Network Working Group
Internet-Draft
Intended status: Informational
Expires: February 28, 2019

P. Hallam-Baker
Comodo Group Inc.

August 27, 2018

Data At Rest Encryption Part 1: DARE Message Syntax
draft-hallambaker-dare-message-02

Abstract

This document describes the Data At Rest Encryption (DARE) message syntax. This syntax is used to digitally sign, digest, authenticate, or encrypt arbitrary message content.

This document is also available online at http://mathmesh.com/Documents/draft-hallambaker-dare-message.html [1] .

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 https://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 February 28, 2019.

Table of Contents

1. Introduction

This document describes the Data At Rest Encryption (DARE) Message Syntax. This syntax is used to digitally sign, digest, authenticate, or encrypt arbitrary message content.

The DARE Message Syntax is based on a subset of the JSON Web Signature [RFC7515] and JSON Web Encryption [RFC7516] standards and shares many fields and semantics. The processing model and data structures have been streamlined to remove alternative means of specifying the same content.

A DARE Message consists of a Header, Payload and an optional Trailer. To enable single pass encoding and decoding, the Header contains all the information required to perform cryptographic processing of the Payload and authentication data (digest, MAC, signature values) may be deferred to the Trailer section.

The DARE Message Syntax is designed to compliment the DARE Container syntax. A DARE Container is an append-only log format consisting of a sequence of frames. Cryptographic enhancements (signature, encryption) may be applied to individual frames or to sets of frames. Thus, a single key exchange may be used to provide a master key to encrypt multiple frames and a single signature may be used to authenticate all the frames in the container up to and including the frame in which the signature is presented.

The DARE Message syntax may be used either as a standalone cryptographic message syntax or as a means of presenting a single DARE Container frame together with the complete cryptographic context required to verify the contents and decrypt them.

1.1. Encryption and Integrity

An important innovation in the DARE Message Syntax is the separation of key exchange and data encryption operations so that a Master Key (MK) established in a single exchange to be applied to multiple octet sequences. This means that a public key operation may be used to

encrypt multiple parts of the same message or to multiple frames in a DARE Container.

To avoid reuse of the key and to avoid the need to communicate separate IVs, each octet sequence is encrypted under a different encryption key (and IV if required) derived from the Master Key by means of a salt that is unique for each octet sequence that is encrypted. The same approach is used to generate keys for calculating a MAC over the octet sequence if required. This approach allows encryption and integrity protections to be applied to the message payload, to header or trailer fields or to application defined Enhanced Data Sequences in the header or trailer.

1.1.1. Key Exchange

Traditional cryptographic containers describe the application of a single key exchange to encryption of a single octet sequence. Examples include PCKS#7/CMS [RFC2315] , OpenPGP [RFC4880] and JSON Web Encryption [RFC7516] .

To encrypt a message using RSA, the encoder first generates a random encryption key and initialization vector (IV). The encryption key is encrypted under the public key of each recipient to create a per-recipient decryption entry. The encryption key, plaintext and IV are used to generate the ciphertext (figure 1).

[[This figure is not viewable in this format. The figure is available at http://mathmesh.com/Documents/draft-hallambaker-dare-message.html [2].]]

Monolithic Key Exchange and Encrypt

This approach is adequate for the task of encrypting a single octet stream. It is less than satisfactory when encrypting multiple octet streams or very long streams for which a rekeying operation is desirable.

In the DARE approach, key exchange and key derivation are separate operations and keys MAY be derived for encryption or integrity purposes or both. A single key exchange MAY be used to derive keys to apply encryption and integrity enhancements to multiple data sequences.

The DARE key exchange begins with the same key exchange used to produce the CEK in JWE but instead of using the CEK to encipher data directly, it is used as one of the inputs to a Key Derivation Function (KDF) that is used to derive parameters for each block of

data to be encrypted. To avoid the need to introduce additional terminology, the term 'CEK' is still used to describe the output of the key agreement algorithm (including key unwrapping if required) but it is more appropriately described as a Master Key (figure 2).

[[This figure is not viewable in this format. The figure is available at http://mathmesh.com/Documents/draft-hallambaker-dare-message.html [3].]]

Exchange of Master Key

A Master Key may be used to encrypt any number of data items. Each data item is encrypted under a different encryption key and IV (if required). This data is derived from the Master Key using the HKDF function [RFC5869] using a different salt for each data item and separate info tags for each cryptographic function (figure 3).

[[This figure is not viewable in this format. The figure is available at http://mathmesh.com/Documents/draft-hallambaker-dare-message.html [4].]]

Data item encryption under Master Key and per-item salt.

This approach to encryption offers considerably greater flexibility allowing the same format for data item encryption to be applied at the transport, message or field level.

1.1.2. Data Erasure

Each encrypted DARE Message specifies a unique Master Salt value of at least 128 bits which is used to derive the salt values used to derive cryptographic keys for the message payload and annotations.

Erasure of the Master Salt value MAY be used to effectively render the message payload and annotations undecipherable without altering the message payload data. The work factor for decryption will be O(2^128) even if the decryption key is compromised.

1.2. Signature

As with encryption, DARE Message signatures MAY be applied to an individual message or a sequence of messages.

1.2.1. Signing Individual Plaintext Messages

When an individual plaintext message is signed, the digest value used to create the signature is calculated over the binary value of the payload data. That is, the value of the payload before the encoding (Base-64, JSON-B) is applied.

1.2.2. Signing Individual Encrypted Messages

When an individual plaintext message is signed, the digest value used to create the signature is calculated over the binary value of the payload data. That is, the value of the payload after encryption but before the encoding (Base-64, JSON-B) is applied.

Use of signing and encryption in combination presents the risk of subtle attacks depending on the order in which signing and encryption take place [Davis2001] .

