Workgroup: Internet Research Task Force (IRTF)
Internet-Draft: draft-irtf-cfrg-hpke-09
Published: 26 May 2021
Intended Status: Informational
Expires: 27 November 2021
Authors: R.L. Barnes (Cisco) K. Bhargavan (Inria) B. Lipp (Inria) C.A. Wood (Cloudflare)

Hybrid Public Key Encryption

Abstract

This document describes a scheme for hybrid public-key encryption (HPKE). This scheme provides authenticated public key encryption of arbitrary-sized plaintexts for a recipient public key. HPKE works for any combination of an asymmetric key encapsulation mechanism (KEM), key derivation function (KDF), and authenticated encryption with additional data (AEAD) encryption function. We provide instantiations of the scheme using widely used and efficient primitives, such as Elliptic Curve Diffie-Hellman key agreement, HKDF, and SHA2.

This document is a product of the Crypto Forum Research Group (CFRG) in the IRTF.

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 27 November 2021.

Copyright Notice

Table of Contents

1. Introduction

Encryption schemes that combine asymmetric and symmetric algorithms have been specified and practiced since the early days of public-key cryptography, e.g., [RFC1421]. Combining the two yields the key management advantages of asymmetric cryptography and the performance benefits of symmetric cryptography. The traditional combination has been "encrypt the symmetric key with the public key." "Hybrid" public-key encryption schemes (HPKE), specified here, take a different approach: "generate the symmetric key and its encapsulation with the public key." Specifically, encrypted messages convey an encryption key encapsulated with a public-key scheme, along with one or more arbitrary-sized ciphertexts encrypted using that key. This type of public key encryption has many applications in practice, including Messaging Layer Security [I-D.ietf-mls-protocol] and TLS Encrypted ClientHello [I-D.ietf-tls-esni].

Currently, there are numerous competing and non-interoperable standards and variants for hybrid encryption, mostly based on ECIES, including ANSI X9.63 (ECIES) [ANSI], IEEE 1363a [IEEE1363], ISO/IEC 18033-2 [ISO], and SECG SEC 1 [SECG]. See [MAEA10] for a thorough comparison. All these existing schemes have problems, e.g., because they rely on outdated primitives, lack proofs of IND-CCA2 security, or fail to provide test vectors.

This document defines an HPKE scheme that provides a subset of the functions provided by the collection of schemes above, but specified with sufficient clarity that they can be interoperably implemented. The HPKE construction defined herein is secure against (adaptive) chosen ciphertext attacks (IND-CCA2 secure) under classical assumptions about the underlying primitives [HPKEAnalysis], [ABHKLR20]. A summary of these analyses is in Section 9.1.

This document represents the consensus of the Crypto Forum Research Group (CFRG).

2. Requirements Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Notation

The following terms are used throughout this document to describe the operations, roles, and behaviors of HPKE:

* (skX, pkX): A Key Encapsulation Mechanism (KEM) key pair used in role X, where X is one of S, R, or E as sender, recipient, and ephemeral, respectively; skX is the private key and pkX is the public key.

* pk(skX): The KEM public key corresponding to the KEM private key skX.

* Sender (S): Role of entity which sends an encrypted message.

* Recipient (R): Role of entity which receives an encrypted message.

* Ephemeral (E): Role of a fresh random value meant for one-time use.

* I2OSP(n, w): Convert non-negative integer n to a w-length, big-endian byte string as described in [RFC8017].

* OS2IP(x): Convert byte string x to a non-negative integer as described in [RFC8017], assuming big-endian byte order.

* concat(x0, ..., xN): Concatenation of byte strings. concat(0x01, 0x0203, 0x040506) = 0x010203040506.

* random(n): A pseudorandom byte string of length n bytes

* xor(a,b): XOR of byte strings; xor(0xF0F0, 0x1234) = 0xE2C4. It is an error to call this function with two arguments of unequal length.

4. Cryptographic Dependencies

HPKE variants rely on the following primitives:

* A Key Encapsulation Mechanism (KEM):

  - GenerateKeyPair(): Randomized algorithm to generate a key pair (skX, pkX).

  - DeriveKeyPair(ikm): Deterministic algorithm to derive a key pair (skX, pkX) from the byte string ikm, where ikm SHOULD have at least Nsk bytes of entropy (see Section 7.1.3 for discussion).

- SerializePublicKey(pkX): Produce a byte string of length Npk encoding the public key pkX.

- DeserializePublicKey(pkXm): Parse a byte string of length Npk to recover a public key. This function can raise a DeserializeError error upon pkXm deserialization failure.

