E.

6. The ciphertext E and the Authentication Tag T are returned as the outputs of the authenticated encryption.

The encryption process can be illustrated as follows. Here K, P, AAD, IV, SIV, T, and E denote the key, plaintext, Additional Authenticated Data, Initialization Vector, Synthetic IV, Authentication Tag, and ciphertext, respectively.

MAC_KEY = initial MAC_KEY_LEN octets of K,

ENC_KEY = final ENC_KEY_LEN octets of K,

T = MAC(MAC_KEY, ASCII(BASE64URL(AAD) || '.' || BASE64URL(IV) || '.') || P),

SIV = initial 16 octets of T,

E = AES-CTR(ENC_KEY, SIV, P).

2.1.2. Decryption

Decryption is performed using the following steps:

1. The secondary keys MAC_KEY and ENC_KEY are generated from the input key K as in Step 1 of Section 2.1.1.

2. The Synthetic IV is set to the first 16 octets of the Authentication Tag T. If the Authentication Tag is missing or not of the expected length for the algorithm (which is always at least 16 octets) then decryption MUST halt with an indication of failure.

3. The plaintext P is decrypted using AES-CTR with ENC_KEY as the key, SIV as the IV, and the ciphertext, E.

4. The Authentication Tag T is checked by recomputing the tag T' as in Step 3 of Section 2.1.1. If T and T' are identical then H and P are considered valid and processing is continued. Otherwise, all of the data used in the MAC computation MUST be discarded and the decryption operation MUST halt with an indication of failure. Tag comparison MUST use a constant-time octet string comparison operation using the known length of the Authentication Tag as specified by the algorithm in use.

5. The plaintext P is returned.

2.2. SIV Key Wrapping

The following JWE algorithms are defined here (to be applied as values of "alg" parameter):

"alg" Param Value	Key Management Algorithm
A128SIVKW	AES SIV Key Wrap using CMAC and 256 bit key.
A128SIVKW-HS256	AES SIV Key Wrap using HMAC-SHA-256-128 and 256 bit key.
A192SIVKW-HS384	AES SIV Key Wrap using HMAC-SHA-384-192 and 384 bit key.
A256SIVKW-HS512	AES SIV Key Wrap using HMAC-SHA-512-256 and 512 bit key.

All of the key wrapping modes use the generic construction from Section 2.1, with the following inputs:

The plaintext P is the octets of the Content Encryption Key (CEK) to be wrapped.

The input key K is the Key Encryption Key (KEK).

The IV is an empty octet sequence.

The AAD is the UTF8 octets of the value of the "alg" parameter (e.g., "A128SIVKW").

In all cases the output ciphertext length will be the same as the input plaintext CEK, in octets. The authentication tag will either be 16, 24 or 32 octets long depending on the algorithm.

The JWE Encrypted Key value is the Ciphertext output.

The Authentication Tag output is represented in base64url encoded form as the "tag" (authentication tag) Header Parameter value, as in Section 4.7.1.2 of [RFC7518]. This specification extends that header value to allow authentication tags of 192 or 256 bits.

2.2.1. A128SIVKW

This algorithm uses the CMAC message authentication code [RFC4493] to provide message authentication and the synthetic IV.

The parameters are as follows:

The input key K is 32 octets long.

MAC_KEY_LEN is 16 octets.

ENC_KEY_LEN is 16 octets.

MAC is CMAC.

The output tag length is 16 octets.

2.2.2. A128SIVKW-HS256

This algorithm uses the HMAC-SHA-256-128 message authentication code as defined in [RFC4868] to provide message authentication and the synthetic IV.

The parameters are as follows:

The input key K is 32 octets long.

MAC_KEY_LEN is 16 octets.

ENC_KEY_LEN is 16 octets.

MAC is HMAC-SHA-256-128.

The output tag length is 16 octets.

2.2.3. A192SIVKW-HS384

This algorithm uses the HMAC-SHA-384-192 message authentication code as defined in [RFC4868] to provide message authentication and the synthetic IV.