Na?ve approaches in which a message is encrypted and then signed present the possibility of a surreptitious forwarding attack. For example, Alice signs a message and sends it to Mallet who then strips off Alice's signature and sends the message to Bob.

Na?ve approaches in which a message is signed and then encrypted present the possibility of an attacker claiming authorship of a ciphertext. For example, Alice encrypts a ciphertext for Bob and then signs it. Mallet then intercepts the message and sends it to Bob.

While neither attack is a concern in all applications, both attacks pose potential hazards for the unwary and require close inspection of application protocol design to avoid exploitation.

To prevent these attacks, each signature on a message that is signed and encrypted MUST include a witness value that is calculated by applying a MAC function to the signature value as described in section XXX.

1.2.3. Signing Sequences of Messages

To sign multiple messages with a single signature, we first construct a Merkle tree of the message payload digest values and then sign the root of the Merkle tree.

[This is not yet implemented but will be soon.]

2. Definitions

2.1. Related Specifications

The DARE message format is based on the following existing standards and specifications.

Object serialization The JSON-B [draft-hallambaker-jsonbcd] encoding is used for object serialization. This encoding is an extension of the JavaScript Object Notation (JSON) [RFC7159] .

Message syntax The cryptographic processing model is based on JSON Web Signature (JWS) [RFC7515] , JSON Web Encryption (JWE) [RFC7516] and JSON Web Key (JWK) [RFC7517] .

Cryptographic primitives. The HMAC-based Extract-and-Expand Key Derivation Function [RFC5869] and Advanced Encryption Standard (AES) Key Wrap with Padding Algorithm [RFC3394] are used.

The Uniform Data Fingerprint method of presenting data digests is used for key identifiers and other purposes [draft-hallambaker-udf] .

Cryptographic algorithms The cryptographic algorithms and identifiers described in JSON Web Algorithms (JWA) [RFC7518] are used together with additional algorithms as defined in the JSON Object Signing and Encryption IANA registry [IANAJOSE] .

2.2. Requirements Language

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] .

2.3. Defined terms

The terms "Authentication Tag", "Content Encryption Key", "Key Management Mode", "Key Encryption", "Direct Key Agreement", "Key Agreement with Key Wrapping" and "Direct Encryption" are defined in the JWE specification [RFC7516] .

The terms "Authentication", "Ciphertext", "Digital Signature", "Encryption", "Initialization Vector (IV)", "Message Authentication Code (MAC)", "Plaintext" and "Salt" are defined by the Internet Security Glossary, Version 2 [RFC4949] .

Annotated Message A DARE Message that contains an Annotations field with at least one entry.

Authentication Data A Message Authentication Code or authentication tag.

Complete Message A DARE message that contains the key exchange information necessary for the intended recipient(s) to decrypt it.

Detached Message A DARE message that does not contain the key exchange information necessary for the intended recipient(s) to decrypt it.

Encryption Context The master key, encryption algorithms and associated parameters used to generate a set of one or more enhanced data sequences.

Encoded data sequence (EDS) A sequence consisting of a salt, content data and authentication data (if required by the encryption context).

Enhancement Applying a cryptographic operation to a data sequence. This includes encryption, authentication and both at the same time.

Generator The party that generates a DARE message.

Group Encryption Key A key used to encrypt data to be read by a group of users. This is typically achieved by means of some form of proxy re-encryption or distributed key generation.

Group Encryption Key Identifier A key identifier for a group encryption key.

Master Key (MK) The master secret from which keys are derived for authenticating enhanced data sequences.

Recipient Any party that receives and processes at least some part of a DARE message.

Related Message A set of DARE messages that share the same key exchange information and hence the same Master Key.

Uniform Data Fingerprint (UDF) The means of presenting the result of a cryptographic digest function over a data sequence and content type identifier specified in the Uniform Data Fingerprint specification [draft-hallambaker-udf]

3. Architecture

A DARE message is a sequence of three parts as follows.

Header A JSON object containing information a reader requires to begin processing the message.

Payload An array of octets.

Trailer A JSON object containing information calculated from the message payload.

For example, the following sequence is a JSON encoded DARE Message with an empty header, a payload of zero length and an empty trailer:

```
[ {}, "", {} ]
```

Figure 1

DARE Messages MAY be encoded using JSON serialization or a binary serialization for greater efficiency.

JSON Offers compatibility with applications and libraries that support JSON. Payload data is encoded using Base64 incurring a 33% overhead.

JSON-B A superset of JSON encoding that permits binary data to be encoded as a sequence of length-data segments. This avoids the Base64 overhead incurred by JSON encoding.

JSON-C A superset of JSON-C which provides additional efficiency by allowing field tags and other repeated string data to be encoded by reference to a dictionary.

DARE Message processors MUST support JSON serialization and SHOULD support JSON-B serialization.

3.1. Processing Considerations

The DARE Message Syntax supports single pass encoding and decoding without buffering of data. All the information required to begin processing a DARE message (key agreement information, digest algorithms), is provided in the message header. All the information that is derived from message processing (authentication codes, digest values, signatures) is presented in the message trailer.

The choice of message encoding does not affect the semantics of message processing. A DARE Message MAY be reserialized under the

same serialization or converted from any of the specified serialization to any other serialization without changing the semantics or integrity properties of the message.

3.2. Content Metadata and Annotations

A header MAY contain header fields describing the payload content. These include:

ContentType Specifies the IANA Content Type.

Annotations A list of Encoded Data Sequences that provide application specific annotations to the message.

The format of the Encoded Data Sequences is described in the following section.

Consider the following mail message:

```
From: Alice@example.com
To: bob@example.com
Subject: TOP-SECRET Product Launch Today!

The CEO told me the product launch is today. Tell no-one!
```

Figure 2

Existing encryption approaches require that header fields such as the subject line be encrypted with the body of the message or not encrypted at all. Neither approach is satisfactory. In this example, the subject line gives away important information that the sender probably assumed would be encrypted. But if the subject line is encrypted together with the message body, a mail client must retrieve at least part of the message body to provide a 'folder' view.