- Encap(pkR): Randomized algorithm to generate an ephemeral, fixed-length symmetric key (the KEM shared secret) and a fixed-length encapsulation of that key that can be decapsulated by the holder of the private key corresponding to pkR.

- Decap(enc, skR): Deterministic algorithm using the private key skR to recover the ephemeral symmetric key (the KEM shared secret) from its encapsulated representation enc. This function can raise a DecapError on decapsulation failure.

- AuthEncap(pkR, skS) (optional): Same as Encap(), and the outputs encode an assurance that the KEM shared secret was generated by the holder of the private key skS.

- AuthDecap(enc, skR, pkS) (optional): Same as Decap(), and the recipient is assured that the KEM shared secret was generated by the holder of the private key skS.

- Nsecret: The length in bytes of a KEM shared secret produced by this KEM.

- Nenc: The length in bytes of an encapsulated key produced by this KEM.

- Npk: The length in bytes of an encoded public key for this KEM.

- Nsk: The length in bytes of an encoded private key for this KEM.

*A Key Derivation Function (KDF):

- Extract(salt, ikm): Extract a pseudorandom key of fixed length Nh bytes from input keying material ikm and an optional byte string salt.

- Expand(prk, info, L): Expand a pseudorandom key prk using optional string info into L bytes of output keying material.

- Nh: The output size of the Extract() function in bytes.

*An AEAD encryption algorithm [RFC5116]:

  - Seal(key, nonce, aad, pt): Encrypt and authenticate plaintext pt with associated data aad using symmetric key key and nonce nonce, yielding ciphertext and tag ct. This function can raise a MessageLimitReachedError upon failure.

  - Open(key, nonce, aad, ct): Decrypt ciphertext and tag ct using associated data aad with symmetric key key and nonce nonce, returning plaintext message pt. This function can raise an OpenError or MessageLimitReachedError upon failure.

  - Nk: The length in bytes of a key for this algorithm.

  - Nn: The length in bytes of a nonce for this algorithm.

Beyond the above, a KEM MAY also expose the following functions, whose behavior is detailed in Section 7.1.2:

*SerializePrivateKey(skX): Produce a byte string of length Nsk encoding the private key skX.

*DeserializePrivateKey(skXm): Parse a byte string of length Nsk to recover a private key. This function can raise a DeserializeError error upon skXm deserialization failure.

A *ciphersuite* is a triple (KEM, KDF, AEAD) containing a choice of algorithm for each primitive.

A set of algorithm identifiers for concrete instantiations of these primitives is provided in Section 7. Algorithm identifier values are two bytes long.

Note that GenerateKeyPair can be implemented as DeriveKeyPair(random(Nsk)).

The notation pk(skX), depending on its use and the KEM and its implementation, is either the computation of the public key using the private key, or just syntax expressing the retrieval of the public key assuming it is stored along with the private key object.

The following two functions are defined to facilitate domain separation of KDF calls as well as context binding:

```
def LabeledExtract(salt, label, ikm):
  labeled_ikm = concat("HPKE-v1", suite_id, label, ikm)
  return Extract(salt, labeled_ikm)

def LabeledExpand(prk, label, info, L):
  labeled_info = concat(I2OSP(L, 2), "HPKE-v1", suite_id,
                        label, info)
  return Expand(prk, labeled_info, L)
```

The value of suite_id depends on where the KDF is used; it is assumed implicit from the implementation and not passed as a parameter. If used inside a KEM algorithm, suite_id MUST start with "KEM" and identify this KEM algorithm; if used in the remainder of HPKE, it MUST start with "HPKE" and identify the entire ciphersuite in use. See sections Section 4.1 and Section 5.1 for details.

4.1. DH-Based KEM

Suppose we are given a KDF, and a Diffie-Hellman group providing the following operations:

- DH(skX, pkY): Perform a non-interactive Diffie-Hellman exchange using the private key skX and public key pkY to produce a Diffie-Hellman shared secret of length Ndh. This function can raise a ValidationError as described in Section 7.1.4.

- Ndh: The length in bytes of a Diffie-Hellman shared secret produced by DH().

- Nsk: The length in bytes of a Diffie-Hellman private key.

