SUIT
Internet-Draft
Intended status: Standards Track
Expires: January 8, 2022
H. Tschofenig
Arm Limited
R. Housley
Vigil Security
B. Moran
Arm Limited
July 07, 2021

Firmware Encryption with SUIT Manifests
draft-ietf-suit-firmware-encryption-00

Abstract

This document specifies a firmware update mechanism where the firmware image is encrypted. This mechanism uses the IETF SUIT manifest with key establishment provided by the hybrid public-key encryption (HPKE) scheme or AES Key Wrap (AES-KW) with a pre-shared key-encryption key. In either case, AES-GCM or AES-CCM is used for firmware encryption.

Table of Contents

1. Introduction

Vulnerabilities with Internet of Things (IoT) devices have raised the need for a reliable and secure firmware update mechanism that is also suitable for constrained devices. To protect firmware images the SUIT manifest format was developed [I-D.ietf-suit-manifest]. The SUIT manifest provides a bundle of metadata about the firmware for an IoT device, where to find the firmware image, and the devices to which it applies.

The SUIT information model [I-D.ietf-suit-information-model] details the information that has to be offered by the SUIT manifest format. In addition to offering protection against modification, which is provided by a digital signature or a message authentication code, the firmware image may also be afforded confidentiality using encryption.

Encryption prevents third parties, including attackers, from gaining access to the firmware image. For example, return-oriented programming (ROP) requires intimate knowledge of the target firmware and encryption makes this approach much more difficult to exploit. The SUIT manifest provides the data needed for authorized recipients of the firmware image to decrypt it.

A symmetric cryptographic key is established for encryption and decryption, and that key can be applied to a SUIT manifest, firmware images, or personalization data, depending on the encryption choices of the firmware author. This symmetric key can be established using a variety of mechanisms; this document defines two approaches for use with the IETF SUIT manifest. Key establishment can be provided by the hybrid public-key encryption (HPKE) scheme or AES Key Wrap (AES-KW) with a pre-shared key-encryption key. These choices reduce the number of possible key establishment options for interoperability of different SUIT manifest implementations. The document also offers a number of examples for developers.

2. Conventions and 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 assumes familiarity with the IETF SUIT manifest [I-D.ietf-suit-manifest] and the SUIT architecture [RFC9019].

The terms "recipient" and "firmware consumer" are used interchangeably.

Additionally, the following abbreviations are used in this document:

- Key Wrap (KW), defined in RFC 3394 [RFC3394] for use with AES.

- Key-encryption key / key-encrypting key (KEK), a term defined in RFC 4949 [RFC4949].

- Content-encryption key (CEK), a term defined in RFC 2630 [RFC2630].

- Hybrid Public Key Encryption (HPKE), defined in [I-D.irtf-cfrg-hpke].

3. AES Key Wrap

The AES Key Wrap (AES-KW) algorithm is described in RFC 3394 [RFC3394], and it can be used to encrypt a randomly generated content-encryption key (CEK) with a pre-shared key-encryption key (KEK). The COSE conventions for using AES-KW are specified in Section 12.2.1 of [RFC8152]. The encrypted CEK is carried in the COSE_recipient structure alongside the information needed for AES-KW. The COSE_recipient structure, which is a substructure of the COSE_Encrypt, contains the CEK encrypted by the KEK. When the firmware image is encrypted for use by multiple recipients, the COSE_recipient structure will contain one encrypted CEK if all of the authorized recipients have access to the KEK.

However, the COSE_recipient structure can contain the same CEK encrypted with many different KEKs if needed to reach all of the authorized recipients.

Note that the AES-KW algorithm, as defined in Section 2.2.3.1 of [RFC3394], does not have public parameters that vary on a per-invocation basis. Hence, the protected structure in the COSE_recipient is a byte string of zero length.

The COSE_Encrypt conveys information for encrypting the firmware image, which includes information like the algorithm and the IV, even though the firmware image is not embedded in the COSE_Encrypt.ciphertext itself since it conveyed as detached content.

