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 January 14, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of
the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction ................................................. 3
   1.1. Existing approaches .................................. 4
   1.2. The DARE Approach .................................. 4
2. Definitions .................................................. 5
   2.1. Related Specifications ............................... 5
   2.2. Requirements Language ............................... 5
   2.3. Defined terms ....................................... 6
3. Applications .................................................. 7
   3.1. Message Types ....................................... 7
      3.1.1. Consolidated Messages ......................... 7
      3.1.2. Related Messages ............................. 7
      3.1.3. Detached Messages ............................ 7
      3.1.4. Annotated Messages ........................... 8
      3.1.5. Proxy Re-encryption ........................... 8
   3.2. Additional Use Cases ................................ 8
      3.2.1. Streaming data ................................ 8
      3.2.2. Information Erasure by Key Deletion .......... 8
      3.2.3. Field level encryption ....................... 9
4. Message Format ............................................... 9
   4.1. Encodings ........................................... 9
      4.1.1. JSON ........................................ 9
      4.1.2. JSON-B ..................................... 10
      4.1.3. Application Directed Encoding .............. 10
5. Processing Model .......................................... 10
   5.1. Consolidated Message Generation .................... 11
      5.1.1. Message Header ................................ 11
      5.1.2. Enhanced Data Sequence ....................... 12
      5.1.3. Trailer ...................................... 15
   5.2. Consolidated Message Recovery ...................... 15
      5.2.1. Verify signers ................................ 15
      5.2.2. Match recipient info .......................... 15
      5.2.3. Recover Master Key ............................ 15
      5.2.4. Process Enhanced Data Sequences .............. 15
      5.2.5. Signature validation ........................... 16
   5.3. Detached Messages .................................. 16
   5.4. Cloaked Messages .................................. 16
6. Algorithms ................................................... 16
   6.1. Field: kwd ....................................... 16
7. Reference ..................................................... 17
   7.1. Message Classes .................................... 17
      7.1.1. Structure: DAREMessageSequence ............. 17
   7.2. Header and Trailer Classes ........................ 18
      7.2.1. Structure: DARETrailer ...................... 18
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 simplified to remove as many redundant features and alternative means of specifying the same content.

An important innovation in the DARE message format 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 messages.

To maintain the security of the cryptographic algorithm, each octet sequence is encrypted under a different encryption key (and IV if required) derived from the Master Key by means of a salt. Depending on application needs, the salt may be explicit or implicit. An explicit salt is an opaque sequence of octets prepended to the data item. An implicit salt is an octet sequence constructed by application specific means such as the sequence number of a message or the byte position of a field in a file.
1.1. Existing approaches

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

To encrypt a message using RSA, the creator 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.

1.2. The DARE Approach

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).
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.

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].


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

Authentication Data A Message Authentication Code or authentication tag.

Buffered Generation Mode A mode of generating a DARE message in which the data input is read completely before beginning output.

Consolidated 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.

Enhanced 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.

Unbuffered Streaming Mode. A mode of generating a DARE message in which data MAY be output before the input is read.

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. Applications

3.1. Message Types

3.1.1. Consolidated Messages

A consolidated DARE message contains the ciphertext and authentication data for a data object together with the key exchange information necessary for the intended recipient(s) to process it.

Consolidated messages provide the same functionality as traditional PKCS#7/CMS and OpenPGP message formats and may be used as a one-for-one replacement.

3.1.2. Related Messages

A set of DARE messages are related if they share the same key exchange information.

Related messages allow a single key exchange to be amortized over a collection of data items. This is particularly useful when a large collection of short data items is required such as in a server log or a chat-room transcript.

3.1.3. Detached Messages

A detached DARE message contains only the ciphertext and authentication data for a data object and does not provide any key exchange information.
The use of detached messages in a protocol allows key exchange information and message data to be passed to a recipient separately, possibly over different channels or even with entirely different partners.

For example, a service returning the last hundred entries from a log file encrypted as a series of DARE related messages need only provide the key exchange data once.

3.1.4. Annotated Messages

An annotated DARE message contains a message header with encrypted metadata in addition to an encrypted body. The use of annotated messages allows a receiver to process message data and metadata separately.

For example, a mail application typically provides users with a display showing a short summary of the messages received (sender, date, subject, etc.). Encrypting the metadata to be shown to the user in the summary display separately from the message body allows this data to be presented to the user without the need to download any part of the message bodies.

