JOSE Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 05, 2013

M. Miller
Cisco Systems, Inc.

February 01, 2013

Using JSON Web Encryption (JWE) for Protecting JSON Web Key (JWK) Objects
draft-miller-jose-jwe-protected-jwk-00

Abstract

This document specifies an approach to protecting JSON Web Key (JWK) objects using JSON Web Encryption (JWE). This document also specifies a set of algorithms for encrypting 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 August 05, 2013.

Table of Contents

1. Introduction

There are times when it is necessary to transport or exchange a private or symmetric key, but 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]), nor might exchange of a key using physical media such as a USB drive that itself is not encrypted.

This document specifies an approach that uses [JWE] to encrypt a private or symmetric key that is formatted as [JWK]. This document also specifies and registers JSON Web Key formats and JSON Web Encryption algorithms based on [RFC2898] that allow for protecting keys 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 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-HS512+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-HS512+A256KW").

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

* 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-HS512+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 input limits inherent in most public-private key operations.

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":"2a43135aa05479f570676fc36e3d693d0ab21d21e38fdd0be71fcc
          3b3a9800931c2cc66d6d4b702aabd50eaded6c4a3764872885b0ed
          b7a49b7e65b382069ba50c4dc6e069a0e39ffdafc780c5cafe586a
          8a0238cbf92a4b5c18e762308d49f9ae046b27ec98a35878d4a47e
          bf3da9621100798ae1b6d5adc55a8b0915620fa7",
     "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",
  "enc":"A128CBC+HS256",
  "jwk":{
    "kty":"PBKDF2",
    "s":"2WCTcJZ1Rvd_CJuJripQ1w",
    "c":4096
  }
}
```

Password:

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

Content Master Key (encoded as base64url per [RFC4686]):

1ICvnpc3zPRNS7JoJ9bnJ929eX7EnRwmc0CHNOF1zKc

Initialization Vector (encoded as base64url per [RFC4686]):

B9BVK3hIsEu9zUOWjKeOSg

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