The CDDL for the COSE_Encrypt_Tagged structure is shown in Figure 1.

```
COSE_Encrypt_Tagged = #6.96(COSE_Encrypt)

SUIT_Encryption_Info = COSE_Encrypt_Tagged

COSE_Encrypt = [
  protected   : bstr .cbor outer_header_map_protected,
  unprotected : outer_header_map_unprotected,
  ciphertext  : null,                  ; because of detached ciphertext
  recipients  : [ + COSE_recipient ]
]

outer_header_map_protected =
{
    1 => int,         ; algorithm identifier
  * label =values     ; extension point
}

outer_header_map_unprotected =
{
    5 => bstr,        ; IV
  * label =values     ; extension point
}

COSE_recipient = [
  protected   : bstr .size 0,
  unprotected : recipient_header_map,
  ciphertext  : bstr        ; CEK encrypted with KEK
]

recipient_header_map =
{
    1 => int,         ; algorithm identifier
    4 => bstr,        ; key identifier
  * label =values     ; extension point
}
```

Figure 1: CDDL for AES Key Wrap-based Firmware Encryption

The COSE specification requires a consistent byte stream for the authenticated data structure to be created, which is defined as shown in Figure 2.

```
Enc_structure = [
  context : "Encrypt",
  protected : empty_or_serialized_map,
  external_aad : bstr
]
```

Figure 2: CDDL for Enc_structure Data Structure

As it can be seen in the CDDL in Figure 1, there are two protected fields and the 'protected' field in the Enc_structure, see Figure 2, refers to the outer protected field, not the protected field of the COSE_recipient structure.

The value of the external_aad is set to null.

The following example illustrates the use of the AES-KW algorithm with AES-128.

We use the following parameters in this example:

- IV: 0x26, 0x68, 0x23, 0x06, 0xd4, 0xfb, 0x28, 0xca, 0x01, 0xb4, 0x3b, 0x80

- KEK: "aaaaaaaaaaaaaaaa"

- KID: "kid-1"

- Plaintext Firmware: "This is a real firmware image."

- Firmware (hex): 546869732069732061207265616C206669726D7761726520696D6167652E

The COSE_Encrypt structure in hex format is (with a line break inserted):

D8608443A10101A1054C26682306D4FB28CA01B43B80F68340A2012204456B69642D
315818AF09622B4F40F17930129D18D0CEA46F159C49E7F68B644D

The resulting COSE_Encrypt structure in a dignostic format is shown in Figure 3.

```
96(
    [
        // protected field with alg=AES-GCM-128
        h'A10101',
        {
           // unprotected field with iv
           5: h'26682306D4FB28CA01B43B80'
        },
        // null because of detached ciphertext
        null,
        [ // recipients array
           h'', // protected field
           {    // unprotected field
              1: -3,            // alg=A128KW
              4: h'6B69642D31'  // key id
           },
           // CEK encrypted with KEK
           h'AF09622B4F40F17930129D18D0CEA46F159C49E7F68B644D'
        ]
    ]
)
```

Figure 3: COSE_Encrypt Example for AES Key Wrap

The CEK was "4C805F1587D624ED5E0DBB7A7F7FA7EB" and the encrypted firmware was:

```
A8B6E61EF17FBAD1F1BF3235B3C64C06098EA512223260
F9425105F67F0FB6C92248AE289A025258F06C2AD70415
```

4. Hybrid Public-Key Encryption (HPKE)

Hybrid public-key encryption (HPKE) [I-D.irtf-cfrg-hpke] is a scheme that provides public key encryption of arbitrary-sized plaintexts given a recipient's public key.

For use with firmware encryption the scheme works as follows: The firmware author uses HPKE, which internally utilizes a non-interactive ephemeral-static Diffie-Hellman exchange to derive a shared secret, which is then used to encrypt plaintext. In the firmware encryption scenario, the plaintext passed to HPKE for encryption is a randomly generated CEK. The output of the HPKE operation is therefore the encrypted CEK along with HPKE encapsulated key (i.e. the ephemeral ECDH public key of the author). The CEK is then used to encrypt the firmware.

Only the holder of recipient's private key can decapsulate the CEK to decrypt the firmware. Key generation is influced by additional parameters, such as identity information.

This approach allows us to have all recipients to use the same CEK to encrypt the firmware image, in case there are multiple recipients, to fulfill a requirement for the efficient distribution of firmware images using a multicast or broadcast protocol.

The CDDL for the COSE_Encrypt structure as used with HPKE is shown in Figure 4.

```
COSE_Encrypt_Tagged = #6.96(COSE_Encrypt)

SUIT_Encryption_Info = COSE_Encrypt_Tagged

COSE_Encrypt = [
  protected   : bstr .cbor header_map, ; must contain alg
  unprotected : header_map,            ; must contain iv
  ciphertext  : null,                  ; because of detached ciphertext
  recipients  : [ + COSE_recipient_outer ]
]

COSE_recipient_outer = [
  protected   : bstr .size 0,
  unprotected : header_map, ; must contain alg
  ciphertext  : bstr        ; CEK encrypted based on HPKE algo
  recipients  : [ + COSE_recipient_inner ]
]

COSE_recipient_inner = [
  protected   : bstr .cbor header_map, ; must contain alg
  unprotected : header_map, ; must contain kid,
  ciphertext  : bstr        ; CEK encrypted based on HPKE algo
  recipients  : null
]

header_map = {
  Generic_Headers,
  * label =values,
}

Generic_Headers = (
    ? 1 => int,         ; algorithm identifier
    ? 2 => crv,         ; EC identifier
    ? 4 => bstr,        ; key identifier
    ? 5 => bstr         ; IV
)
```