3.1.5. Proxy Re-encryption

The ability to re-use the output of a key exchange is of particular importance when using proxy re-encryption or distributed key generation as completing each key agreement incurs an interaction with the key server.

3.2. Additional Use Cases

3.2.1. Streaming data

The DARE message format supports encryption and decryption in ‘streaming mode’ in which blocks of output data are emitted as the input is presented.

This allows synchronous communications (e.g. video, voice) to be supported and permits files of arbitrary size to be encrypted with finite state. It is not necessary to buffer the entire plaintext or ciphertext before generating a message.

3.2.2. Information Erasure by Key Deletion

Overwriting a DARE salt value prevents decryption of the corresponding data unless the salt can be recovered. Use of a suitably large random salt allows erasure of the salt to be
considered equivalent to erasure of the message data. For example, if a salt of 128 random bits and an encryption algorithm of at least 128 bits are used, the work factor for decryption will be $O(2^{128})$ even if the decryption key is compromised.

3.2.3. Field level encryption

The DARE key exchange and data item encoding may be applied to encrypt multiple fields in a single file under a Common Encryption Key. Field level encryption is particularly useful in database and spreadsheet applications.

4. Message Format

A consolidated DARE message is a sequence of four parts as follows.

Header  Information a reader requires to begin processing the message body.

By definition, the header of a consolidated message contains the key agreement information and the header of a detached message does not. The header may also be used to specify content metadata (such as the data type) and encrypted annotations.

Body  The data block

Trailer  Information a reader requires to complete processing the message body that is not provided in the header.

A detached DARE message has the same sequence of four parts but the header part is empty. The header being communicated out-of-band with respect to the message to which it appears.

4.1. Encodings

A DARE message MAY be presented in JSON encoding or a compact encoding based on JSON-B.

4.1.1. JSON

The DARE message is encoded as JSON sequence with up to three entries. The position of the item in the sequence specifies its function. Thus the Header entry MAY be empty but MUST not be absent.

Header  The header is encoded as a JSON object

Body  The body is encoded as a base64url encoded string
Trailer The trailer is encoded as a JSON object

For example, the following sequence is a JSON encoded message with an empty header and trailer and a salt and body of zero length:

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

Figure 1

4.1.2. JSON-B

JSON-B Encoding provides a more compact representation and in particular, allows ciphertext to be presented in binary form as opposed to Base-64 encoding. Note that JSON-B encoding is a superset of JSON, a JSON-B decoder will be able to decode either format without additional tagging to specify which format is being used.

The square braces used to specify a JSON sequence MUST be present when a DARE Message is embedded in a JSON-B encoded object but MAY be omitted in situations where no ambiguity arises from doing so. For example, when presenting a DARE Message as a standalone file or in a DARE Container.

4.1.3. Application Directed Encoding

Applications MAY define their own encoding mechanisms to suit their needs.

5. Processing Model

The DARE processing model is based on the model in JWE [RFC7516] with the following extensions:

- Support for multiple recipients
- Support for multiple message bodies encrypted under keys derived from a single key exchange.
- Signature and encryption are supported in a single format rather than separately.
- Authentication tags are appended to the end of the message body.
- Message Authentication Codes are supported as a means of authentication.
5.1. Consolidated Message Generation

Two generation modes are supported:

Buffered  The data body is buffered in memory while the data header is completed.

Streaming  The DARE message is generated in a single pass.

Use of buffered mode avoids the need for data chunking and allows messages to provide all the information required for processing in the message header.

Use of streamed mode avoids the need to buffer the data body while assembly of the header is completed. This allows messages of arbitrary size to be processed with fixed resources.

5.1.1. Message Header

The Message header contains the key exchange information which MAY be shared by multiple related messages.

5.1.1.1. Signatures and Signers

If the message is to be authenticated by means of a digital signature, the Header MUST contain either a Signatures field or a Signers field but not both.

The Signatures field contains the actual signature value which cannot usually be calculated until processing of the message body is complete. The Signers field contains all the information from a signature except for the signature itself. This allows a verifier to perform decryption and signer verification processing in parallel and to reject a message that is not signed by an accepted signer before completing processing of the message body.

5.1.1.2. 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.

5.1.1.3. Key Identifier

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 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.

5.1.2. Enhanced Data Sequence

A DARE Enhanced Data Segment is an atomic unit that contains a salt and the result(s) of applying the salt and Master Key to a plaintext under the enhancement mode specified in the key exchange. The following enhancement modes are supported:
Plaintext  The EDS consists of a plaintext.