```
eyJhbGciOiJQQkVTMi1IUzI1NitBMTI4S1ciLCJlbmMiOiJBMTI4Q0JDK0hTMjU
2IiwiandrIjp7Imt0eSI6IlBCS0RGMiIsInNhbHQiOiIyV0NUY0paMVJ2ZF9DSn
VKcmlwUTF3IiwiaXRycyI6NDA5Nn19.
dO4VTHV1JJnnPxbc9xswMA6ezNLCQ1Nq0Bnt4l2hjzxyXfbgM3w-cQ.
B9BVK3hIsEu9zUOWjKeOSg.
mUt0YpU0Gsfis7tK70aggz5Qb5J6oppjl7aSn_S4DIHJkSF-Gd9KEu4XF0vbMc1
kvovNOmnahDLoJIq7hoB666zt1yp4umZKuzhOQ5O3jwOwC4rUf1W3hDM5p2nYZq
LFqDBme-Z8KVwosfBj2TkZSnFWC6cqMy6d7K6egWz7uct39ZH6_FAUrr0JiMMnj
0jxTEBP6apiLD-GJutxITnWISmGmX0jOWyTKRh-F-I3lJ3pXE0_4vWR8-gXaFGW
CgItofAOsdKKoxiSJvXPVxK-oqPFPlS0H5OLHw9tkjmPVZ0Ot-RLgoMPQJ0Yi47
U0DBh_2-IQBcpPvwBnUzl8KcrLFh-NpTSoC_zIn6xos5r4JVXf5v1V79MKGvk3-
evdQoEKtS9LnimKTP8YmeqAnU1Y3VoUkgeRTZikKhQ_kwpAxakaRuBzfaIOKjqY
X5J_jSJuuBQU_4fgHPkvrq1zJ3TnUAWZIqFcw27dKGMxXLMlz_sbW2YerZ0BULc
xbe8bJpbHkE64r7Nc86iZyL6e-8htkUvpGmdQGqno-q723Ry-u2Iz1-exjgjbNu
```

z4tbSrf49Y0_VSFEuHLnCRChVQE8cgupCy8GioEpuC7v1hMNE7JZTVbG3Ud3QzZ
Webr0sAYjWwqa2ySoJ7AaRfIiVgMdTAS2_gBdbYRSYI2pkQTHpyZ8D_lp5RvYFR
FNsHWoEToXeL1P_csuKaGsikKbzgXdevrjmCJTOpByGYO5cb2O8xFyYy1VX_pM6
bxqj5ai8-jyiCoDKvF_40wT_9F6H79UIZtuc3L7r7OxKrmY32LLvfofVRsfdIcK
AhMvnvOCpzRN3sHiR7iAx1bvaIaz40Mshf8z-__70GEqr5oWxRwyv2uhCiPHtV6
lBIhCqki6th317-n_lR58LHw894ja0xbOOxUJfZziaG14O9tqjetTSB0JGKXC4S
ry6zzeyqDwsSpY439Uqn7va_V5ELLfJf9S7fh0AtZS3ogpw4oT-6LbywFPf0jH9
LgNmatORG_J2wgu8R8QpKogo_WTQK7izHVtEMe1vQQElAy8P2SUjOJiOU6wAW4o
neU7HUUwD3cvFSn-ykze48FurI5f6DXyPNTVtzaEjBnVVP517bGb_3V_-1SCBdZ
1f_sYWJ0S8r2u5ccy28CP_7MhZOY8r4oJJn2NmT_jDOllCNtJLKVdKon0QslEER
GXa8r6Jtp3UbfLVjZQbnFl2mBygH4PoWbyGVxUNDE66mZVKM4Y-6ng5L84PrDt5
aAQ0HREqjvX4bsQgWyXKq-HRe5DXO2qdoPAIoPGhqOlZz70iYHO5lkX_Oso3eBv
DCDVbkD7Gr-LZJusrVtA6Hf82YpUUldjyPsD985T2hdcSQMoxol8aWNDL52T4Eo
FTTLFNG0Xy3qvS79ScgMOSIOE8k_DgzyBrcGJibeeaVKGdbqUQRADaeWO2QM3Rq
9TOyizLNyyjkJBgecWtIfLAWSTPx7n39zuBhGb5oChAFlR1HqrV76KO22rj08ra
J_zi8YQpp4qTTXmYwY5sjftE8IDDcLxk6ovvDM7262onZ5CP3GKp09s9dTJZi2L
tmZ8NAHlJ5Q9v9XMHi0tvLKKlahdcvgnG3sC_gL46osnNE3gBipLe_IQNl0X0zU
cakwzYtC02JeBbNFQSduY4OBiAAhD2VhyhLMNvGyt9QjntCru1z_gkr1qZR4y5W
Uec9WtxPqqC-dny-MZOINZd13LdStVY_g0KaAJET0etT4vYZJ6IfqdEXRSKKQKN
CJpencNhJH99NEumPwY0J8hYGC-KkGb6ZQ0yeQDIpcH2reMS9_DU0JaFtd5BpKm
pNDAvmekzWv65o6MCzLVN4Wlf4tZ6a7FnLxDg0jQBsPGh3ZKEeGmvxJKSvGK_Or
5-gXJdMyj1n0bWyex-ZTmtbR3OHKgAlFQUdLuB-47UFS0HtF0Lo1ENPZUoxwFcg
Uu2Yr30O8vhHLhkRAyAvrFPBHhSqcGr954FeMhMo18dKu3zdykyueIQpLIpU7qN
btXXSOu1c1rPqBMm3h77iLmNzSByFER32-W7hQwx9-Ikbwy52DaZACpXdyzZOQ2
PSYEJeuy8CyHap8_Dwucb3CQiqBi-iz25Mmwx8qBjWq3K31W4WeF0heC_rCe6LS
vOCuHJXpu8Wqa1LeyTbuHyQawH8Aev-MzVLXp0E-WtggfNi-mqcThs25993uveI
TpmK6lyQu6SX31t6MlQKI_29e46cYRMFhsEZSGM--acnZiryvi39IKsVNtOf33E
9zlKntRdmMoROenR-q5--9nObKaF81k3Xb-yWNTlQh1ot59rWAeqpPv0uA.
big8BFT-dXJqoKGScpLk0ssjuyWn7fxDvJc7AN7ONPw

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",
  "enc":"A128CBC+HS256",
  "jwk":{
    "kty":"RSA",
    "n":"ALekPD1kotXZCY_YUz_ITWBZb2nTOw35VvZlnqTiYSeusO58qCtYDz
         ahTEkEcjtduRqfkxJKHYVq9Iro4x1cewXFdJZUuMOQAhoD63AHemXE
         kdPiKqJvkBXDT_Eo4NPOjMKKkFPy2MsJQBmdtVknUvzxEchhYjZ490
         EJTvGJ7OYwrSwkcCxy9D29XxL-OQLkSLlH1XD8kgVmJw8hsb42Bg0j
         PgKlkvcyENmYpYE_hqlJoqYNFzgtAnNtK4C3tspix46R3IgilQG2Of
         i99vpUnmTvjrOlNef2l65PRsPHD1Gl9fyPLCxrkolXbdwvxZ9j2d2f
         Iu-OBTxRhnBtarNls_k",
    "e":"AQAB"
  }
}
```

