JOSE Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 15, 2013

M. Miller
Cisco Systems, Inc.

June 13, 2013

Using JavaScript Object Notation (JSON) Web Encryption (JWE) for Protecting JSON Web Key (JWK) Objects draft-miller-jose-jwe-protected-jwk-02

Abstract

This document specifies an approach to protecting a private key formatted as a JavaScript Syntax Object Notation (JSON) Web Key (JWK) object using JSON Web Encryption (JWE). This document also specifies a set of algorithms for protecting such content using password-based cryptography.

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 15, 2013.

Copyright Notice

Table of Contents

1. Introduction 2
2. Terminology 3
3. Protecting Keys 3
3.1. Details for Private Keys 4
3.2. Details for Symmetric Keys 5
4. Private Key Example 5
5. Symmetric Key Example 8
6. Using Password-Based Cryptography 9
6.1. PBKDF2 Key Type 9
6.1.1. 's' Parameter 10
6.1.2. 'c' Parameter 10
6.1.3. 'hint' Parameter 10
6.2. PBES2 Key Encryption Algorithms 10
6.2.1. PBES2-HS256+A128KW 11
6.2.2. PBES2-HS256+A256KW 11
7. IANA Considerations 11
7.1. JSON Web Key Types Registration 11
7.2. JSON Web Key Parameters Registration 11
7.3. JSON Web Encryption Algorithms 12
8. Security Considerations 13
8.1. Re-using Keying Material 13
8.2. Password Considerations 13
9. Internationalization Considerations 13
10. References 14
10.1. Normative References 14
10.2. Informative References 14
Appendix A. Acknowledgements 15
Appendix B. Document History 15
Author's Address 16

1. Introduction

There are times when it is necessary to transport a private key, whether the private component to an asymmetric cipher key-pair or a symmetric cipher key used for encryption or generating a message authentication code (MAC), where the transport mechanism might not

provide adequate content protection for the key. For instance, end-to-end scenarios where the key holder and key recipient are linked through multiple network hops that might or might not employ transport layer security (TLS, [RFC5246]), or the key holder an key recipient (often the same human being) might exchange a private key using physical media such as a USB drive that itself is not encrypted.

This document specifies an approach that uses JavaScript Object Notation (JSON) Web Encryption [JWE] to encrypt a private key that is formatted as a JSON Web Key [JWK]. While [JWE] provides protection of symmetric keys, this key is itself intended for the protection of content, not as the content itself. Further, [JWE] does not itself provide protection of an asymmetric private key.

Ofttimes the transport of private keys involves direct interaction with human beings. In these scenarios the use of a human-understandable password or passphrase to protect the private key is desirable. Therefore, this document also specifies and registers JWK formats and JWE algorithms based on [RFC2898] to allow for protecting content using a password.

2. Terminology

This document inherits JSON Web Algorithms (JWA)-related terminology from [JWA], JSON Web Encryption (JWE)-related terminology from [JWE], JSON Web Key (JWK)-related terminology from [JWK], and password-based cryptography-related terminology from [RFC2898]. Security-related terms are to be understood in the sense defined in [RFC4949].

The capitalized 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].

3. Protecting Keys

The process for protecting private keys and symmetric keys are identical. The only differences are typically the algorithms used to protect the key.

To protect a private key, the key holder performs the following steps:

1. Converts the JWK object to a UTF-8 encoded string (K').

2. Performs the message encryption steps from [JWE] to generate the JWE header H, JWE Encrypted Key E, JWE Initialization Vector IV, JWE Ciphertext C, and JWE Integrity Value I, using the following inputs:

   * The 'alg' property set to the intended key encryption algorithm (e.g., "RSA-OAEP", or "PBES2-HS256+A256KW" from below).

   * Keying material appropriate for the selected key encryption algorithm (e.g., private key for "RSA-OAEP", or shared password, salt, and iteration count for "PBES2-HS256+A256KW").

   * The 'enc' property set to the intended content encryption algorithm (e.g., "A256GCM" or "A256CBC+HS512").

   * The 'cty' property set to "application/jwk+json", indicating the content is a JWK object.

   * Keying material appropriate for the selected content encryption algorithm (e.g., Content Encryption Key and Initialization Vector).

   * K' as the plaintext content to encrypt.