Authenticated  The EDS consists of a plaintext with an authentication tag appended to the end.

Encrypted  The EDS consists of a ciphertext only

Encrypted and Authenticated  The EDS consists of a ciphertext with an authentication tag appended to the end.

In each case the encryption and/or authentication algorithms and all associated parameters (key size, output length) are specified in the associated key exchange header.

A DARE message MAY contain multiple Enhanced Data Sequences. The message body, cloaked headers, annotations and signature values are all presented as Enhanced Data Sequences.

The message body is distinct from all other Enhanced Data Sequences in that each message MUST have exactly one message body and only the message body can be signed. Additionally, when the compact encoding is used, it is the only Enhanced Data Sequence that can be of variable length.

5.1.2.1. Salt

A salt is a sequence of zero or more octets that is unique within the scope of a Master Key.

Generators 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 will be sufficient.

5.1.2.2. 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.

5.1.2.3. Content

The content of an Enhanced Data Sequence is the plaintext or ciphertext with the appended authentication tag as directed by the enhancement mode.

Support for enhancement of unbuffered streaming data presents implementations with two cases of interest:

5.1.2.3.1. Known Length

If the length of the body is known in advance of enhancement the generator can calculate the final length of the Enhanced Data Segment before encryption begins. This allows encryption of (e.g.) data files as a single data-last production when the compact encoding is being used.

5.1.2.3.2. Unknown Length

If the final length of the body is not known in advance of enhancement the generator must either buffer the output data in memory before generation or use an encoding that permits indefinite
length octet sequences to be represented. In the compact encoding, this means a (possibly zero length) sequence of data-chunk productions terminated by a single data-last production.

5.1.3. Trailer

The trailer only contains information when the message is authenticated by means of a signature.

5.1.3.1. Signatures

A list of message signature values.

If the message header contains a Signer field, the trailer MUST contain a Signatures field giving the corresponding signature values.

5.2. Consolidated Message Recovery

Decryption and verification is the opposite of the generation process.

5.2.1. Verify signers

Recipients MAY verify that the message signatures are adequate to meet the requirements of the security policy at any point in message recovery.

The recipient determines which signer or signature entries are appropriate to their needs and if so, which digest algorithms are to be applied to the message plaintext.

5.2.2. Match recipient info

The first step in recovering the master key is to determine which (if any) of the recipient entries can be used for decryption by examining the kid fields.

5.2.3. Recover Master Key

Having identified the key exchange to use, the Master key.

5.2.4. Process Enhanced Data Sequences

Having recovered the Master Key, the recipient can decrypt whichever Enhanced Data Sequences it requires.
5.2.5. Signature validation

If validation of one or more signature entries is required, the recipient recovers the signature value from the corresponding Enhanced Data Sequences and performs signature verification according to the specified algorithm.

5.3. Detached Messages

Processing of a detached message is the same as for a consolidated message except for the fact that at least some of the header information is passed out of band with respect to the message. There are thus two sets of headers:

Context Header The header that contains the key exchange (passed out of band with respect to the message).

Message Header Additional header data contained in the message itself.

Message headers take precedence over context headers. Thus if the encryption algorithm field ‘enc’ is specified in both places, the value specified in the message header takes precedence.

5.4. Cloaked Messages

A cloaked message is a DARE message that contains a Cloaked field in the header.

Implementations MAY support generation and parsing of cloaked messages. Implementations SHOULD NOT generate and MAY reject nested cloaked headers unless specifically directed by an application specification.

The use of cloaked headers allows a second layer of key agreement to be specified within the first. The message body is always encrypted under the Master Key corresponding to the innermost key exchange.

Enhanced Data Sequences that are contained in a cloaked header are encrypted under the master key contained in that header.

6. Algorithms

6.1. 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.

7. 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.

7.1. 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.

7.1.1. 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.

7.2. Header and Trailer Classes

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

7.2.1. 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.

7.2.2. 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.

7.3. 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.

7.3.1. 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.

7.3.2. Structure: X509Certificate

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

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

7.3.3. Structure: DARESignature

Inherits: DARESigner

The signature value

SignatureValue: Binary (Optional) The signature value as an Enhanced Data Sequence under the message Master Key.
7.3.4. 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.