The parameters are as follows:

The input key K is 48 octets long.

MAC_KEY_LEN is 24 octets.

ENC_KEY_LEN is 24 octets.

MAC is HMAC-SHA-384-192.

The output tag length is 24 octets.

2.2.4. A256SIVKW-HS512

This algorithm uses the HMAC-SHA-512-256 message authentication code as defined in [RFC4868] to provide message authentication and the synthetic IV.

The parameters are as follows:

The input key K is 64 octets long.

MAC_KEY_LEN is 32 octets.

ENC_KEY_LEN is 32 octets.

MAC is HMAC-SHA-512-256.

The output tag length is 32 octets.

2.3. SIV Content Encryption

The following content encryption methods are defined here (to be applied as values of the "enc" parameter):

"enc" Param Value	Content Encryption Method
A128SIV	AES SIV using CMAC and 256 bit key.
A128SIV-HS256	AES SIV using HMAC-SHA-256-128 and 256 bit key.
A192SIV-HS384	AES SIV using HMAC-SHA-384-192 and 384 bit key.
A256SIV-HS512	AES SIV using HMAC-SHA-512-256 and 512 bit key.

All of the SIV content encryption methods use the generic construction from Section 2.1, with the following inputs:

The plaintext P is the octets of JWE plaintext.

The input key K is the Content Encryption Key (CEK).

The IV is either a randomly or pseudorandomly generated 16 octet value, or an empty octet string.

The AAD is the UTF8 octets of the JWE Protected Header.

In all cases the output ciphertext length will be the same as the input plaintext, in octets. The authentication tag will either be 16, 24 or 32 octets long depending on the algorithm. The Ciphertext and Authentication Tag outputs become the JWE Ciphertext and JWE Authentication Tag values respectively.

2.3.1. A128SIV

This algorithm uses the CMAC message authentication code [RFC4493] to provide message authentication and the synthetic IV.

The parameters are as follows:

The input key K is 32 octets long.

MAC_KEY_LEN is 16 octets.

ENC_KEY_LEN is 16 octets.

MAC is CMAC.

The output tag length is 16 octets.

2.3.2. A128SIV-HS256

This algorithm uses the HMAC-SHA-256-128 message authentication code as defined in [RFC4868] to provide message authentication and the synthetic IV.

The parameters are as follows:

The input key K is 32 octets long.

MAC_KEY_LEN is 16 octets.

ENC_KEY_LEN is 16 octets.

MAC is HMAC-SHA-256-128.

The output tag length is 16 octets.

2.3.3. A192SIV-HS384

This algorithm uses the HMAC-SHA-384-192 message authentication code as defined in [RFC4868] to provide message authentication and the synthetic IV.

The parameters are as follows:

The input key K is 48 octets long.

MAC_KEY_LEN is 24 octets.

ENC_KEY_LEN is 24 octets.

MAC is HMAC-SHA-384-192.

The output tag length is 24 octets.

2.3.4. A256SIV-HS512

This algorithm uses the HMAC-SHA-512-256 message authentication code as defined in [RFC4868] to provide message authentication and the synthetic IV.

The parameters are as follows:

The input key K is 64 octets long.

MAC_KEY_LEN is 32 octets.

ENC_KEY_LEN is 32 octets.

MAC is HMAC-SHA-512-256.

The output tag length is 32 octets.

3. IANA considerations

The following are added to JSON Web Signature and Encryption Algorithms registry:

- Algorithm Name: "A128SIVKW"
- Algorithm Description: AES SIV Key Wrap with CMAC using 256 bit key
- Algorithm Usage Location(s): "alg"
- JOSE Implementation Requirements: Recommended
- Change Controller: IESG
- Specification Document(s): Section 2.2.1

- Algorithm Name: "A128SIVKW-HS256"
- Algorithm Description: AES SIV Key Wrap with HMAC-SHA-256-128 using 256 bit key
- Algorithm Usage Location(s): "alg"
- JOSE Implementation Requirements: Recommended
- Change Controller: IESG
- Specification Document(s): Section 2.2.2