3. Serializes to the appropriate format for exchange, such as the Compact Serialization documented in [JWE].

3.1. Details for Private Keys

Private keys are typically protected using a symmetric key. This symmetric key can be exchanged or determined in various ways, such as deriving one from a user-supplied password; the algorithms "PBES2-HS256+A128KW" and "PBES2-HS256+A256KW" (defined in Section 6.2) enable this.

3.2. Details for Symmetric Keys

Symmetric keys are typically protected using public-private key pairs. It is assumed the key holder has the appropriate public key(s) for the key recipient(s).

The process defined herein expects JWK objects. While more compact to simply encrypt the symmetric key directly with a public key, using the complete JWE process on complete JWK objects allows additional properties to be protected (e.g., expected lifetime, acceptable uses) without exceeding the very restrictive plaintext length limits in most public-private key operations (e.g., 234 octects when using the "RSA-OAEP" algorithm with a 2048-bit key).

4. Private Key Example

NOTE: unless otherwise indicated, all line breaks are included for readability.

The key holder begins with the [JWK] representation of the private key (here using a [RFC3447] RSA private key, formatted per [JPSK]):

```
{
    "kty":"RSA",
    "kid":"juliet@capulet.lit",
    "n":"ALekPD1kotXZCY_YUz_ITWBZb2nTOw35VvZlnqTiYSeusO58qCtYDz
         ahTEkEcjtduRqfkxJKHYVq9Iro4x1cewXFdJZUuMOQAhoD63AHemXE
         kdPiKqJvkBXDT_Eo4NPOjMKKkFPy2MsJQBmdtVknUvzxEchhYjZ490
         EJTvGJ7OYwrSwkcCxy9D29XxL-OQLkSLlH1XD8kgVmJw8hsb42Bg0j
         PgKlkvcyENmYpYE_hqlJoqYNFzgtAnNtK4C3tspix46R3IgilQG2Of
         i99vpUnmTvjrOlNef2l65PRsPHD1Gl9fyPLCxrkolXbdwvxZ9j2d2f
         Iu-OBTxRhnBtarNls_k",
    "e":"AQAB",
    "d":"GRtbIQmhOZtyszfgKdg4u_N-R_mZGU_9k7JQ_jn1DnfTuMdSNprTea
         STyWfSNkuaAwnOEbIQVy1IQbWVV25NY3ybc_IhUJtfri7bAXYEReWa
         Cl3hdlPKXy9UvqPYGR0kIXTQRqns-dVJ7jahlI7LyckrpTmrM8dWBo
         4_PMaenNnPiQgO0xnuToxutRZJfJvG4Ox4ka3GORQd9CsCZ2vsUDms
         XOfUENOyMqADC6p1M3h33tsurY15k9qMSpG9OX_IJAXmxzAh_tWiZO
         wk2K4yxH9tS3Lq1yX8C1EWmeRDkK2ahecG85-oLKQt5VEpWHKmjOi_
         gJSdSgqcN96X52esAQ",
    "p":"ANq50jleISkjfLEuAoHEBxW7NPF26BQ6irpt7HOIdxkca05kHZdWSv
         bsPjyB30D9BZMV1a8flhPmRG66orx_9ogi1Eu8AJel7wEbdSpCGlMT
         z0mAfcpN9bNEPFCvehN_zqwAwGLQCbPjNycQi3zYKoeehw5xE00IR9
         6wk-U98icL",
    "q":"ANbv0YhQz-ywWIdzeBly0_TqUimD9LkGcommcAbTggTSYEMWo9dEVo
         7GbtHOiHnYrOEuwf3KEigdCo_T2j2gc4PiMkkb73ELj2pkLuuq4jIY
         1bRuk5VfAiwmCq2Jeds4qitBP8ptkJ5MLFF-3mEwey2wB0SvRqqHAx
         OQdH_NPCOL",
```

```
      "dp":"KkMTWqBUefVwZ2_Dbj1pPQqyHSHjj90L5x_MOzqYAJMcLMZtbUtwK
            qvVDq3tbEo3ZIcohbDtt6SbfmWzggabpQxNxuBpoOOf_a_HgMXK_l
            hqigI4y_kqS1wY52IwjUn5rgRrJ-yYo1h41KR-vz2pYhEAeYrhttW
            txVqLCRViD6c",
      "dq":"AvfS0-gRxvn0bwJoMSnFxYcK1WnuEjQFluMGfwGitQBWtfZ1Er7t1
            xDkbN9GQTB9yqpDoYaN06H7CFtrkxhJIBQaj6nkF5KKS3TQtQ5qCz
            kOkmxIe3KRbBymXxkb5qwUpX5ELD5xFc6FeiafWYY63TmmEAu_lRF
            COJ3xDea-ots",
      "qi":"AJUkIvsPQqclEXjBKz9UbAS5O8DbTr7OREKT6prjL6luezQVHM0nB
            KD8JlKqmm7vVdPj8uHUOe_22qaCkbtUfdG77hZ1Ot0h1hBYJWULyQ
            zHgL5o-LJvhadKGLv53qLYENIc2yOYK8u2o3WMvftpTcf--mgWaDl
            LvRwiflLH0jiP"
  }
```