Figure 4: CDDL for HPKE-based COSE_Encrypt Structure

The COSE_Encrypt structure in Figure 4 requires the encrypted CEK and the ephemeral public key of the firmare author to be generated. This is accomplished with the HPKE encryption function as shown in Figure 5.

```
CEK = random()
pkR = DeserializePublicKey(recipient_public_key)
info = "cose hpke" || 0x00 || COSE_KDF_Context
enc, context = SetupBaseS(pkR, info)
ciphertext = context.Seal(null, CEK)
```

Figure 5

Legend:

- The functions DeserializePublicKey(), SetupBaseS() and Seal() are defined in HPKE [I-D.irtf-cfrg-hpke].

- CEK is a random byte sequence of keysize length whereby keysize corresponds to the size of the indicated symmetric encryption algorithm used for firmware encryption. For example, AES-128-GCM requires a 16 byte key. The CEK would therefore be 16 bytes long.

- 'recipient_public_key' represents the public key of the recipient.

- 'info' is a data structure described below used as input to the key derivation internal to the HPKE algorithm. In addition to the constant prefix, the COSE_KDF_Context structure is used. The COSE_KDF_Context is shown in Figure 6.

The result of the above-described operation is the encrypted CEK (denoted as ciphertext) and the enc - the HPKE encapsulated key (i.e. the ephemeral ECDH public key of the author).

```
PartyInfo = (
   identity : bstr,
   nonce : nil,
   other : nil
)

COSE_KDF_Context = [
   AlgorithmID : int,
   PartyUInfo : [ PartyInfo ],
   PartyVInfo : [ PartyInfo ],
   SuppPubInfo : [
       keyDataLength : uint,
       protected : empty_or_serialized_map
   ],
]
```

Figure 6: COSE_KDF_Context Data Structure

Notes:

- PartyUInfo.identity corresponds to the kid found in the COSE_Sign_Tagged or COSE_Sign1_Tagged structure (when a digital signature is used. When utilizing a MAC, then the kid is found in the COSE_Mac_Tagged or COSE_Mac0_Tagged structure.

- PartyVInfo.identity corresponds to the kid used for the respective recipient from the inner-most recipients array.

- The value in the AlgorithmID field corresponds to the alg parameter in the protected structure in the inner-most recipients array.

- keyDataLength is set to the number of bits of the desired output value.

- protected refers to the protected structure of the inner-most array.

The author encrypts the firmware using the CEK with the selected algorithm.

The recipient decrypts the received ciphertext, i.e. the encrypted CEK, using two input parameters:

- the private key skR corresponding to the public key pkR used by the author when creating the manifest.

- the HPKE encapsulated key (i.e. ephemeral ECDH public key) created by the author.

If the HPKE operation is successful, the recipient obtains the CEK and can decrypt the firmware.

Figure 7 shows the HPKE computations performed by the recipient for decryption.

```
info = "cose hpke" || 0x00 || COSE_KDF_Context
context = SetupBaseR(ciphertext, skR, info)
CEK = context.Open(null, ciphertext)
```

Figure 7

An example of the COSE_Encrypt structure using the HPKE scheme is shown in Figure 8.

```
96(
    [
        // protected field with alg=AES-GCM-128
        h'A10101',
        {    // unprotected field with iv
             5: h'26682306D4FB28CA01B43B80'
        },
        // null because of detached ciphertext
        null,
        [  // COSE_recipient_outer
            h'',          // empty protected field
            {             // unprotected field with ...
                 1: 1     //     alg=A128GCM
            },
            // Encrypted CEK
            h'FA55A50CF110908DA6443149F2C2062011A7D8333A72721A',
            [    // COSE_recipient_inner
                 // protected field with alg HPKE/P-256+HKDF-256 (new)
                 h'A1013818',
                 {  // unprotected field with ...
                    //    HPKE encapsulated key
                    -1: h'A4010220012158205F...979D51687187510C445',
                    //    kid for recipient static ECDH public key
                     4: h'6B69642D31'
                 },
                 // empty ciphertext
                 null
            ]
        ]
    ]
)
```

Figure 8: COSE_Encrypt Example for HPKE

5. Complete Examples

TBD: Add example for complete manifest here (which also includes the digital signature). TBD: Add multiple recipient example as well. TBD: Add encryption of manifest (in addition of firmware encryption).