The following is a plaintext DARE Message in which the header fields of the mail message are presented as annotations:

```
[{
    "cty":"application/example-mail",
    "Annotations":["iAEBiBdGcm9tOiBBbGljZUBleGFtcGxlLmNvbYgA",
      "iAECiBNUbzogYm9iQGV4YW1wbGUuY29tiAA",
      "iAEDiClTdWJqZWN0OiBUT1AtU0VDUkVUIFByb2R1Y3QgTGF1bmNoIFRvZGF5
  IYgA"
    ]},
  "VGhlIENFTyB0b2xkIG1lIHRoZSBwcm9kdWN0IGxhdW5jaCBpcyB0b2RheS4gVGVs
  bCBuby1vbmUh"
  ]
```

Figure 3

3.3. Encoded Data Sequence

An encoded data sequence (EDS) is a sequence of octets that encodes a data sequence according to cryptographic enhancements specified in the context in which it is presented. An EDS MAY be encrypted and MAY be authenticated by means of a MAC. The keys and other cryptographic parameters used to apply these enhancements are derived from the cryptographic context and a Salt prefix specified in the EDS itself.

An EDS sequence contains exactly three binary fields encoded in JSON-B serialization as follows:

Salt Prefix A sequence of octets used to derive the encryption key, Initialization Vector and MAC key as required.

Body The plaintext or encrypted content.

Authentication Tag The authentication code value in the case that the cryptographic context specifies use of authenticated encryption or a MAC, otherwise is a zero-length field.

Requiring all three fields to be present, even in cases where they are unnecessary simplifies processing at the cost of up to six additional data bytes.

The encoding of the 'From' header of the previous example as a plaintext EDS is as follows:

```
88 01
  01
88 17
  46 72 6f 6d 3a 20 41 6c   69 63 65 40 65 78 61 6d
  70 6c 65 2e 63 6f 6d
88 00

~~~~
```

Figure 4

3.4. Encryption and Integrity

Encryption and integrity protections MAY be applied to any DARE Message Payload and Annotations.

The following is an encrypted version of the message shown earlier. The payload and annotations have both increased in size as a result of the block cipher padding. The header now includes Recipients and Salt fields to enable the content to be decoded.