The key holder uses the following [JWE] inputs:

JWE Header:


```
{
  "alg":"PBES2-HS256+A128KW",
  "jwk":{
    kty:"PBKDF2",
    kid:"27a4c46f-6d36-4a8c-814c-c954165f6dc9",
    s:"2WCTcJZ1Rvd_CJuJripQ1w",
    c:4096
  },
  "enc":"A128CBC+HS256",
  "cty":"application/jwk+json"
}
```


Password:

Thus from my lips, by yours, my sin is purged.

Content Master Key (encoded as base64url per [RFC4648]):


```
D0GoLoMS35BtD4_rSF56VGg_Syj0VG6-lb4xrpQIQmU
```


Initialization Vector (encoded as base64url per [RFC4648]):

XYqmb7uopcN1pCNRGJ5hKw

The key holder performs steps 1 and 2 to generate the [JWE] outputs (represented using the Compact Serialization):

eyJhbGciOiJQQkVTMi1IUzI1NitBMTI4S1ciLCJqd2siOnsia3R5IjoiUEJLREY
yIiwia2lkIjoiMjdhNGM0NmYtNmQzNi00YThjLTgxNGMtYzk1NDE2NWY2ZGM5Ii
wicyI6IjJXQ1RjSloxUnZkX0NKdUpyaXBRMXciLCJjIjo0MDk2fSwiZW5jIjoiQ
TEyOENCQytIUzI1NiIsImN0eSI6ImFwcGxpY2F0aW9uL2p3aytqc29uIn0.
b735tKJtEzS9VNxgEO6hT6WYZ9-zOpEIAFk3k0jIiCju7bPLb7FQKw.
XYqmb7uopcN1pCNRGJ5hKw.
dmqTOCIGwHdNOixUkmQ9H0g-JWU74ayVUeuSnsRnRdBPy0wRdBkZBsiQC6-Cl8q
QSjmC376EJvffG2xUBqjt4omuzMX9KY3Kn64GAHr528N5Bv5487fu-iHBy7uVvT
F0zgBaSV4Rt-44FWoMoE7H4vcQn_Lf7mv0dviciDM3Spp_IZrb5ufzDhrQlzArM
xOh7rBTgwoeOaFywXuFrxqr9GbV4Qzn7Vy78T8UUd5alr6GlfF-_O0hW37Gwju_
AT4bN6fs42NKYvqsAq90ZDujQjRlj3BJc1wAJbw9Ev7oxEvPvUSXgfDk6rvnB-n
uKD-0KU-M9td2QM8De0AXYRf2rTMiIIuNsRWeJxgeL97Unz9yNywAfcf4SX1P38
pgCZAbVwLRdbZwcOjK0_R3BQAtyysX-f4rtDCH9BKKFLB_YLcDkQn547QS2RMWf
GrVPT5CKp5Z5H8RSC47HEnmwppAKtGfUPb4wSs6zT8yV60RxOYD8Ze5DK9UJrPN
MFfN33_JlpeNKF7w9ulN57-ooYbXkX0WI4JjdF9G9NdJbh8F1NRqLc4KyQBW2bJ
S_SCZdeVZ4O_spCKfwKpIDFoXE9Nm-3o8mxhfdUbq_Ck8WqiJ6CKm-XjN1b7Z0f
lkGz6YkXdbd3-F62bB09VzsYERnSBIdsWtwaKMvSyqi8MkhMyhZe2-Iz481r4gi