8. Security Considerations

8.1. Encryption/Signature nesting

8.2. Side channel

8.3. Salt reuse

9. IANA Considerations

10. Acknowledgements

11. Test Examples

   In the following examples, Alice’s public key parameters are:
{  
"PrivateKeyDH":{  
  "kid":"MDD7L-HSRZY-6DYKQ-7FHIW-PS6I5-VOC4F-A",
  "Domain":"YE6bnq1MlX5ojaJto6PLP_PEwA",
  "Public":"LnHmYAeS4RrQ5r0e2p7U1mpgqFs84ySt0vHm7XexZWjuywdJ6sMG9
_CPAOC6ZSeRgKnh7QkNeYbTutFKJQwOE0KJ8inMFXFHVQ4_xOy6tJOLJg6Km6QNHv
GDNH2yo2zk_r5pSEysWr8p50QZX5vklMlqHlwqQ5nsZJgedW_61L-9T8n6o0QTH
9ZkkDNfILNV7r1isJVAz56u-dpWLN-I-0yRbr9EEMqf3b-MU4dOi131clcXXLxU
mxG44TkRyGfqVLvq41lCyaicGE1SESrdy0T_VMxNn9NO3nm6tGyGrE6jad7-y7w-
P AoMoSZ7PUK-o19hzh58kVftjd1HQ",
  "Private":"SZzvjY85RigTp2CTu_N7M-Cxo2zFakVyd9Da-tQV3A61snj70n2e
JN61Do1txwSv UXUSTzx29b4LskAWv1OamEytaYq9rmlEtboePKLTHFLlfEmno-
xdP3ZbDO5NNnjBE6DYc6eteXWIG-JHMRxvF70NU_fS9kFqg_fLIHqTXDAVrL-L
ZNTybnAJCW38e1zbwerFrsf13mlc9y9TDRJzhM1viApb4107rrNHM2pusgKZMtgys9oday16ar-smLfp118FjwVzot-Pvs-r7sQJlyede-tUhkRzSqbit4fbzpVdPxyJwnQycarQd7BLEVu8YSfxSzsLTpJtdSO"}

Figure 2

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 3

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

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

Figure 4

11.1. Plaintext Message

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

{  
  "DAREMessage":[]
}

Figure 5
11.2. Plaintext Message with EDS

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

{  
    "DAREMessage":{  
        "eds":["iAEAiC1TdWJqZWN0OiBNZXNzYWdlIGIlGFkYXRhIHNob3VsZCBiZSBlbmNyeXB0ZW51aAA",  
            "iAEAiAoyMDE4LTAyLTAxiAA"],  
        "VGhpcyBpcyBhIHRlc3QgbG9uZyBlbm91Z2ggdG8gcmVxdWllyZSBtdWx0aXBzZW5ieSBGbG9ja3M"  
    }  
}

Figure 6

11.3. Encrypted Message

The creator generates a master session key:

0A CE 34 A2 D3 12 1C 06 B0 72 FB 4C 50 47 1F B6  
19 0B E0 06 C5 5E 70 F9 CC 6F 11 D6 71 52 37 26

Figure 7

The creator generates an ephemeral key:

{  
    "PrivateKeyDH":{  
        "kid":"MADSN-KKGHV-3TV37-2U2VV-HT66X-WTYML-A",  
        "Domain":"YE6bnq1MlX5ojaIco6PLP_PewA",  
        "Public":":zTwurVdG03ELMaB-2bAI-pW_Pr0n0fQZBpTmaXbn2o0Glj22hcr7RUYoluMVjCsuQBH14Qrzv7LcCO6vM1u2mQsZdf71gkQ7MhPLgKnnwUE8T-mOP_qn  
        rIe3gOjobKKBt9g_Nv110gmPSN_R1-8xHBCSSVcygQlp20KfXReDwAoNzDm-jj3  
        6sQqbBFrJWjOBWNi56rkcMavHXMvGgK3MoGkAgyeO3_Q4Zwt-Q-Pc8em0reJCjcIUR  
        b_k8m5d3uG9Q06Fs7cFe661ExDcIy7yqqFsp7ss7Pjz3cCX_HqoMmpm34-1FWm6  
        BOODw9LG8UXVn_guJV4BM80JIBUVQ",  
        "Private":":ZJeUWOQwzckelRLLxnz-mSB0N2jRA606BoKoaI4sG2UD-qTv1d0FzU2J5U8Td38UNTn9Cdt-1MmzD3N8ZGqoo0UsyumyeckiMzZodKistHiKvdaGyRO  
        vrQBOVjyf-g_NCPsj-Y0sRyjFcGefmE443QowL-E_rYoH3Az1zLwtWOIDj_Ya3ii  
        sBu9cFEnasSYT4p1RoVFLSwMVQy_f-c0qQqRtFEKYNZKSAXrqkOJ24XChWiNYQgHS  
        PcETUmo77Bh-1YyasLuLugVjK8cEl3SuHn6L2uwq6y0yfAEr1Eq2zWQbgC7maX24M  
        cPhNXE22_it-DEoXKXKTeYJn-717AA"  
    }  
}