Then we can construct a KEM that implements the interface defined in Section 4 called DHKEM(Group, KDF) in the following way, where Group denotes the Diffie-Hellman group and KDF the KDF. The function parameters pkR and pkS are deserialized public keys, and enc is a serialized public key. Since encapsulated keys are Diffie-Hellman public keys in this KEM algorithm, we use SerializePublicKey() and DeserializePublicKey() to encode and decode them, respectively. Npk equals Nenc. GenerateKeyPair() produces a key pair for the Diffie-Hellman group in use. Section 7.1.3 contains the DeriveKeyPair() function specification for DHKEMs defined in this document.

```
def ExtractAndExpand(dh, kem_context):
  eae_prk = LabeledExtract("", "eae_prk", dh)
  shared_secret = LabeledExpand(eae_prk, "shared_secret",
                                kem_context, Nsecret)
  return shared_secret

def Encap(pkR):
  skE, pkE = GenerateKeyPair()
  dh = DH(skE, pkR)
  enc = SerializePublicKey(pkE)

  pkRm = SerializePublicKey(pkR)
  kem_context = concat(enc, pkRm)

  shared_secret = ExtractAndExpand(dh, kem_context)
  return shared_secret, enc

def Decap(enc, skR):
  pkE = DeserializePublicKey(enc)
  dh = DH(skR, pkE)

  pkRm = SerializePublicKey(pk(skR))
  kem_context = concat(enc, pkRm)

  shared_secret = ExtractAndExpand(dh, kem_context)
  return shared_secret

def AuthEncap(pkR, skS):
  skE, pkE = GenerateKeyPair()
  dh = concat(DH(skE, pkR), DH(skS, pkR))
  enc = SerializePublicKey(pkE)

  pkRm = SerializePublicKey(pkR)
  pkSm = SerializePublicKey(pk(skS))
  kem_context = concat(enc, pkRm, pkSm)

  shared_secret = ExtractAndExpand(dh, kem_context)
  return shared_secret, enc

def AuthDecap(enc, skR, pkS):
  pkE = DeserializePublicKey(enc)
  dh = concat(DH(skR, pkE), DH(skR, pkS))

  pkRm = SerializePublicKey(pk(skR))
  pkSm = SerializePublicKey(pkS)
  kem_context = concat(enc, pkRm, pkSm)

  shared_secret = ExtractAndExpand(dh, kem_context)
  return shared_secret
```

The implicit suite_id value used within LabeledExtract and LabeledExpand is defined as follows, where kem_id is defined in Section 7.1:

```
suite_id = concat("KEM", I2OSP(kem_id, 2))
```

The KDF used in DHKEM can be equal to or different from the KDF used in the remainder of HPKE, depending on the chosen variant. Implementations MUST make sure to use the constants (Nh) and function calls (LabeledExtract, LabeledExpand) of the appropriate KDF when implementing DHKEM. See Section 9.3 for a comment on the choice of a KDF for the remainder of HPKE, and Section 9.5 for the rationale of the labels.

For the variants of DHKEM defined in this document, the size Nsecret of the KEM shared secret is equal to the output length of the hash function underlying the KDF. For P-256, P-384 and P-521, the size Ndh of the Diffie-Hellman shared secret is equal to 32, 48, and 66, respectively, corresponding to the x-coordinate of the resulting elliptic curve point [IEEE1363]. For X25519 and X448, the size Ndh of is equal to 32 and 56, respectively (see [RFC7748], Section 5).

It is important to note that the AuthEncap() and AuthDecap() functions of the DHKEM variants defined in this document are vulnerable to key-compromise impersonation (KCI). This means the assurance that the KEM shared secret was generated by the holder of the private key skS does not hold if the recipient private key skR is compromised. See Section 9.1 for more details.

Senders and recipients MUST validate KEM inputs and outputs as described in Section 7.1.

5. Hybrid Public Key Encryption

In this section, we define a few HPKE variants. All variants take a recipient public key and a sequence of plaintexts pt, and produce an encapsulated key enc and a sequence of ciphertexts ct. These outputs are constructed so that only the holder of skR can decapsulate the key from enc and decrypt the ciphertexts. All the algorithms also take an info parameter that can be used to influence the generation of keys (e.g., to fold in identity information) and an aad parameter that provides Additional Authenticated Data to the AEAD algorithm in use.

In addition to the base case of encrypting to a public key, we include three authenticated variants, one which authenticates possession of a pre-shared key, one which authenticates possession of a KEM private key, and one which authenticates possession of both a pre-shared key and a KEM private key. All authenticated variants contribute additional keying material to the encryption operation.

The following one-byte values will be used to distinguish between modes:

Mode	Value
mode_base	0x00
mode_psk	0x01
mode_auth	0x02
mode_auth_psk	0x03

Table 1: HPKE Modes

All these cases follow the same basic two-step pattern:

1. Set up an encryption context that is shared between the sender and the recipient.

2. Use that context to encrypt or decrypt content.

A *context* encodes the AEAD algorithm and key in use, and manages the nonces used so that the same nonce is not used with multiple plaintexts. It also has an interface for exporting secret values, as described in Section 5.3. See Section 5.2 for a description of this structure and its interfaces. HPKE decryption fails when the underlying AEAD decryption fails.