Content Master Key (encoded as base64url per [RFC4686]):

```
ci5IlLIHnFLn-1lhL5CWOS3DdbXGU-BPuFCrLspXAkA
```

Initialization Vector (encoded as base64url per [RFC4686]):

```
_dYbckd_xuJUBUNsxkT9yw
```

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

```
eyJhbGciOiJSU0EtT0FFUCIsImVuYyI6IkExMjhDQkMrSFMyNTYiLCJqd2siOns
ia3R5IjoiUlNBIiwibiI6IkFMZWtQRDFrb3RYWkNZX1lVel9JVFdCWmIyblRPdz
M1VnZabG5xVGlZU2V1c081OHFDdFlEemFoVEVrRWNqdGR1UnFma3hKS0hZVnE5S
XJvNHgxY2V3WEZkSlpVdU1PUUFob0Q2M0FIZW1YRWtkUGlLcUp2a0JYRFRfRW80
TlBPak1LS2tGUHkyTXNKUUJtZHRWa25Vdnp4RWNoaFlqWjQ5MEVKVHZHSjdPWXd
yU3drY0N4eTlEMjlYeEwtT1FMa1NMbEgxWEQ4a2dWbUp3OGhzYjQyQmcwalBnS2
xrdmN5RU5tWXBZRV9ocWxKb3FZTkZ6Z3RBbk50SzRDM3RzcGl4NDZSM0lnaWxRR
zJPZmk5OXZwVW5tVHZqck9sTmVmMmw2NVBSc1BIRDFHbDlmeVBMQ3hya29sWGJk
d3Z4WjlqMmQyZkl1LU9CVHhSaG5CdGFyTmxzX2siLCJlIjoiQVFBQiJ9fQ.
UW26BTjbDGjgQ288JXJpGGpbZVV3WyZYJdm1Fvux8wM_t7bh6UbjbOdYXjL3rzD
phZteaD-DXiKrdGaVndTMLUmjg78H0R9K2WHxWNhTWiq8iMDGtw6BYnHW_r5DSf
vhojcDhCCBKWyO-QLXvTO1mfgo6UQwYY32Lvz-IrbxphrfBePsEXVYm75PLIIft
```