v8ESWXAMeVihmOU9HLtgO4MMY3kSbB1qLzhbH7-CRh4h7k83tCmHPvNIQc-JYLm
80aHs_W_91SPRwnUZJHKasybepqika_CNwkmYsRkiV0GOpzrl2T28Nor74xPrBb
tk5LJMT_ZKErrCQoIvcgXrWcaTknCpe2sDYkOMuvNlsT8g6r45HuPJ6u561-sw7
wvam2P1AEg4wuQBAt7Y1_VDy6N-q71ZejayANTCtMGeJiWea79X6xdUJQ_py5xR
SuSjSwjsXCvisWyiKLAKXoVO9gQGEZLZMhYqRSGwip-KSkYpFYPd5ofn21MHXKG
D6r0gapo7lMysKlCpfd5v0_sB0JJYKsm12F49cvtK_CEtMYQw9n5R2wo8_2m5og
HGG3hMajGmem3anRAoSfifBBzx4kP_OOSqo_FoDbRzGluImVwcGL_pzCRRVNwAx
e5Bx8Al2xGLYncgs-QG7MRKu6LRB5pUq_ZbarL8JJengaa6AbxWsIMkTPEqilyi
SPpl7zmOFrUtuu-UUnNwhr6WEdLJm7o0iUoXr-Eyi8rfnZgdSvJOdMj-pGKQrWl
xyAo-Td8IqU-3DHk-otvjCd_i9SW0zRoL4GmqMkiJkIzZ7mjLdFLIFnX85sx4Qt
yYhMzEIfpgqnv66RnKVLyQ-sIap-9XO_mvSxsLL0yr8a4c_jufv0aFAbbLa_bo0
Mz_U309z0PmMp7BMh7CuwbiLhaoM6ZoafsxxVcOTHMbmEpybvsDOf8HPQ_k2kN2
qrVUfvYW5Be9ViOBNxKZWSiDDY0YWs5MhMZUvqnfq8amtAQNYTrpu5w2LfJIWhA
KkqzYAkzH7Jm7NFnOlcSrLPzFndjVZgIysYnBqkziTqtDoSNHCFY2TaJyZ3cT-o
WZQkVn07E8zuzMd6SGqPRAzY51CKbXdEfRaNgvaSb-V9TZYyhCmSHCGbwo0iErG
TfGiHtrfo4Jf6GD8-CcdmggWN-824rGOtp3Bgl8VAi_jmKkzF5s_sIEwhe7oa1H
S6PMYPkp9llZAiwmFCKHdQbfOdKXbD8FI7p7kUX8llOFLk3w12R2ffVR-gOm7qs
9MJjmi14nXmp13mV9YP_CgkNNss45B1DdcNMhtippHJ07CWIvKm1pkQOrsXG45C
4bNJ6YCn63X9ctdzhnFGmCJxCji3TasWWbnI4eA6XthWkJC5e5Nbz_2K-99PC9K
zwmauA98sqU1yKZFugSYOB6NRwN_y_GB1LEXDSE_FPRSEPNZNJyEMvKo5CeAtEj
7YPvFR6-yzWDTG0Uq1PafxITByg6UXHl9xBRborklCdfL3gUj3EoXHkvEsXdg22
jkpGZUmhWWlNvHeM5y0FUHZTIgyyJqHx_Y8v7yaZ881xwFaYAW52aSnL_8h68U1
8Sv7Q66FKi1gtOYU41FRW6i7oAC9xPYr1Jt5A-am4IwPPR-CPL071mGqOPrDd7l

gSCumoFESqi24d0IQuzPdEh643DHbWbeAQ7YB-LpZR_hTEC4IndRugQA.
3c4RF_muOYT02o5Klxv-IQ

5. Symmetric Key Example

NOTE: unless otherwise indicated, all line breaks are included for readability.