The constructions described here presume that the relevant non-private parameters (enc, psk_id, etc.) are transported between the sender and the recipient by some application making use of HPKE. Moreover, a recipient with more than one public key needs some way of determining which of its public keys was used for the encapsulation operation. As an example, applications may send this information alongside a ciphertext from sender to recipient. Specification of such a mechanism is left to the application. See Section 10 for more details.

Note that some KEMs may not support AuthEncap() or AuthDecap(). For such KEMs, only mode_base or mode_psk are supported. Future specifications which define new KEMs MUST indicate whether these modes are supported. See Section 7.1.6 for more details.

The procedures described in this session are laid out in a Python-like pseudocode. The algorithms in use are left implicit.

5.1. Creating the Encryption Context

The variants of HPKE defined in this document share a common key schedule that translates the protocol inputs into an encryption context. The key schedule inputs are as follows:

- *mode - A one-byte value indicating the HPKE mode, defined in Table 1.

- *shared_secret - A KEM shared secret generated for this transaction.

- *info - Application-supplied information (optional; default value "").

- *psk - A pre-shared key (PSK) held by both the sender and the recipient (optional; default value "").

- *psk_id - An identifier for the PSK (optional; default value "").

Senders and recipients MUST validate KEM inputs and outputs as described in Section 7.1.

The psk and psk_id fields MUST appear together or not at all. That is, if a non-default value is provided for one of them, then the other MUST be set to a non-default value. This requirement is encoded in VerifyPSKInputs() below.

The psk, psk_id, and info fields have maximum lengths that depend on the KDF itself, on the definition of LabeledExtract(), and on the constant labels used together with them. See Section 7.2.1 for precise limits on these lengths.

The key, base_nonce, and exporter_secret computed by the key schedule have the property that they are only known to the holder of the recipient private key, and the entity that used the KEM to generate shared_secret and enc.

In the Auth and AuthPSK modes, the recipient is assured that the sender held the private key skS. This assurance is limited for the DHKEM variants defined in this document because of key-compromise impersonation, as described in Section 4.1 and Section 9.1. If in the PSK and AuthPSK modes, the psk and psk_id arguments are provided as required, then the recipient is assured that the sender held the corresponding pre-shared key. See Section 9.1 for more details.

The HPKE algorithm identifiers, i.e., the KEM kem_id, KDF kdf_id, and AEAD aead_id 2-byte code points as defined in Table 2, Table 3, and Table 5, respectively, are assumed implicit from the implementation and not passed as parameters. The implicit suite_id

value used within LabeledExtract and LabeledExpand is defined based on them as follows:

```
suite_id = concat(
  "HPKE",
  I2OSP(kem_id, 2),
  I2OSP(kdf_id, 2),
  I2OSP(aead_id, 2)
)

default_psk = ""
default_psk_id = ""

def VerifyPSKInputs(mode, psk, psk_id):
  got_psk = (psk != default_psk)
  got_psk_id = (psk_id != default_psk_id)
  if got_psk != got_psk_id:
    raise Exception("Inconsistent PSK inputs")

  if got_psk and (mode in [mode_base, mode_auth]):
    raise Exception("PSK input provided when not needed")
  if (not got_psk) and (mode in [mode_psk, mode_auth_psk]):
    raise Exception("Missing required PSK input")

def KeySchedule<ROLE>(mode, shared_secret, info, psk, psk_id):
  VerifyPSKInputs(mode, psk, psk_id)

  psk_id_hash = LabeledExtract("", "psk_id_hash", psk_id)
  info_hash = LabeledExtract("", "info_hash", info)
  key_schedule_context = concat(mode, psk_id_hash, info_hash)

  secret = LabeledExtract(shared_secret, "secret", psk)

  key = LabeledExpand(secret, "key", key_schedule_context, Nk)
  base_nonce = LabeledExpand(secret, "base_nonce",
                             key_schedule_context, Nn)
  exporter_secret = LabeledExpand(secret, "exp",
                                  key_schedule_context, Nh)

  return Context<ROLE>(key, base_nonce, 0, exporter_secret)
```

The ROLE template parameter is either S or R, depending on the role of sender or recipient, respectively. See Section 5.2 for a discussion of the key schedule output, including the role-specific Context structure and its API.

Note that the key_schedule_context construction in KeySchedule() is equivalent to serializing a structure of the following form in the TLS presentation syntax:

```
struct {
    uint8 mode;
    opaque psk_id_hash[Nh];
    opaque info_hash[Nh];
} KeyScheduleContext;
```

5.1.1. Encryption to a Public Key

The most basic function of an HPKE scheme is to enable encryption to the holder of a given KEM private key. The SetupBaseS() and SetupBaseR() procedures establish contexts that can be used to encrypt and decrypt, respectively, for a given private key.