- Algorithm Name: "A192SIVKW-HS384"
- Algorithm Description: AES SIV Key Wrap with HMAC-SHA-384-192 using 384 bit key
- Algorithm Usage Location(s): "alg"
- JOSE Implementation Requirements: Optional
- Change Controller: IESG
- Specification Document(s): Section 2.2.3

- Algorithm Name: "A256SIVKW-HS512"
- Algorithm Description: AES SIV Key Wrap with HMAC-SHA-512-256 using 512 bit key
- Algorithm Usage Location(s): "alg"
- JOSE Implementation Requirements: Optional
- Change Controller: IESG
- Specification Document(s): Section 2.2.4

- Algorithm Name: "A128SIV"
- Algorithm Description: AES SIV with CMAC using 256 bit key
- Algorithm Usage Location(s): "enc"
- JOSE Implementation Requirements: Recommended
- Change Controller: IESG
- Specification Document(s): Section 2.3.1

- Algorithm Name: "A128SIV-HS256"
- Algorithm Description: AES SIV with HMAC-SHA-256-128 using 256 bit key
- Algorithm Usage Location(s): "enc"
- JOSE Implementation Requirements: Recommended
- Change Controller: IESG
- Specification Document(s): Section 2.3.2

- Algorithm Name: "A192SIV-HS284"
- Algorithm Description: AES SIV with HMAC-SHA-384-192 using 384 bit key
- Algorithm Usage Location(s): "enc"
- JOSE Implementation Requirements: Optional
- Change Controller: IESG
- Specification Document(s): Section 2.3.3

- Algorithm Name: "A256SIV-HS512"
- Algorithm Description: AES SIV with HMAC-SHA-512-256 using 512 bit key
- Algorithm Usage Location(s): "enc"
- JOSE Implementation Requirements: Optional
- Change Controller: IESG
- Specification Document(s): Section 2.3.4

4. Security Considerations

The security considerations of [RFC5297] apply here.

In total, no more than 16 * 2^48 octets of data (approx. 4 exabytes) should be encrypted with the same key in any SIV mode. For example, when using SIV128KW to wrap 128-bit keys, then no more than 2^48 messages should be encrypted with the same key encryption key (KEK). This is over 281 trillion messages, so is expected to provide sufficient capacity for extremely long-lived or high-usage keys.

SIV uses AES in CTR mode for encryption, which produces ciphertexts that are exactly the same length as the plaintext. If the length of

the plaintext is sensitive (for instance, when encrypting a password that may be short) then the application should pad such values to some minimum/fixed size before encryption. If such padding is performed, then it MUST be applied before calling the AES-SIV encryption modes defined in this specification, so that the padding is included in the authentication tag. When decrypting, authentication tag validation in Step 4 of Section 2.1.2 MUST be performed before any validation or processing of the padding is performed.

Care should be taken when combining JWE plaintext compression with SIV encryption for a related reason: compression varies the size of the plaintext based on the (confidential) content of that plaintext. In SIV mode (and other cipher modes, such as GCM and, to a lesser extent, CBC), this will vary the size of the ciphertext by the same amount. If an attacker is able to control any part of the content of the plaintext then they may be able to infer confidential parts of the same plaintext according to variations in the size of the compressed and encrypted ciphertext. It is therefore recommended not to use compression with SIV mode encryption (or any encryption) unless the expected information leakage is acceptable.

5. References

5.1. Normative References

[RFC20] Cerf, V., "ASCII format for Network Interchange", RFC 20, October 1969.

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

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <http://www.rfc-editor.org/info/rfc3629>.

[RFC4493] Song, JH., Poovendran, R., Lee, J., and T. Iwata, "The AES-CMAC Algorithm", RFC 4493, DOI 10.17487/RFC4493, June 2006, <http://www.rfc-editor.org/info/rfc4493>.

[RFC4868] Kelly, S. and S. Frankel, "Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec", RFC 4868, DOI 10.17487/RFC4868, May 2007, <http://www.rfc-editor.org/info/rfc4868>.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007, <http://www.rfc-editor.org/info/rfc4949>.