Figure 8
The key agreement value is calculated:

```
249775263816487483931896010299097564854454347516831840756543977145862
774009093007097779281910260145784886411647160269599615422500039748492
444179619394328116490036117795948984185835682430902489631851978778082
27885460759871589399243819622482623881775676267763488836284748681710
94358362474393996835460628460290095419061008573813653124401427422872
330169175818408031753756082153864435963738210046522095495994111881975
127191421307826283359719837317669393381464188247341690048493579073631
562885821436525380302849739889927476167265009606070422023062572146801
002551443229480737511384690916393012771976659473341526524
```

Figure 9

The key agreement value is used as the input to a HKDF key derivation function with the info parameter "master".

```
FA CD 02 3D 07 66 BE 56 84 2E 85 8A E1 8A 54 AB
C2 67 5F E4 40 DC CD 0E D8 2D 14 AF 7B A1 64 EA
```

Figure 10

To create the first EDS, a salt value is assigned. In this case a single octet with the value '01'. The salt value is then used to create the encryption key and IV as follows:

Salt:
Example.DareEDSSalt.ToStringBase16FormatHex()

Encryption Key:
Example.DareMessageKeyEncrypt.ToStringBase16FormatHex()

IV:
Example.DareMessageKeyIV.ToStringBase16FormatHex()

Figure 11

The output sequence is the salt followed by the ciphertext:

```
88 40  FE 3E  DB 48 01 BB 46 32 95 31 D2 2B 32 E0
D5 93 BD 3C 25 8O 7E E5 6F 46 BD 84 B8 C8 86 1F
45 88 F2 3C CC F6 3F 13 17 AD 5E 1B 09 53 6C 88
04 AF 31 F9 CE 4F 7F BD 26 C6 65 11 48 B2 74 8B
38 86
```

Figure 12
The completed message is:

```json
{
  "DAREMessage":{[
    "recipients":{[
      "kid":"MDD7L-HSRZY-6DYKQ-7FHIW-PS6I5-VOC4F-A",
      "epk":{
        "PublicDH":{
          "kid":"MADSN-KKGHV-3TV37-2U2VW-UT6BX-WTYML-A",
          "Domain":"YE6bnq1MlX5ojaJto6PLP_PEwA",
          "Public":"zTwurVdG03ELMaB-hAI-pW_PrIn00fQZBpTmaXbnZo0G1j22hcr7RUYoluMVicuQ4H14Qrxv7LccO6vM1kU2mQsZdf71gkQ7MhPLgNkwU8E7mOP_qnrIe3gOjobKKbT9g_Nv110gmpSN_R1-8XxHBCSSVcygQ1pZOKfXRdWAoNzDm-jjh36sQqbBFrJWwjoBWNi56rkcMavHxmVgK3ModKAgYeO3_Q4ZwtQ-Pc8em0reJCjclUR_b_ksm5D3u9cQQ6sFn7cFp661ExDcIyTYqqFsp7ss7PJn3zCX_HQoMpm34-1FwM6BOOWd9LG8uXVn_guJV4BM80JIBUVg"},
      "wmk":"jjd4k069eAyUZUBB6RizaPW9u5b_Fq6PPc9k9Hfjx6JIIwU mLdA"
    ]},
    "iED-PttIAbtGMpUx0isy4NWTjTwlgH7lb0a9hLjIhh9Fi8zPY_ExetXhsJU2yIBK8x-c5Pf70mmxURSLJ0iziG"
  ]}
}
```

Figure 13

11.4. Signed Messages

This is not yet implemented.

12. References

12.1. Normative References

[draft-hallambaker-jsonbcd]

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

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


12.2. URIs


Author’s Address

Phillip Hallam-Baker
Comodo Group Inc.

Email: philliph@comodo.com