The KEM shared secret is combined via the KDF with information describing the key exchange, as well as the explicit info parameter provided by the caller.

The parameter pkR is a public key, and enc is an encapsulated KEM shared secret.

```
def SetupBaseS(pkR, info):
  shared_secret, enc = Encap(pkR)
  return enc, KeyScheduleS(mode_base, shared_secret, info,
                           default_psk, default_psk_id)

def SetupBaseR(enc, skR, info):
  shared_secret = Decap(enc, skR)
  return KeyScheduleR(mode_base, shared_secret, info,
                      default_psk, default_psk_id)
```

5.1.2. Authentication using a Pre-Shared Key

This variant extends the base mechanism by allowing the recipient to authenticate that the sender possessed a given PSK. The PSK also improves confidentiality guarantees in certain adversary models, as described in more detail in Section 9.1. We assume that both parties have been provisioned with both the PSK value psk and another byte string psk_id that is used to identify which PSK should be used.

The primary difference from the base case is that the psk and psk_id values are used as ikm inputs to the KDF (instead of using the empty string).

The PSK MUST have at least 32 bytes of entropy and SHOULD be of length Nh bytes or longer. See Section 9.4 for a more detailed discussion.

```
def SetupPSKS(pkR, info, psk, psk_id):
  shared_secret, enc = Encap(pkR)
  return enc, KeyScheduleS(mode_psk, shared_secret, info, psk, psk_id)

def SetupPSKR(enc, skR, info, psk, psk_id):
  shared_secret = Decap(enc, skR)
  return KeyScheduleR(mode_psk, shared_secret, info, psk, psk_id)
```

5.1.3. Authentication using an Asymmetric Key

This variant extends the base mechanism by allowing the recipient to authenticate that the sender possessed a given KEM private key. This assurance is based on the assumption that AuthDecap(enc, skR, pkS) produces the correct KEM shared secret only if the encapsulated value enc was produced by AuthEncap(pkR, skS), where skS is the private key corresponding to pkS. In other words, at most two entities (precisely two, in the case of DHKEM) could have produced this secret, so if the recipient is at most one, then the sender is the other with overwhelming probability.

The primary difference from the base case is that the calls to Encap() and Decap() are replaced with calls to AuthEncap() and AuthDecap(), which add the sender public key to their internal context string. The function parameters pkR and pkS are public keys, and enc is an encapsulated KEM shared secret.

Obviously, this variant can only be used with a KEM that provides AuthEncap() and AuthDecap() procedures.

This mechanism authenticates only the key pair of the sender, not any other identifier. If an application wishes to bind HPKE ciphertexts or exported secrets to another identity for the sender (e.g., an email address or domain name), then this identifier should be included in the info parameter to avoid identity mis-binding issues [IMB].

```
def SetupAuthS(pkR, info, skS):
  shared_secret, enc = AuthEncap(pkR, skS)
  return enc, KeyScheduleS(mode_auth, shared_secret, info,
                           default_psk, default_psk_id)

def SetupAuthR(enc, skR, info, pkS):
  shared_secret = AuthDecap(enc, skR, pkS)
  return KeyScheduleR(mode_auth, shared_secret, info,
                      default_psk, default_psk_id)
```

5.1.4. Authentication using both a PSK and an Asymmetric Key

This mode is a straightforward combination of the PSK and authenticated modes. The PSK is passed through to the key schedule

as in the former, and as in the latter, we use the authenticated KEM variants.

```
def SetupAuthPSKS(pkR, info, psk, psk_id, skS):
  shared_secret, enc = AuthEncap(pkR, skS)
  return enc, KeyScheduleS(mode_auth_psk, shared_secret, info,
                           psk, psk_id)

def SetupAuthPSKR(enc, skR, info, psk, psk_id, pkS):
  shared_secret = AuthDecap(enc, skR, pkS)
  return KeyScheduleR(mode_auth_psk, shared_secret, info,
                      psk, psk_id)
```

The PSK MUST have at least 32 bytes of entropy and SHOULD be of length Nh bytes or longer. See Section 9.4 for a more detailed discussion.

5.2. Encryption and Decryption

HPKE allows multiple encryption operations to be done based on a given setup transaction. Since the public-key operations involved in setup are typically more expensive than symmetric encryption or decryption, this allows applications to amortize the cost of the public-key operations, reducing the overall overhead.