```
[{
    "enc":"A256CBC",
    "Salt":"cvbklMnkhoWE1-3_t3Z-UQ",
    "cty":"application/example-mail",
    "Annotations":["iAEBiCDUKHYfjqs5SHQAzwezya-guHTj0SkhmeHtjbHSWjLa
  Fg",
      "iAECiCCI5gIDQIVQLnQmnc3ZUHb4rtnq3U-ctF52eEXRdNapxw",
      "iAEDiDAsnYWB_5SpxqyX_GLKSW4ztdPbEfsxMAUQnRzr7moJoORh1SxzLLYt
  NPovkFVqXBg"
      ],
    "recipients":[{
        "kid":"MDAD3-E4BYE-MK6CH-QA2HD-TKRS2-KIX5Y-A",
        "epk":{
          "PublicKeyDH":{
            "kid":"MDIGT-2AA2Q-HFVRN-WYCNY-F32NC-FB4NP-A",
            "Domain":"YE6bnq1MlX5ojaJto6PLP_PEwA",
            "Public":"a3VlD_zF5tPfV3GxjfsdkdjDjqT0isXW9iwbZeXWGiCBV
  dwTIgXAzDF0SqBMLbqe31p1ZEMFh0mas6evqNUCM-yb9HuVtjzaaDcn3_jH2kkgQw
  S9_4cD8_qFCazvxwAgKCnLn_VzKOBduT0uvQz1FEuCSv4t0kLlbCwDIe_7TMaoQPN
  RcsVvMPPW6lfWkC5CiHaGN78bNCzkmplAbHV2g10oC4_NdRZ6m8kBkXWz3bnnLFGI
  KWzXKTgfQXrksSktKa9pkwA_6KWEMiLPPHQhtj7wud09o7TzWfdcz3lFaGeMmTBSH
  dFyRl0LSbia9qsrgN6tLUCwOtbuKV3XoEYlvwA"}},
        "wmk":"mzXRWy5OjlsZFHT_teqx3E6wtL6eLVFekiUXO_txMUSt3ILj9xgJ
  Lw"}
    ]},
  "vTg5MAhAmIKF7LCFeE8xFhqOFQ5znVtkIctqyyctDkTNL69Tf0KhsC5VoVty3JCq
  RNth3hIfdHJZlU_BnMRzyA"
  ]
```


Figure 5

3.4.1. Key Exchange

The DARE key exchange is based on the JWE key exchange except that encryption modes are intentionally limited and the output of the key exchange is the DARE Master Key rather than the Content Encryption Key.

A DARE Key Exchange MAY contain any number of Recipient entries, each providing a means of decrypting the Master Key using a different private key.

If the Key Exchange mechanism supports message recovery, Direct Key Agreement is used, in all other cases, Key Wrapping is used.

This approach allows messages with one intended recipient to be handled in the exact same fashion as messages with multiple recipients. While this does require an additional key wrapping

operation, that could be avoided if a message has exactly one intended recipient, this is offset by the reduction in code complexity.

If the key exchange algorithm does not support message recovery (e.g. Diffie Hellman and Elliptic Curve Diffie-Hellman), the HKDF Extract-and-Expand Key Derivation Function is used to derive a master key using the following info tag:

"dare-master" [64 61 72 65 2d 6d 61 73 74 65 72] Key derivation info field used when deriving a master key from the output of a key exchange.

The master key length is the maximum of the key size of the encryption algorithm specified by the key exchange header, the key size of the MAC algorithm specified by the key exchange header (if used) and 256.

3.4.2. Key Identifiers

The JWE/JWS specifications define a kid field for use as a key identifier but not how the identifier itself is constructed. All DARE key identifiers are either UDF key fingerprints [draft-hallambaker-udf] or Mesh/Recrypt Group Key Identifiers.

A UDF fingerprint is formed as the digest of an IANA content type and the digested data. A UDF key fingerprint is formed with the content type application/pkix-keyinfo and the digested data is the ASN.1 DER encoded PKIX certificate keyInfo sequence for the corresponding public key.

A Group Key Identifier has the form <fingerprint>@<domain>. Where <fingerprint> is a UDF key fingerprint and <domain> is the DNS address of a service that provides the encryption service to support decryption by group members.

3.4.3. Salt Derivation

A Master Salt is a sequence of 16 or more octets that is specified in the Salt field of the header.

The Master Salt is used to derive salt values for the message payload and associated encoded data sequences as follows.

Payload

EDS

Encoders SHOULD NOT generate salt values that exceed 1024 octets.

The salt value is opaque to the DARE encoding but MAY be used to encode application specific semantics including:

- Frame number to allow reassembly of a data sequence split over a sequence of messages which may be delivered out of order.
- Transmit the Master Key in the manner of a Kerberos ticket to allow some (but not necessarily all) to avoid the need to perform a key exchange.
- Identify the Master Key under which the Enhanced Data Sequence was generated.
- Enable erasure of the encrypted data plaintext by erasure of the encryption key.

For data erasure to be effective, the salt MUST be constructed so that the difficulty of recovering the key is sufficiently high that it is infeasible. For most purposes, a salt with 128 bits of appropriately random data is sufficient.

3.4.4. Key Derivation

Encryption and/or authentication keys are derived from the Master Key using a Extract-and-Expand Key Derivation Function as follows:

1. The Master Key and salt value are used to extract the PRK (pseudorandom key)
2. The PRK is used to derive the algorithm keys using the application specific information input for that key type.

The application specific information inputs are:

"dare-encrypt" [64 61 72 65 2d 65 6e 63 72 79 70 74] To generate an encryption or encryption with authentication key.

"dare-iv" [64 61 72 65 2d 65 6e 63 72 79 70 74] To generate an initialization vector.

"dare-mac" [dare-mac] To generate a Message Authentication Code key.

3.5. Signature

While encryption and integrity enhancements can be applied to any part of a DARE message, signatures are only applied to payload digest values calculated over one or more message payloads.

The payload digest value for a message is calculated over the binary payload data. That is, after any encryption enhancement has been applied but before the message encoding is applied. This allows messages to be converted from one encoding to another without affecting signature verification.

Single Payload The signed value is the payload digest of the message payload.

Multiple Payload. The signed value is the root of a Merkle Tree in which the payload digest of the message is one of the leaves.

Verification of a multiple payload signature naturally requires the additional digest values required to construct the Merkle Tree. These are provided in the Trailer in a format that permits multiple signers to reference the same tree data.

4. Algorithms

4.1. Field: kwd

The key wrapping and derivation algorithms.

Since the means of public key exchange is determined by the key identifier of the recipient key, it is only necessary to specify the algorithms used for key wrapping and derivation.

The default (and so far only) algorithm is kwd-aes-sha2-256-256.

Advanced Encryption Standard (AES) Key Wrap with Padding Algorithm [RFC3394] is used to wrap the Master Exchange Key. AES 256 is used.

HMAC-based Extract-and-Expand Key Derivation Function [RFC5869] is used for key derivation. SHA-2-256 is used for the hash function.

5. Reference

A DARE Message consists of a Header, an Enhanced Data Sequence (EDS) and an optional trailer. This section describes the JSON data fields used to construct headers, trailers and complete messages.

Wherever possible, fields from JWE, JWS and JWK have been used. In these cases, the fields have the exact same semantics. Note however that the classes in which these fields are presented have different structure and nesting.

5.1. Message Classes

A DARE Message contains a single DAREMessageSequence in either the JSON or Compact serialization as directed by the protocol in which it is applied.

5.1.1. Structure: DAREMessageSequence

A DARE Message containing Header, EDS and Trailer in JSON object encoding. Since a DAREMessage is almost invariably presented in JSON sequence or compact encoding, use of the DAREMessage subclass is preferred.

Although a DARE Message is functionally an object, it is serialized as an ordered sequence. This ensures that the message header field will always precede the body in a serialization, this allowing processing of the header information to be performed before the entire body has been received.

Header: DAREHeader (Optional) The message header. May specify the key exchange data, pre-signature or signature data, cloaked headers and/or encrypted data sequences.

Body: Binary (Optional) The message body

Trailer: DARETrailer (Optional) The message trailer. If present, this contains the signature.

5.2. Header and Trailer Classes

A DARE Message sequence MUST contain a (possibly empty) DAREHeader and MAY contain a DARETrailer.

5.2.1. Structure: DARETrailer

A DARE Message Trailer

Signatures: DARESignature [0..Many] A list of signatures. A message trailer MUST NOT contain a signatures field if the header contains a signatures field.

5.2.2. Structure: DAREHeader

Inherits: DARETrailer

A DARE Message Header. Since any field that is present in a trailer MAY be placed in a header instead, the message header inherits from the trailer.

EncryptionAlgorithm: String (Optional) The encryption algorithm as specified in JWE

AuthenticationAlgorithm: String (Optional) Message Authentication Code algorithm

Cloaked: Binary (Optional) If present in a header or trailer, specifies an encrypted data block containing additional header fields whose values override those specified in the message and context headers.

When specified in a header, a cloaked field MAY be used to conceal metadata (content type, compression) and/or to specify an additional layer of key exchange. That applies to both the Message body and to headers specified within the cloaked header.

Processing of cloaked data is described in?

ContentType: String (Optional) The content type field as specified in JWE

EDSS: Binary [0..Many] If present, the Encrypted Data Segments field contains a sequence of Encrypted Data Segments encrypted under the message Master Key. The interpretation of these fields is application specific.

Signers: DARESigner [0..Many] A list of 'presignature'

Recipients: DARERecipient [0..Many] A list of recipient key exchange information blocks.

5.3. Cryptographic Data

DARE Message uses the same fields as JWE and JWS but with different structure. In particular, DARE messages MAY have multiple recipients and multiple signers.

5.3.1. Structure: DARESigner

The signature value

Dig: String (Optional) Digest algorithm hint. Specifying the digest algorithm to be applied to the message body allows the body to be processed in streaming mode.

Alg: String (Optional) Key exchange algorithm

KeyIdentifier: String (Optional) Key identifier of the signature key.

Certificate: X509Certificate (Optional) PKIX certificate of signer.

Path: X509Certificate (Optional) PKIX certificates that establish a trust path for the signer.

5.3.2. Structure: X509Certificate

X5u: String (Optional) URL identifying an X.509 public key certificate

X5: Binary (Optional) An X.509 public key certificate

5.3.3. Structure: DARESignature

Inherits: DARESigner

The signature value

SignatureValue: Binary (Optional) The signature value as an Enhanced Data Sequence under the message Master Key.

5.3.4. Structure: DARERecipient

Recipient information

KeyIdentifier: String (Optional) Key identifier for the encryption key.

The Key identifier MUST be either a UDF fingerprint of a key or a Group Key Identifier

KeyWrapDerivation: String (Optional) The key wrapping and derivation algorithms.

WrappedMasterKey: Binary (Optional) The wrapped master key. The master key is encrypted under the result of the key exchange.

RecipientKeyData: String (Optional) The per-recipient key exchange data.

6. Security Considerations

6.1. Encryption/Signature nesting

6.2. Side channel

6.3. Salt reuse

7. IANA Considerations

8. Acknowledgements

9. Test Examples

In the following examples, Alice's encryption private key parameters are:


```
{
  "PrivateKeyDH":{
    "kid":"MDAD3-E4BYE-MK6CH-QA2HD-TKRS2-KIX5Y-A",
    "Domain":"YE6bnq1MlX5ojaJto6PLP_PEwA",
    "Public":"GaMqeF8I2s00AUMm1D1dcXflA61rrbUflBbpYHLtz3GrwkVR_JsGd
  qlCVWQEX9McxsYTJUoFJzeHivLsyoUlAY9H3Qtm_aClOeARO8yXboUTjKtwgH1qu0
  s6kKl-p-tcaqu2PXReTVfOG9HtSR60iSuRZ9G9JHpKQNZCTdpoq4B2oBw6LvS1y-p
  18bDCB_JeOOcqT8Aba08TjkhIG2OHaZDEgdDOU1nIAT1hIxh48sLuPSdou-F76l47
  8Jte_oJKTqGCUUeOm_397_d4sbaCiPO0RMllIFC7VEhi2TIDL6bRF7ujmCdxreYn9
  DlDjr1nF7hkcXEALL_5NyLVxwKdBw",
    "Private":"cokgZSxP_IdDAswgUZWDFu6qlKrEVX83uwDU2jI5LWiIoknyE6dY
  wW2_mADDQ_2DRxm73J3fTH6C73KAOx-cJnJYDqr8kU3FhBEXBn8wxFL6M-SgVxyKa
  jZ5MRL0-3EAyNCAE_HLNoeUqAqGAchw-lnDgQO7e4F0hkEwa29JdwrBKTJxY93WAB
  Xd3xZahbNWs3sh2MU_FGU6AzfnTWPdvWTTxImySVF47pex-gRbYFVV-Cm1ZUuof-5
  flM-RwUlnf28qJ8SJ_-zKmjLc6TSrMky8ox1u6v7WHoFRiweMCyBQ3x2Zc2ypXPfF
  DvvAu5RYiVoUbrlES1UswPObiE2lOg"}}
```


Figure 6

Alice's signature private key parameters are:

```
{
  "PrivateKeyRSA":{
    "kid":"MBWNO-2J43U-ESWKW-XQWL6-6YGEW-UOPWU-A",
    "n":"1NzbmakMalVH1mRv7TEDEhwXDNojn5wZbq1tv1gp5PgZwzX-klYXuFhj0-
  MpO0zcwptsUaYJhwdvvgW_1udUpISQYluXOB3UMj2e_0yl64MvnqTL47SZQuAN3QQ
  9cuCw_-_Eyj_jerspauqa6RpNzGcabZrtRl7J7DPVZ3SNlw-H_Wxd4HkrFVW_Yqup
  htNL1JciQJYm2DSu9dbetqPZ80x6IBargY850mBYOzvNNE5S-dRJQHoJY4SG-ESYF
  BuAHtBlOMgbI0XNiq96EegA-vPW9XRF-SHdX5mkafefDGK4rT_RoE4gRwhDM3jbZ8
  1-F2ZA_GpNVEvB-25_vF96lQ",
    "e":"AQAB",
    "d":"k_v_h7Jo-TvUt44X6jSax-pTdBHrljk1zSYxGEe4yIBbmMVe-Gl2ECkTLe
  nNbnafO4RGJ_Vgxkk7PEZO-p7Uz5OBtX-rf83tCgihEyg8aaFIZ-h1_xY9Pqr5uGA
  MQGNJaoVMsLb99QNNZhE4JTquP56mVvDQaI3Zn6bhhA0ZqpxS_x6iRUV5KnHCRd47
  DKGHcAsQR_caxGec7M_XNPqpl0tcoGQz46-I0SVVcqtjb_YysEVez4eJhb_ZTU4C7
  pz4wXDj6B0ppFJVZEMaIhKo8FCnoQdXEJl4vSiBzUFrSlPc6gjQk2CBhxc_kb782w
  VwW5wtgOVxhV1k2piQ2NrlOQ",
    "p":"6osXfrJLiPfK0ky17vqzRs-M5mOxZU2LEFGyHTXxx6EYTWixEsx2Sdf6kx
  UD5K_QdYnqqd3yobbGdDpMwwEuwgogWVCz90nc9QdCUy4MCpz8lpOQd1i_tXMmtOg
  GBu9mapMTEOwc7HlLlSDRezv_TzTAH4izv-CUEZ_M5EcwyFM",
    "q":"6FYD6NV4rKU1ACtGQyIvwGrkrS_F6phB-whx0nSVFkbkwppJiPnC6XqjZ3
  OYPCZylxaTHFnxCs0nntrraeNEcWfNPrpTN5XIZjbOiIaKA2iCQkWlDoi8oduEtTK
  oIcuy32oBz6MpUeCWzl0ZQ4EeTF3RyCc_jpf8oRvZ0e3ItHc",
    "dp":"hfjbe9BmWx-HqCaPSanEW-9UQYmym_X2OGUiA5N7vxci5ZymgOFvs_B9v
  iQj7C4NOgaEl3EjFgJsS5m9nSoAxm-4WKxDkD6NyxzRYugLkshnc69otvNn1kKnWn
  CqeK2o57mJC4KDZwRGCzIK1oTH6jtsfta8Lh8fFQ4doEuV7uc",
    "dq":"r6R_ViE0Foja1aLhflU09mmZMViBbkXm86nBqtHZ97pmrLvJRdVTxgCh0
  c6w0yBZ1uEJHBDeykSoZE6qVCWtE3Le1kI0MTx6ANQENXBInCUA_Kr8Ck3TFSYIYJ
  fIRaxiMMZKUjfOQAji2WXGeKL_TcpLkt4hDWLXaNDOTgdOiSc",
    "qi":"DfHtLB1Ox1Kgp3E4jqy5Qxeb7-v7_uv8n_5-E1OQ3NLSRV2m_auojkR19
  nY3gokHKNSXM41qKlJLU00lROjOO2KUq57s8GZkheVfbJLNCJ6KAw_aRT2IgyJm2b
  e2v5OCHSkm88tgJWbtKj-OPKTFV5gOMVdeCzGX286ErjDHGCM"}}
```


Figure 7

The body of the test message is the UTF8 representation of the following string:


```
"This is a test long enough to require multiple blocks"
```


Figure 8

The EDS sequences, are the UTF8 representation of the following strings:


```
"Subject: Message metadata should be encrypted"
"2018-02-01"
```


Figure 9

9.1. Plaintext Message

A plaintext message without associated EDS sequences is an empty header followed by the message body:

```
{
  "DAREMessage":[{},
    "VGhpcyBpcyBhIHRlc3QgbG9uZyBlbm91Z2ggdG8gcmVxdWlyZSBtdWx0aXBsZS
  BibG9ja3M"
    ]}
```

Figure 10

9.2. Plaintext Message with EDS

If a plaintext message contains EDS sequences, these are also in plaintext:

```
{
  "DAREMessage":[{
      "Annotations":["iAEBiC1TdWJqZWN0OiBNZXNzYWdlIG1ldGFkYXRhIHNob3
  VsZCBiZSBlbmNyeXB0ZWSIAA",
       "iAECiAoyMDE4LTAyLTAxiAA"
       ]},
    "VGhpcyBpcyBhIHRlc3QgbG9uZyBlbm91Z2ggdG8gcmVxdWlyZSBtdWx0aXBsZS
  BibG9ja3M"
    ]}
```

Figure 11

9.3. Encrypted Message

The creator generates a master session key:

```
  94 D7 0B C4  8C 43 B2 0E  7C 2A F8 C2  37 9C 8F E5
  CA A7 8F BC  4C B3 9F CB  14 AA 5F 4C  41 AF 52 4B
```

Figure 12

The creator generates an ephemeral key:

```
{
  "PrivateKeyDH":{
    "kid":"MAK3V-IOKWY-NFGD7-YBPY3-OWINU-3WRPC-A",
    "Domain":"YE6bnq1MlX5ojaJto6PLP_PEwA",
    "Public":"aeZpao1EpPIXrAFheMEHjjxJMucxpFJ5LvDXJunryv0xfpDseipYF
  8jng0pBAE-P7CDPdw_WQgYDdW4NgF7BUbYq1EaXbcV1uLYBB2Yw2MAj-Up-7pO1Fc
  5kjiDnRz0Sd0lJARCtkR5A2cuyMvuEg8cntCQWcoyzh4ep1PktlWDKoSKltovO0cM
  -9hj-uGFiBV6-cb15fkQ7pyKp-XiH_2YkMCiUYhkPx9ZObr1WNMVHn9TyLgufPnLH
  skaE2JDFckEBMljwXdI9z_DeUx4FNESRQQ68UGfDhfepPwR3_xbL9OFbFiJSjdd51
  pxfzvi2SbJm8uKY5K3omsUMszyqjQA",
    "Private":"iXfaIGr7vRVqJyTZEj_5F-3n7DL5KZzQ0sL4wydy1-F27A6kHjBW
  yOrV0qr9BLuLRGH4vvfYSael_ILWzgKfC22KqN1CjJdzChwZvFyQCCXynrkH06KAk
  uDWpczlC0n8T8WvTboalzVNvOfVM__QcywomDoY6tUCbIn6fQ5xJyWaR8EGQHPFdF
  7W-O0ezKIieSKC25fIkIOt3c2-dqQdzUl2Vlk6R_6wkFxhX2NNqpV7VktrglQ6AUJ
  b8gFt6H7gmsSvdks8lXv-1VHwwH_8HBRAOcqsRu9THuz8T4H30d30FZ-NXtCpUEwi
  citGRXZ1UzmChc_IuIkfdQfuAGvttAA"}}
```


Figure 13

The key agreement value is calculated:


```
  A2 CD 5A 08  24 70 27 5F  6F 28 A5 6D  B0 AA 47 31
  50 D0 F3 DA  07 13 4A 72  F7 BB 19 8E  72 60 82 51
  32 0B CF 7B  A1 8A FD 40  7E D5 E1 79  87 20 8B 2A
  73 F5 20 2D  1C 94 FB 2D  8D 4F C0 DA  6D D7 C0 7A
  D1 C6 35 A8  AD D2 DD BF  F9 19 C3 BB  60 20 04 F7
  D7 F0 81 F6  F3 94 68 DE  6F B6 B4 67  CD 9B F3 E4
  A0 28 6E 59  5A A2 FE 00  10 17 B3 34  2C 07 20 06
  2C B9 34 F0  4C C8 E9 C1  CA 80 1D 02  15 B6 CE D4
  EC BB 91 2B  FA 7D 9B 14  24 13 16 46  1C D2 5B 12
  4A 0E 60 28  D6 38 E1 35  79 D6 DF 66  C0 C6 7F A7
  E3 C1 9F 25  CD 01 A5 52  7A C1 B5 ED  3C 24 0F 8F
  DD E6 27 60  F7 2D CF D5  7E 13 F8 29  53 7B 43 FC
  13 F6 7B 55  8A 44 AD 16  A9 27 75 E0  9A DF 95 F6
  F6 2C D5 26  88 1B EB 20  EE 26 C6 4E  17 20 54 BA
  DB A5 37 A4  99 A2 FC 49  93 DE F5 8F  F5 3B D1 F3
  FF 9D 71 38  B1 AE 94 E0  10 61 16 CA  8F B9 16 1C
```


Figure 14

The key agreement value is used as the input to a HKDF key derivation function with the info parameter System.Byte[] to create the key used to wrap the master key:

```
40 39 4F 58  34 F4 0F B3  AA 0F 88 7D  EE AC 98 1A
91 84 25 3C  74 F6 E4 25  35 DB D6 78  6A ED 6B F5
```

Figure 15

The wrapped master key is:

```
14 98 FB 8C  0A D0 C8 C6  65 D3 9F F5  6B 80 42 96
0A 71 E9 93  F7 14 D2 29  C9 DB 96 FE  FA 9A DB 74
29 F4 35 36  BC CF BF 41
```

Figure 16

This information is used to calculate the Recipient information shown in the example below.

To encrypt a message, we first generate a unique salt value:

```
82 D1 8C BA  A0 F0 26 5C  7A 35 5F 82  1F 88 35 CD
```

Figure 17

The salt value and master key are used to generate the payload encryption key:

```
D3 74 4A 8E  69 65 A4 71  8B 14 44 AA  CC E6 7F 66
07 87 91 3E  F3 41 DE 2D  DE 5F 4A 7C  19 D5 75 79
```

Figure 18

Since AES is a block cipher, we also require an initializarion vector:

```
0A 5A 94 71  54 7B C2 16  E8 BF D2 66  5D 8D E5 BF
```

Figure 19

The output sequence is the encrypted bytes:

```
  30 33 D9 A1  12 19 EF 96  1C 43 45 98  50 7B D2 B1
  A7 94 C2 C8  1B 52 00 3E  DD A2 B4 9F  51 A5 BD 8C
  F4 3D 81 15  95 D0 6D D7  19 5F 28 E3  A0 FF EA 26
  29 27 78 B7  49 E7 B2 E3  AD A7 8C D1  C0 61 D5 68
```

Figure 20

Since the message is not signed, there is no need for a trailer. The completed message is:

```
{
  "DAREMessage":[{
      "enc":"A256CBC",
      "Salt":"gtGMuqDwJlx6NV-CH4g1zQ",
      "recipients":[{
          "kid":"MDAD3-E4BYE-MK6CH-QA2HD-TKRS2-KIX5Y-A",
          "epk":{
            "PublicKeyDH":{
              "kid":"MAK3V-IOKWY-NFGD7-YBPY3-OWINU-3WRPC-A",
              "Domain":"YE6bnq1MlX5ojaJto6PLP_PEwA",
              "Public":"aeZpao1EpPIXrAFheMEHjjxJMucxpFJ5LvDXJunryv0
  xfpDseipYF8jng0pBAE-P7CDPdw_WQgYDdW4NgF7BUbYq1EaXbcV1uLYBB2Yw2MAj
  -Up-7pO1Fc5kjiDnRz0Sd0lJARCtkR5A2cuyMvuEg8cntCQWcoyzh4ep1PktlWDKo
  SKltovO0cM-9hj-uGFiBV6-cb15fkQ7pyKp-XiH_2YkMCiUYhkPx9ZObr1WNMVHn9
  TyLgufPnLHskaE2JDFckEBMljwXdI9z_DeUx4FNESRQQ68UGfDhfepPwR3_xbL9OF
  bFiJSjdd51pxfzvi2SbJm8uKY5K3omsUMszyqjQA"}},
          "wmk":"FJj7jArQyMZl05_1a4BClgpx6ZP3FNIpyduW_vqa23Qp9DU2vM
  -_QQ"}
        ]},
    "MDPZoRIZ75YcQ0WYUHvSsaeUwsgbUgA-3aK0n1GlvYz0PYEVldBt1xlfKOOg_-
  omKSd4t0nnsuOtp4zRwGHVaA"
    ]}
```

Figure 21

9.4. Signed Message

Signed messages specify the digest algorithm to be used in the header and the signature value in the trailer. Note that the digest algorithm is not optional since it serves as notice that a decoder should digest the payload value to enable signature verification.