6. Security Considerations

The algorithms described in this document assume that the firmware author

- has either shared a key-encryption key (KEK) with the firmware consumer (for use with the AES-Key Wrap scheme), or

- is in possession of the public key of the firmware consumer (for use with HPKE).

Both cases require some upfront communication interaction, which is not part of the SUIT manifest. This interaction is likely provided by a IoT device management solution, as described in [RFC9019].

For AES-Key Wrap to provide high security it is important that the KEK is of high entropy, and that implementations protect the KEK from disclosure. Compromise of the KEK may result in the disclosure of all key data protected with that KEK.

Since the CEK is randomly generated, it must be ensured that the guidelines for random number generations are followed, see [RFC8937].

7. IANA Considerations

This document requests IANA to create new entries in the COSE Algorithms registry established with [I-D.ietf-cose-rfc8152bis-algs].

Name	Value	KDF	Ephemeral-Static	Key Wrap	Description
HPKE/P-256+ HKDF-256	TBD1	HKDF - SHA-256	yes	none	HPKE with ECDH-ES (P-256) + HKDF-256
HPKE/P-384+ HKDF-SHA384	TBD2	HKDF - SHA-384	yes	none	HPKE with ECDH-ES (P-384) + HKDF-384
HPKE/P-521+ HKDF-SHA521	TBD3	HKDF - SHA-521	yes	none	HPKE with ECDH-ES (P-521) + HKDF-521
HPKE X25519 + HKDF-SHA256	TBD4	HKDF - SHA-256	yes	none	HPKE with ECDH-ES (X25519) + HKDF-256
HPKE X448 + HKDF-SHA512	TBD4	HKDF - SHA-512	yes	none	HPKE with ECDH-ES (X448) + HKDF-512

8. References

8.1. Normative References

[I-D.ietf-cose-rfc8152bis-algs]
August Cellars, "CBOR Object Signing and Encryption (COSE): Initial Algorithms", draft-ietf-cose-rfc8152bis-algs-12 (work in progress), September 2020.

[I-D.ietf-suit-manifest]
Arm Limited, Arm Limited, Fraunhofer SIT, and Inria, "A Concise Binary Object Representation (CBOR)-based Serialization Format for the Software Updates for Internet of Things (SUIT) Manifest", draft-ietf-suit-manifest-12 (work in progress), February 2021.

[I-D.irtf-cfrg-hpke]
Cisco, Inria, Inria, and Cloudflare, "Hybrid Public Key Encryption", draft-irtf-cfrg-hpke-08 (work in progress), February 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>.

[RFC3394] Schaad, J. and R. Housley, "Advanced Encryption Standard (AES) Key Wrap Algorithm", RFC 3394, DOI 10.17487/RFC3394, September 2002, <https://www.rfc-editor.org/info/rfc3394>.

[RFC8152] Schaad, J., "CBOR Object Signing and Encryption (COSE)", RFC 8152, DOI 10.17487/RFC8152, July 2017, <https://www.rfc-editor.org/info/rfc8152>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

8.2. Informative References

[I-D.ietf-suit-information-model]
Arm Limited, Arm Limited, and Fraunhofer SIT, "A Manifest Information Model for Firmware Updates in IoT Devices", draft-ietf-suit-information-model-11 (work in progress), April 2021.

[RFC2630] Housley, R., "Cryptographic Message Syntax", RFC 2630, DOI 10.17487/RFC2630, June 1999, <https://www.rfc-editor.org/info/rfc2630>.

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

[RFC8937] Cremers, C., Garratt, L., Smyshlyaev, S., Sullivan, N., and C. Wood, "Randomness Improvements for Security Protocols", RFC 8937, DOI 10.17487/RFC8937, October 2020, <https://www.rfc-editor.org/info/rfc8937>.

[RFC9019] Moran, B., Tschofenig, H., Brown, D., and M. Meriac, "A Firmware Update Architecture for Internet of Things", RFC 9019, DOI 10.17487/RFC9019, April 2021, <https://www.rfc-editor.org/info/rfc9019>.

Appendix A. Acknowledgements

We would like to thank Henk Birkholz for his feedback on the CDDL description in this document. Additionally, we would like to thank Michael Richardson and Carsten Bormann for their review feedback.

Authors' Addresses

Hannes Tschofenig
Arm Limited

EMail: hannes.tschofenig@arm.com

Russ Housley
Vigil Security, LLC

EMail: housley@vigilsec.com

Brendan Moran
Arm Limited

EMail: Brendan.Moran@arm.com