In order to avoid nonce reuse, however, this encryption must be stateful. Each of the setup procedures above produces a role-specific context object that stores the AEAD and Secret Export parameters. The AEAD parameters consist of:

- The AEAD algorithm in use
- A secret key
- A base nonce base_nonce
- A sequence number (initially 0)

The Secret Export parameters consist of:

- The HPKE ciphersuite in use
- An exporter_secret used for the Secret Export interface; see Section 5.3

All these parameters except the AEAD sequence number are constant. The sequence number provides nonce uniqueness: The nonce used for each encryption or decryption operation is the result of XORing base_nonce with the current sequence number, encoded as a big-endian integer of the same length as base_nonce. Implementations MAY use a

sequence number that is shorter than the nonce length (padding on the left with zero), but MUST raise an error if the sequence number overflows.

Encryption is unidirectional from sender to recipient. The sender's context can encrypt a plaintext pt with associated data aad as follows:

```
def ContextS.Seal(aad, pt):
  ct = Seal(self.key, self.ComputeNonce(self.seq), aad, pt)
  self.IncrementSeq()
  return ct
```

The recipient's context can decrypt a ciphertext ct with associated data aad as follows:

```
def ContextR.Open(aad, ct):
  pt = Open(self.key, self.ComputeNonce(self.seq), aad, ct)
  if pt == OpenError:
    raise OpenError
  self.IncrementSeq()
  return pt
```

Each encryption or decryption operation increments the sequence number for the context in use. The per-message nonce and sequence number increment details are as follows:

```
def Context<ROLE>.ComputeNonce(seq):
  seq_bytes = I2OSP(seq, Nn)
  return xor(self.base_nonce, seq_bytes)

def Context<ROLE>.IncrementSeq():
  if self.seq >= (1 << (8*Nn)) - 1:
    raise MessageLimitReachedError
  self.seq += 1
```

The sender's context MUST NOT be used for decryption. Similarly, the recipient's context MUST NOT be used for encryption. Higher-level protocols re-using the HPKE key exchange for more general purposes can derive separate keying material as needed using use the Secret Export interface; see Section 5.3 and Section 9.7 for more details.

It is up to the application to ensure that encryptions and decryptions are done in the proper sequence, so that encryption and decryption nonces align. If ContextS.Seal() or ContextR.Open() would cause the seq field to overflow, then the implementation MUST fail with an error. (In the pseudocode below, Context<ROLE>.IncrementSeq() fails with an error when seq overflows, which causes ContextS.Seal() and ContextR.Open() to fail

accordingly.) Note that the internal Seal() and Open() calls inside correspond to the context's AEAD algorithm.

5.3. Secret Export

HPKE provides an interface for exporting secrets from the encryption context using a variable-length PRF, similar to the TLS 1.3 exporter interface (see [RFC8446], Section 7.5). This interface takes as input a context string exporter_context and a desired length L in bytes, and produces a secret derived from the internal exporter secret using the corresponding KDF Expand function. For the KDFs defined in this specification, L has a maximum value of 255*Nh. Future specifications which define new KDFs MUST specify a bound for L.

The exporter_context field has a maximum length that depends on the KDF itself, on the definition of LabeledExpand(), and on the constant labels used together with them. See Section 7.2.1 for precise limits on this length.

```
def Context.Export(exporter_context, L):
  return LabeledExpand(self.exporter_secret, "sec",
                       exporter_context, L)
```

Applications that do not use the encryption API in Section 5.2 can use the export-only AEAD ID 0xFFFF when computing the key schedule. Such applications can avoid computing the key and base_nonce values in the key schedule, as they are not used by the Export interface described above.

6. Single-Shot APIs

6.1. Encryption and Decryption

In many cases, applications encrypt only a single message to a recipient's public key. This section provides templates for HPKE APIs that implement stateless "single-shot" encryption and decryption using APIs specified in Section 5.1.1 and Section 5.2:

```
def Seal<MODE>(pkR, info, aad, pt, ...):
  enc, ctx = Setup<MODE>S(pkR, info, ...)
  ct = ctx.Seal(aad, pt)
  return enc, ct

def Open<MODE>(enc, skR, info, aad, ct, ...):
  ctx = Setup<MODE>R(enc, skR, info, ...)
  return ctx.Open(aad, ct)
```

The MODE template parameter is one of Base, PSK, Auth, or AuthPSK. The optional parameters indicated by "..." depend on MODE and may be

empty. SetupBase(), for example, has no additional parameters. SealAuthPSK() and OpenAuthPSK() would be implemented as follows:

```
def SealAuthPSK(pkR, info, aad, pt, psk, psk_id, skS):
  enc, ctx = SetupAuthPSKS(pkR, info, psk, psk_id, skS)
  ct = ctx.Seal(aad, pt)
  return enc, ct

def OpenAuthPSK(enc, skR, info, aad, ct, psk, psk_id, pkS):
  ctx = SetupAuthPSKR(enc, skR, info, psk, psk_id, pkS)
  return ctx.Open(aad, ct)
```