PnRwN3fWDMkKLCq2AesvWDuRWtKVjsh7uu_AX2ky6dvov2Lw_xMiDshsNz62RES
bNUW-ylMqNNn_VAYWKHK83k5CZppPlBb_8MDEUBPyGe_NDCa3118eo20tJGMn8v
XRg8k7D0nJ051hGlzAkMQ0S_x9g.
_dYbckd_xuJUBUNsxkT9yw.
QHkwptAlO0Gdo5BrbkaGPSek6wASr-twP8ZQ3YYHOkbRMedtsM0QwpvdnfUXoJL
7HZ0fbI-5YwO3PHvTOMXKx-K26cA-Hn9aseBvuICL2getFnrUcXs3vQI416PCw7
n39HtFLozCaBhJBaMv5x41CT6eaeRCMXE9Kbgz8lFicKNS9GYB7bUQfPXJl-qP7
05v-qkY4m2BAfofWtyGK-bnbs2ZKhKgaRTdRZlbnZTC19eSS8bPlalSNBrpr2eP
kDGDsYIKJvtbqD_xHeb7u5BT3ELmDl4CDjMruB3RiNwINcwNkCAhkg9tgV_vFCm
Qzsc9DH6dlnkYrbnyFrAD4Qe7xI1anDjLmRvU2cBB68-AtRd3PmpVzbr4t2eo33
GX9c_6fUILJS8WuExPs7KJkOuqDfKCPWDwRfX-yf8YUkrBZMj9aImyY2MyMJZFu
ITxseXbhkQFPJn-A4scSt1suWEXch5fKfl1KfxJlMio-Bm8kfUba4mgOKen7cJ0
whFSh7cupuSG0vVMLeBoBVGnndjARUxl5I57NZupH7xq3L8lDVHu_mQ_8Ae1j2h
QmPOA7_882j1ubpM9Kq-YmOPP4R8wk9SD0bl0k222mSMUMMEZoiSVk7OSHPHkXj
v-IkXkMmKT0lZiRohdKanb-Dfa1GWXeFTKFSKKJUQibLQ2zy6jlBptIEkxQNrJe
F-96Ie08r93r79ZRs3lNegzhI6ShMot1YBiG5C-WiOvPw6zysh_-AIGbuvZ9hBn
nuNfeGr2VPPcO5Z0FKGYGJ0DDnwlNHkdztEj_xLzSZjMxeO6o2hVJScLzzyG_Fy
T2v2kUHv_qeal18x-ICLR3yyF6JzoT0zKYPouBEIM5EZNAX9A3UjG0kNWW4Lou9
dlbLIZm-T7H6kj89RATMbDCwHa-nsF2kmnTRR6djt7lzDU1CHEEcM83ZWxi9qTR
cN_YrCokMkSjZYI87GqmnYHFsTktBBMNDX0u9SBdDZSwmz4EPsn7blYFt6Oiyd-
EBJlE_ocI4desERjIj0vKsgCoa9ea46S4m_gPa8xlCHK9wcsXU0ZhH8Lo_PyLm0
4CnjAbhKV0g8WvupjRQijkA6WoqYxQCP9EsDWH7kHJxDWV0uXXFoPZkcW5DBWrG
1NUC4TAjjI-9qhW7bpBZTLZHDJ99Awsh5lBqPYwUoPMS-lx-FPv9pAEih0rdHdH
__PelgmXKSwsoBSXzS_ErzePKwTfMAqedpe4WVykQ1rOTRn3uW01sv5gfRBRpo.
PnWwSPZ6TWOHKQwCIIt1HcCLhXof1j4frx2IcLXjIQ0

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 [RFC4686]). This value MUST NOT be the empty string "".

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].

6.2. PBES2 Key Encryption Algorithms

The "PBES2-HS256+A128KW" and "PBES2-HS512+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-HS512+A256KW

The "PBES2-HS512+A256KW" algorithm uses "HMAC-SHA512" 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

- "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

- 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]]

7.3. JSON Web Encryption Algorithms

- 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-HS512+A256KW"
- Algorithm Usage Location(s): "alg"
- Implementation Requirements: OPTIONAL
- Change Controller: IETF
- 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. When using "PBES2-HS256+A128KW" or "PBES2-HS512+A256KW", implementations MUST NOT use the same Key Encryption Key for a given password, and SHOULD take steps to prevent the same Key Encryption Key being used by different passwords when possible.

8.2. Lifetime of Protected Keys

Depending on the application, a protected JWK could potentially be stored for an indefinite time, anywhere from milliseconds (e.g., broadcasted over a computer network) to years (e.g., stored as a file).

8.3. Password Considerations

While convenient for end users, passwords have many drawbacks. To help mitigate these limitations, this document applies principles from [RFC2898] to derive cryptographic keys from user-supplied passwords.

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.

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.

An ideal password is one that is as large (or larger) than the derived key length but less than the PRF's block size. For "PBES2-HS256+A128KW", the ideal password is between 16 and 64 octets long; for "PBES2-HS512+A256KW", the ideal password is between 32 and 128 octets long.

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 user-supplied password 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.

[RFC4686] 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.

[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

-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