The key holder begins with the [JWK] representation of the symmetric key (here using a [AES] 128-bit key, formatted as per [JPSK]):

```
{
    "kty":"oct",
    "kid": "b8acba65-8af2-4e93-a8e0-d4abd7f25e52",
    "k": "fKrBr19_ne9Cp3akXGpqgA"
}
```

The key holder uses the following [JWE] inputs:

JWE Header:

```
{
  "alg":"RSA-OAEP",
  "jwk":{
    "kty":"RSA",
    "kid":"juliet@capulet.lit",
    "n":"ALekPD1kotXZCY_YUz_ITWBZb2nTOw35VvZlnqTiYSeusO58qCtYDz
         ahTEkEcjtduRqfkxJKHYVq9Iro4x1cewXFdJZUuMOQAhoD63AHemXE
         kdPiKqJvkBXDT_Eo4NPOjMKKkFPy2MsJQBmdtVknUvzxEchhYjZ490
         EJTvGJ7OYwrSwkcCxy9D29XxL-OQLkSLlH1XD8kgVmJw8hsb42Bg0j
         PgKlkvcyENmYpYE_hqlJoqYNFzgtAnNtK4C3tspix46R3IgilQG2Of
         i99vpUnmTvjrOlNef2l65PRsPHD1Gl9fyPLCxrkolXbdwvxZ9j2d2f
         Iu-OBTxRhnBtarNls_k",
    "e":"AQAB"
  },
  "enc":"A128CBC+HS256",
  "cty":"application/jwk+json"
}
```

Content Master Key (encoded as base64url per [RFC4648]):

QkWU4j0bOc_meVgxNYoad74fQAosvz-4rnKqAhHEV-c

Initialization Vector (encoded as base64url per [RFC4648]):

VMmZ6nLXHkcOUmBTlZaSsQ

The key holder performs steps 1 and 2 to generate the [JWE] outputs (represented using the Compact Serialization):

eyJhbGciOiJSU0EtT0FFUCIsImp3ayI6eyJrdHkiOiJSU0EiLCJraWQiOiJqdWx
pZXRAY2FwdWxldC5saXQiLCJuIjoiQUxla1BEMWtvdFhaQ1lfWVV6X0lUV0JaYj
JuVE93MzVWdlpsbnFUaVlTZXVzTzU4cUN0WUR6YWhURWtFY2p0ZHVScWZreEpLS
FlWcTlJcm80eDFjZXdYRmRKWlV1TU9RQWhvRDYzQUhlbVhFa2RQaUtxSnZrQlhE
VF9FbzROUE9qTUtLa0ZQeTJNc0pRQm1kdFZrblV2enhFY2hoWWpaNDkwRUpUdkd
KN09Zd3JTd2tjQ3h5OUQyOVh4TC1PUUxrU0xsSDFYRDhrZ1ZtSnc4aHNiNDJCZz
BqUGdLbGt2Y3lFTm1ZcFlFX2hxbEpvcVlORnpndEFuTnRLNEMzdHNwaXg0NlIzS
WdpbFFHMk9maTk5dnBVbm1UdmpyT2xOZWYybDY1UFJzUEhEMUdsOWZ5UExDeHJr
b2xYYmR3dnhaOWoyZDJmSXUtT0JUeFJobkJ0YXJOb HNfayIsImUiOiJBUUFCIn0
sImVuYyI6IkExMjhDQkMrSFMyNTYiLCJjdHkiOiJhcHBsaWNhdGlvbi9qd2sran
NvbiJ9.
ReivAR0RfDi-03K9Db3gC3MSJQJvCe378Anzg0vKj45DJGwfEaPFym_tt6HkbgB
vgIBaFX_WZE1E3xXMngH_oBz-zUJzB9Gc_hAeov6uLz0pp4knb20pOZCls0Lcjs
xqgAF_RwB7l_mcPP3HVAwfoEz-_Um7FOztq5Wjse1fBmEX0fwqJT3VC7HVKzJpo
pJgrrsYFyGPlraNBJJ3yvmRMYLOzTLNoNDYqQz89yZ_dYDcN7zjrke8T3NnSwx2
9xF3kwiD_AO2SUsA23Zw3xEFQoiskK0w54KKa75yFlSbnObFLOOvqncxJy0bbha
GqW6I-jeoXVaG7aia6hGU9aMX2g.
VMmZ6nLXHkcOUmBTlZaSsQ.
N3j7CW5JfJj7C6uL9PCVIm4U_NWRtAVjrnqnPRXIwhepaGoL-TQHeMyHveg5Uyg
rPP_PBwk-VkwAyFBJClPNJ6cGSS_VN5a9Z60rxlXEQi8nBhCgQzA3wU1XMTHCs-
QF.
trBdLTmkE2mIPdA7eefNyQ

6. Using Password-Based Cryptography

There are often times when a key is exchanged through immediate human interaction. To help facilitate such exchanges, a number of password-based cryptography schemes utilizing [RFC2898] are defined to supplement the key format and encryption algorithms from [JWA].