6.2. Secret Export

Applications may also want to derive a secret known only to a given recipient. This section provides templates for HPKE APIs that implement stateless "single-shot" secret export using APIs specified in Section 5.3:

```
def SendExport<MODE>(pkR, info, exporter_context, L, ...):
  enc, ctx = Setup<MODE>S(pkR, info, ...)
  exported = ctx.Export(exporter_context, L)
  return enc, exported

def ReceiveExport<MODE>(enc, skR, info, exporter_context, L, ...):
  ctx = Setup<MODE>R(enc, skR, info, ...)
  return ctx.Export(exporter_context, L)
```

As in Section 6.1, the MODE template parameter is one of Base, PSK, Auth, or AuthPSK. The optional parameters indicated by "..." depend on MODE and may be empty.

7. Algorithm Identifiers

7.1. Key Encapsulation Mechanisms (KEMs)

Value	KEM	Nsecret	Nenc	Npk	Nsk	Auth	Reference
0x0000	(reserved)	N/A	N/A	N/A	N/A	yes	N/A
0x0010	DHKEM(P-256, HKDF-SHA256)	32	65	65	32	yes	[NISTCurves], [RFC5869]
0x0011	DHKEM(P-384, HKDF-SHA384)	48	97	97	48	yes	[NISTCurves], [RFC5869]
0x0012	DHKEM(P-521, HKDF-SHA512)	64	133	133	66	yes	[NISTCurves], [RFC5869]
0x0020	DHKEM(X25519, HKDF-SHA256)	32	32	32	32	yes	[RFC7748], [RFC5869]
0x0021	DHKEM(X448, HKDF-SHA512)	64	56	56	56	yes	[RFC7748], [RFC5869]

Table 2: KEM IDs

The Auth column indicates if the KEM algorithm provides the AuthEncap()/AuthDecap() interface. The meaning of all other columns is explained in Section 11.1.

7.1.1. SerializePublicKey and DeserializePublicKey

For P-256, P-384 and P-521, the SerializePublicKey() function of the KEM performs the uncompressed Elliptic-Curve-Point-to-Octet-String conversion according to [SECG]. DeserializePublicKey() performs the uncompressed Octet-String-to-Elliptic-Curve-Point conversion.

For X25519 and X448, the SerializePublicKey() and DeserializePublicKey() functions are the identity function, since these curves already use fixed-length byte strings for public keys.

Some deserialized public keys MUST be validated before they can be used. See Section 7.1.4 for specifics.

7.1.2. SerializePrivateKey and DeserializePrivateKey

As per [SECG], P-256, P-384, and P-521 private keys are field elements in the scalar field of the curve being used. For this section, and for Section 7.1.3, it is assumed that implementers of ECDH over these curves use an integer representation of private keys that is compatible with the OS2IP() function.

For P-256, P-384 and P-521, the SerializePrivateKey() function of the KEM performs the Field-Element-to-Octet-String conversion according to [SECG]. If the private key is an integer outside the range [0, order-1], where order is the order of the curve being used, the private key MUST be reduced to its representative in [0, order-1] before being serialized. DeserializePrivateKey() performs the Octet-String-to-Field-Element conversion according to [SECG].

For X25519 and X448, private keys are identical to their byte string representation, so little processing has to be done. The SerializePrivateKey() function MUST clamp its output and DeserializePrivateKey() MUST clamp its input, where *clamping* refers to the bitwise operations performed on k in the decodeScalar25519() and decodeScalar448() functions defined in section 5 of [RFC7748].

To catch invalid keys early on, implementers of DHKEMs SHOULD check that deserialized private keys are not equivalent to 0 (mod order), where order is the order of the DH group. Note that this property is trivially true for X25519 and X448 groups, since clamped values can never be 0 (mod order).

7.1.3. DeriveKeyPair

The keys that DeriveKeyPair() produces have only as much entropy as the provided input keying material. For a given KEM, the ikm parameter given to DeriveKeyPair() SHOULD have length at least Nsk, and SHOULD have at least Nsk bytes of entropy.

All invocations of KDF functions (such as LabeledExtract or LabeledExpand) in any DHKEM's DeriveKeyPair() function use the DHKEM's associated KDF (as opposed to the ciphersuite's KDF).