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

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <http://www.rfc-editor.org/info/rfc7515>.

[RFC7516] Jones, M. and J. Hildebrand, "JSON Web Encryption (JWE)", RFC 7516, DOI 10.17487/RFC7516, May 2015, <http://www.rfc-editor.org/info/rfc7516>.

[RFC7518] Jones, M., "JSON Web Algorithms (JWA)", RFC 7518, DOI 10.17487/RFC7518, May 2015, <http://www.rfc-editor.org/info/rfc7518>.

[UNICODE] The Unicode Consortium, "The Unicode Standard", 1991-, <http://www.unicode.org/versions/latest/>.

5.2. Informative References

[RFC3610] Whiting, D., Housley, R., and N. Ferguson, "Counter with CBC-MAC (CCM)", RFC 3610, DOI 10.17487/RFC3610, September 2003, <http://www.rfc-editor.org/info/rfc3610>.

[SIV] Rogaway, P. and T. Shrimpton, "Deterministic Authenticated-Encryption. A Provable-Security Treatment of the Key-Wrap Problem.", IACR ePrint 2006/221, August 2007.

Appendix A. Test Cases

The following test cases can be used to validate implementations of the AES SIV algorithms defined in this specification.

The variable names are those defined in Section 2.1.1. All values are hexadecimal.

A.1. Test Cases for A128SIVKW

NB: K here is the KEK, and P is the CEK to be wrapped, T is the output "tag" value, and E is the wrapped CEK.

A128SIVKW

```
K =       00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
          10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f

MAC_KEY = 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f

ENC_KEY = 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f

P =       0f 0e 0d 0c 0b 0a 09 08 07 06 05 04 03 02 01 00

IV =      <empty octet sequence>

AAD =     41 31 32 38 53 49 56 4b 57

T =       c3 eb 04 f1 c7 07 8b 92 e0 dc f6 fe 17 f5 82 46

SIV =     c3 eb 04 f1 c7 07 8b 92 e0 dc f6 fe 17 f5 82 46

E =       ef 96 fd 87 24 ea f9 9b 54 15 8a fa 20 5f 77 de
```

A.2. Test Cases for A192SIVKW-HS384

NB: K here is the KEK, and P is the CEK to be wrapped, T is the output "tag" value, and E is the wrapped CEK.

A192SIVKW-HS384

```
K =       00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
          10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
          20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f

MAC_KEY = 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
          10 11 12 13 14 15 16 17

ENC_KEY = 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 26 27
          28 29 2a 2b 2c 2d 2e 2f

P =       17 16 15 14 13 12 11 10 0f 0e 0d 0c 0b 0a 09 08
          07 06 05 04 03 02 01 00

IV =      <empty octet sequence>

AAD =     41 31 39 32 53 49 56 4b 57 2d 48 53 33 38 34

T =       27 86 b6 03 3b b1 4f f7 cb 85 6d ae 69 6e 3d 98
          ff e2 0b 59 77 b3 e5 36

SIV =     c3 eb 04 f1 c7 07 8b 92 e0 dc f6 fe 17 f5 82 46

E =       65 c5 52 72 4e d3 4f 9e ab 20 32 4d af 0d 2d 31
          7f df 69 13 06 c5 0a c8
```