6.1. PBKDF2 Key Type

The "PBKDF2" key type is used to contain the parameters necessary to derive a cipher key from a password using the PBKDF2 algorithm from [RFC2898]. The following parameters are defined:

6.1.1. 's' Parameter

The REQUIRED "s" parameter contains the PBKDF2 salt value (S), as a base64url encoded string (per [RFC4648]). This value MUST NOT be the empty string "".

The salt expands the possible keys that can be derived from a given password. [RFC2898] originally recommended a minimum salt length of 8 octets (since there is no concern here of a derived key being re-used for different purposes). The salt MUST be generated randomly; see [RFC4086] for considerations on generating random values.

6.1.2. 'c' Parameter

The REQUIRED "c" parameter contains the PBKDF2 iteration count (c), as an integer. This value MUST NOT be less than 1, as per [RFC2898].

The iteration count adds computational expense, ideally compounded by the possible range of keys introduced by the salt. [RFC2898] originally recommended a minimum iteration count of 1000.

6.1.3. 'hint' Parameter

The OPTIONAL "hint" parameter contains a description clue to the password, as a string. If present, this value SHOULD NOT be the empty string "".

The hint is typically displayed to the user as a reminder or mnemonic for the actual password used. This parameter MUST NOT contain the actual password, and implementations MAY use various heuristic algorithms to prohibit hints that are alternate forms of the actual password.

6.2. PBES2 Key Encryption Algorithms

The "PBES2-HS256+A128KW" and "PBES2-HS256+A256KW" algorithms defined below are used to encrypt a JWE Content Master Key using a user-supplied password to derive the key encryption key. With these algorithms, the derived key is used to encrypt the JWE Content Master Key. These algorithms combine a key derivation function with an encryption scheme to encrypt the JWE Content Master Key according to PBES2 from section 6.2 of [RFC2898].

6.2.1. PBES2-HS256+A128KW

The "PBES2-HS256+A128KW" algorithm uses "HMAC-SHA256" as the PRF and "AES128-WRAP" as defined in [RFC3394] for the encryption scheme. The salt (S) and iteration count (c) MUST be specified by the "s" and "c" parameters (respectively) in the applicable "PBKDF2" JWK object. The derived-key length (dkLen) is 16 octets.

6.2.2. PBES2-HS256+A256KW

The "PBES2-HS256+A256KW" algorithm uses "HMAC-SHA256" as the PRF "and "AES256-WRAP" as defined in [RFC3394] for the encryption scheme. The salt (S) and iteration count (c) MUST be specified by the "s" and "c" parameters (respectively) in the applicable "PBKDF2" JWK object. The derived-key length (dkLen) is 32 octets.

7. IANA Considerations

7.1. JSON Web Key Types Registration

This document registers the following to the JSON Web Key Types registry:

- "kty" Paramater value: "PBKDF2"
- Implementation Requirements: OPTIONAL
- Change Controller: IETF
- Specification Document(s): Section 6.1 of [[this document]]

7.2. JSON Web Key Parameters Registration

This document registers the following to the JSON Web Key Parameters registry:

- Parameter Name: "s"
- Change Controller: IETF
- Specification Document(s): Section 6.1.1 of [[this document]]

- Parameter Name: "c"
- Change Controller: IETF
- Specification Document(s): Section 6.1.2 of [[this document]]
- Parameter Name: "hint"
- Change Controller: IETF
- Specification Document(s): Section 6.1.3 of [[this document]]

7.3. JSON Web Encryption Algorithms

This document registers the following to the JSON Web Encryption Algorithms registry:

- Algorithm Name: "PBES2-HS256+A128KW"
- Algorithm Usage Location(s): "alg"
- Implementation Requirements: OPTIONAL
- Change Controller: IETF
- Specification Document(s): Section 6.2.1 of [[this document]]
- Algorithm Name: "PBES2-HS256+A256KW"
- Algorithm Usage Location(s): "alg"
- Implementation Requirements: OPTIONAL
- Change Controller: IETF

o Specification Document(s): Section 6.2.2 of [[this document]]

8. Security Considerations

8.1. Re-using Keying Material

It is NOT RECOMMENDED to re-use the same keying material (Key Encryption Key, Content Master Key, Initialization Vector, etc) to protect multiple JWK objects, or to protect the same JWK object multiple times. One suggestion for preventing re-use is to always generate a new set keying material for each protection operation, based on the considerations noted in this document as well as from [RFC4086].