```
{
  "DAREMessage":[{
      "dig":"S512"},
    "VGhpcyBpcyBhIHRlc3QgbG9uZyBlbm91Z2ggdG8gcmVxdWlyZSBtdWx0aXBsZS
  BibG9ja3M",
    {
      "signatures":[{
          "signature":"s4LxUqNOV1-0uryJCiYFqnKak3EHZuLBaaUrehaRCaZH
  2KqASBftk87fTl73XUeV-0TfvlfV8STP7l0brcWnPTKQgXSMXvuoxqF0n9qcpXubB
  xzdGHdFX-5GeQMFsr8NutBBng-LSZVe2eCb7n29dpCgZ84v5Y4JGzvKHOy8vaU"}
        ],
      "PayloadDigest":"raim8SV5adPbWWn8FMM4mrRAQCO9A2jZ0NZAnFXWlG0x
  F6sWGJbnKSdtIJMmMU_hjarlIPEoY3vy9UdVlH5KAg"}
    ]}
```

Figure 22

9.5. Signed and Encrypted Message

A signed and encrypted message is encrypted and then signed. The signer proves knowledge of the payload plaintext by providing the plaintext witness value.

```
{
  "DAREMessage":[{
      "enc":"A256CBC",
      "dig":"S512",
      "Salt":"AB4x8M6bLZjdSr6W9ntB2A",
      "recipients":[{
          "kid":"MDAD3-E4BYE-MK6CH-QA2HD-TKRS2-KIX5Y-A",
          "epk":{
            "PublicKeyDH":{
              "kid":"MDJLN-DFSIO-ZOL6P-2NOZD-YHBQH-3JQSK-A",
              "Domain":"YE6bnq1MlX5ojaJto6PLP_PEwA",
              "Public":"XBYmS2ui9AgETUvwK9d3eLWQiQ8240yddATCETYAAUH
  H0m29aDiBpIT9TcDVs6dtnWH-mnZMTfwT_RQJvyMcfM1AYFvmMfu3GoFS6Z4nngor
  hZGFNGgWEG7yX9eOvaiBWpTs2gw7AM4PvUh9r5G0IolfIvQcTRZHpKuU3GqV0E0xp
  PhrEajKzLSrDNu6tZ2yDe7BQBSI7YNgZGeXqOTr3KmfdfYU7Hk_ak3M_LhtCCe2-m
  bwPuDUD0PUWScZLrKbMWzLIzO0OrDYMwjSrJmirH2RxQKM2a_kJAQoBFtmXjlaFLC
  yvOKK1ibPnWn7_JZtUy_k4IvDex6b7VKVap6ySQ"}},
          "wmk":"qS1rvoKhPjlVy3B6uYkgRYYTJtOzDEeiqqezpTAoGh8Op2VV6x
  cbUw"}
        ]},
    "MudZFUAZRBINTZO9szFJCr9NUrE1s1meuZfRftorYK5_gI8-NAjLud8Jx8LM_R
  DVNXfi5GKGNFuvfr0MlqyFRQ",
    {
      "signatures":[{
          "signature":"SAzZxGhqUVUSQCta3vZ1OL9AkdLD40sVSE9k2opol_FW
  4aEyMEm4c5UZTnkf6Ndkz1O47TrjHowzagSwkTdq-dV7r7OxH-oQ9fTeG1GyT8xwo
  Yw4KsSjD71X_lwgi2BI-WMvdVliOFk78MC52eF7SsA0mkbWnSSB0WHBjoJvEaI",
          "witness":"zvxEBQmdHJCin9brH4lpD-8CNJXWezsSMhWAT-CSlk0"}
        ],
      "PayloadDigest":"ScoVguz7ufoe547gh9LIC00ptI24ZzpwLplJFzeJQx9d
  G7LfeBTi2oNPr2GBuLp29pRRuDqgdRCftuLeRBl8kg"}
    ]}
```


Figure 23

10. References

10.1. Normative References

[draft-hallambaker-jsonbcd]
Hallam-Baker, P., "Binary Encodings for JavaScript Object Notation: JSON-B, JSON-C, JSON-D", draft-hallambaker-jsonbcd-13 (work in progress), July 2018.

[draft-hallambaker-udf]
Hallam-Baker, P., "Uniform Data Fingerprint (UDF)", draft-hallambaker-udf-10 (work in progress), April 2018.

[IANAJOSE]
"[Reference Not Found!]".

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997.

[RFC2315] Kaliski, B., "PKCS #7: Cryptographic Message Syntax Version 1.5", RFC 2315, DOI 10.17487/RFC2315, March 1998.

[RFC3394] Schaad, J. and R. Housley, "Advanced Encryption Standard (AES) Key Wrap Algorithm", RFC 3394, DOI 10.17487/RFC3394, September 2002.

[RFC4880] Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R. Thayer, "OpenPGP Message Format", RFC 4880, DOI 10.17487/RFC4880, November 2007.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007.

[RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010.

[RFC7159] Bray, T., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015.

[RFC7516] Jones, M. and J. Hildebrand, "JSON Web Encryption (JWE)", RFC 7516, DOI 10.17487/RFC7516, May 2015.

[RFC7517] Jones, M., "JSON Web Key (JWK)", RFC 7517, DOI 10.17487/RFC7517, May 2015.

[RFC7518] Jones, M., "JSON Web Algorithms (JWA)", RFC 7518, DOI 10.17487/RFC7518, May 2015.

10.2. Informative References

[Davis2001]
Davis, D., "Defective Sign & Encrypt in S/MIME, PKCS#7, MOSS, PEM, PGP, and XML", May 2001.

10.3. URIs

[1] http://mathmesh.com/Documents/draft-hallambaker-dare-message.html

[2] http://mathmesh.com/Documents/draft-hallambaker-dare-message.html

[3] http://mathmesh.com/Documents/draft-hallambaker-dare-message.html

[4] http://mathmesh.com/Documents/draft-hallambaker-dare-message.html

Author's Address

Phillip Hallam-Baker
Comodo Group Inc.

Email: philliph@comodo.com