A.3. Test Cases for A128SIV-HS256

A128SIV-HS256

```
K =       00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
          10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f

MAC_KEY = 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f

ENC_KEY = 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f

P =       41 20 63 69 70 68 65 72 20 73 79 73 74 65 6d 20
          6d 75 73 74 20 6e 6f 74 20 62 65 20 72 65 71 75
          69 72 65 64 20 74 6f 20 62 65 20 73 65 63 72 65
          74 2c 20 61 6e 64 20 69 74 20 6d 75 73 74 20 62
          65 20 61 62 6c 65 20 74 6f 20 66 61 6c 6c 20 69
          6e 74 6f 20 74 68 65 20 68 61 6e 64 73 20 6f 66
          20 74 68 65 20 65 6e 65 6d 79 20 77 69 74 68 6f
          75 74 20 69 6e 63 6f 6e 76 65 6e 69 65 6e 63 65

IV =      1a f3 8c 2d c2 b9 6f fd d8 66 94 09 23 41 bc 04

AAD =     7b 22 61 6c 67 22 3a 22 64 69 72 22 2c 22 65 6e
          63 22 3a 22 41 31 32 38 53 49 56 2d 48 53 32 35
          36 22 7d

T =       5e cd e7 ca 4a eb 39 bc 05 11 2b a9 00 17 a3 76

SIV =     5e cd e7 ca 4a eb 39 bc 05 11 2b a9 00 17 a3 76

E =       22 70 54 15 99 71 ca d6 01 8c d9 30 29 e6 e5 20
          5d 0a d3 d2 1e 8c 10 ce 6f 84 36 e3 68 20 24 42
          59 e8 ae bd 55 16 ce 37 ab 5a 44 3b 22 0a 94 a0
          03 7f 4a ad 4d 11 57 db 55 cb 6a 01 70 8b 05 0d
          6f 39 ad b4 d8 3b 5c 77 ac 16 6a 98 cc 0e 0a 75
          93 f6 34 6e 67 b1 9d 4c 43 17 11 95 7b b5 e3 8b
          ee cb df 2e 7f 49 c0 ba c3 58 5b 90 32 b4 bc ca
          08 6b 51 a8 c5 d3 81 a7 fd d8 c3 fb 99 6e 25 46
```

A.4. Test Cases for A256SIV-HS512

A256SIV-HS512

```
K =       00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
          10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
          20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f
          30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f

MAC_KEY = 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
          10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f

ENC_KEY = 20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f
          30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f

P =       41 20 63 69 70 68 65 72 20 73 79 73 74 65 6d 20
          6d 75 73 74 20 6e 6f 74 20 62 65 20 72 65 71 75
          69 72 65 64 20 74 6f 20 62 65 20 73 65 63 72 65
          74 2c 20 61 6e 64 20 69 74 20 6d 75 73 74 20 62
          65 20 61 62 6c 65 20 74 6f 20 66 61 6c 6c 20 69
          6e 74 6f 20 74 68 65 20 68 61 6e 64 73 20 6f 66
          20 74 68 65 20 65 6e 65 6d 79 20 77 69 74 68 6f
          75 74 20 69 6e 63 6f 6e 76 65 6e 69 65 6e 63 65

IV =      1a f3 8c 2d c2 b9 6f fd d8 66 94 09 23 41 bc 04

AAD =     7b 22 61 6c 67 22 3a 22 64 69 72 22 2c 22 65 6e
          63 22 3a 22 41 32 35 36 53 49 56 2d 48 53 35 31
          32 22 7d

T =       f9 e5 2d 5c 58 9d 3a f8 3f 98 3f ce 3b 98 aa ae
          97 aa 0c 02 e1 80 a4 ec a3 0b 5e 7b 47 97 a5 b2

SIV =     f9 e5 2d 5c 58 9d 3a f8 3f 98 3f ce 3b 98 aa ae

E =       cc 05 71 16 ad 3d 44 9b 50 ba 7b bd b4 42 f7 08
          20 fe bc d0 58 0e 8d 4d e0 f3 61 70 6b db b6 17
          a6 d6 a9 56 e5 69 cc 74 d3 16 7d 2c a2 a6 54 2e
          e7 69 64 9c db 4d 9b 68 b7 01 74 f8 a4 4e eb 9e
          a0 26 8a 3c 48 e9 c8 88 56 c4 2c eb 36 95 d2 90
          39 18 34 5d d2 f8 17 20 bb ce be 24 bf f1 74 68
          26 bb c9 c8 11 92 9d 45 ce dd 63 49 2d ed b6 c0
          b2 b5 bd c4 93 a6 0f e6 c7 c6 e7 fd 94 90 3d 03
```

Author's Address

Neil Madden
ForgeRock

Email: neil.madden@forgerock.com