8.2. Password Considerations

While convenient for end users, passwords are vulnerable to a number of attacks. To help mitigate some of these limitations, this document applies principles from [RFC2898] to derive cryptographic keys from user-supplied passwords.

However, the strength of the password still has a significant impact. A high-entry password has greater resistance to dictionary attacks. [NIST-800-63-1] contains guidelines for estimating password entropy, which can help applications and users generate stronger passwords.

An ideal password is one that is as large (or larger) than the derived key length but less than the PRF's block size. Passwords larger than the PRF's block size are first hashed, which reduces an attacker's effective search space to the length of the hash algorithm (32 octects for HMAC-SHA-256). It is RECOMMENDED that the password be no longer than 64 octets long; for "PBES2-HS256+A256KW".

Still, care needs to be taken in where and how password-based encryption is used. Such algorithms MUST NOT be used where the attacker can make an indefinite number of attempts to circumvent the protection.

9. Internationalization Considerations

Passwords obtained from users are likely to require preparation and normalization to account for differences of octet sequences generated by different input devices, locales, etc. It is RECOMMENDED for applications to perform the steps outlined in [SASLPREP] to prepare a password supplied directly by a user before performing key derivation and encryption.

10. References

10.1. Normative References

[JWA] Jones, M., "JSON Web Algorithms (JWA)", draft-ietf-jose-json-web-algorithms-08 (work in progress), December 2012.

[JWE] Jones, M., Rescola, E., and J. Hildebrand, "JSON Web Encryption (JWE)", draft-ietf-jose-json-web-encryption-08 (work in progress), December 2012.

[JWK] Jones, M., "JSON Web Key (JWK)", draft-ietf-jose-json-web-key-08 (work in progress), December 2012.

[JPSK] Jones, M., "JSON Private and Symmetric Key", draft-jones-jose-json-private-and-symmetric-key-00 (work in progress), December 2012.

[SASLPREP] Saint-Andre, P., "Preparation and Comparison of Internationalized Strings Representing Simple User Names and Passwords", draft-melnikov-precis-saslprepbis-04 (work in progress), September 2012.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2898] Kaliski, B., "Password-Based Cryptography Specification", RFC 2898, September 2000.

[RFC3394] Schaad, J. and R. Housley, "Advanced Encryption Standard (AES) Key Wrap Algorithm", RFC 3394, September 2002.

[RFC4086] Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", RFC 4086, June 2005.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", RFC 4949, August 2007.

10.2. Informative References

[AES] National Institute of Standards and Technology (NIST), "Advanced Encryption Standard (AES)", FIPS PUB 197, November 2001.

[NIST-800-63-1]
National Institute of Standards and Technology (NIST), "Electronic Authentication Guideline", NIST 800-63-1, December 2011.

[RFC3447] Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 2898, February 2003.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

Appendix A. Acknowledgements

Appendix B. Document History

-02

* Incorporated changes suggested at the JOSE interim meeting on 2012-04-28:

  + Replaced JWE key encryption algorithm "PBES2-HS512+A256KW" with "PBES2-HS256+A256KW".

  + Added considerations for password-based encryption algorithms around dictionary and brute force attacks.

* Updated to latest versions of JOSE dependencies.

-01 Incorporated changes suggested by Jim Schaad:

* Expanded the acronym "JSON" on first use.

* Expanded the introduction to explain how this document's protection of symmetric keys differs from [JWE].

* Expanded the introduction to better explain why password-based encryption algorithms are needed.

* Moved information on PBKDF2 salt from the security considerations to the "s" JWK parameter definition.

* Moved information on PBKDF2 iteration count from security considerations to the "c" JWK parameter definition.

* Added the "hint" JWK parameter.

* Explicitly noted what registries are updated by the IANA considerations.

* Relaxed language around re-use of keying material.

* Removed section discussing protected key lifetimes.

* Improved recommendations around password lengths.

-00 Initial revision

Author's Address

Matthew Miller
Cisco Systems, Inc.
1899 Wynkoop Street, Suite 600
Denver, CO 80202
USA

Phone: +1-303-308-3204
Email: mamille2@cisco.com