For P-256, P-384 and P-521, the DeriveKeyPair() function of the KEM performs rejection sampling over field elements:

```
def DeriveKeyPair(ikm):
  dkp_prk = LabeledExtract("", "dkp_prk", ikm)
  sk = 0
  counter = 0
  while sk == 0 or sk >= order:
    if counter > 255:
      raise DeriveKeyPairError
    bytes = LabeledExpand(dkp_prk, "candidate",
                          I2OSP(counter, 1), Nsk)
    bytes[0] = bytes[0] & bitmask
    sk = OS2IP(bytes)
    counter = counter + 1
  return (sk, pk(sk))
```

order is the order of the curve being used (see section D.1.2 of [NISTCurves]), and is listed below for completeness.

```
P-256:
0xffffffff00000000ffffffffffffffffbce6faada7179e84f3b9cac2fc632551

P-384:
0xffffffffffffffffffffffffffffffffffffffffffffffffc7634d81f4372ddf
  581a0db248b0a77aecec196accc52973

P-521:
0x01ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
  fa51868783bf2f966b7fcc0148f709a5d03bb5c9b8899c47aebb6fb71e91386409
```

bitmask is defined to be 0xFF for P-256 and P-384, and 0x01 for P-521. The precise likelihood of DeriveKeyPair() failing with DeriveKeyPairError depends on the group being used, but it is negligibly small in all cases.

For X25519 and X448, the DeriveKeyPair() function applies a KDF to the input:

```
def DeriveKeyPair(ikm):
  dkp_prk = LabeledExtract("", "dkp_prk", ikm)
  sk = LabeledExpand(dkp_prk, "sk", "", Nsk)
  return (sk, pk(sk))
```

7.1.4. Validation of Inputs and Outputs

The following public keys are subject to validation if the group requires public key validation: the sender MUST validate the recipient's public key pkR; the recipient MUST validate the ephemeral public key pkE; in authenticated modes, the recipient MUST validate the sender's static public key pkS. Validation failure yields a ValidationError.

For P-256, P-384 and P-521, senders and recipients MUST perform partial public-key validation on all public key inputs, as defined in section 5.6.2.3.4 of [keyagreement]. This includes checking that the coordinates are in the correct range, that the point is on the curve, and that the point is not the point at infinity. Additionally, senders and recipients MUST ensure the Diffie-Hellman shared secret is not the point at infinity.

For X25519 and X448, public keys and Diffie-Hellman outputs MUST be validated as described in [RFC7748]. In particular, recipients MUST check whether the Diffie-Hellman shared secret is the all-zero value and abort if so.

7.1.5. KEM Key Reuse

An ikm input to DeriveKeyPair() (Section 7.1.3) MUST NOT be reused elsewhere, in particular not with DeriveKeyPair() of a different KEM.

The randomness used in Encap() and AuthEncap() to generate the KEM shared secret or its encapsulation MUST NOT be reused elsewhere.

As a sender or recipient KEM key pair works with all modes, it can be used with multiple modes in parallel. HPKE is constructed to be secure in such settings due to domain separation using the suite_id variable. However, there is no formal proof of security at the time of writing; [HPKEAnalysis] and [ABHKLR20] only analyze isolated modes.

7.1.6. Future KEMs

Section 9.2 lists security requirements on a KEM used within HPKE.

The AuthEncap() and AuthDecap() functions are OPTIONAL. If a KEM algorithm does not provide them, only the Base and PSK modes of HPKE

are supported. Future specifications which define new KEMs MUST indicate whether or not Auth and AuthPSK modes are supported.

A KEM algorithm may support different encoding algorithms, with different output lengths, for KEM public keys. Such KEM algorithms MUST specify only one encoding algorithm whose output length is Npk.

7.2. Key Derivation Functions (KDFs)

Value	KDF	Nh	Reference
0x0000	(reserved)	N/A	N/A
0x0001	HKDF-SHA256	32	[RFC5869]
0x0002	HKDF-SHA384	48	[RFC5869]
0x0003	HKDF-SHA512	64	[RFC5869]

Table 3: KDF IDs

7.2.1. Input Length Restrictions

This document defines LabeledExtract() and LabeledExpand() based on the KDFs listed above. These functions add prefixes to their respective inputs ikm and info before calling the KDF's Extract() and Expand() functions. This leads to a reduction of the maximum input length that is available for the inputs psk, psk_id, info, exporter_context, ikm, i.e., the variable-length parameters provided by HPKE applications. The following table lists the maximum allowed lengths of these fields for the KDFs defined in this document, as inclusive bounds in bytes:

Input	HKDF-SHA256	HKDF-SHA384	HKDF-SHA512
psk	2^{61} - 88	2^{125} - 152	2^{125} - 152
psk_id	2^{61} - 93	2^{125} - 157	2^{125} - 157
info	2^{61} - 91	2^{125} - 155	2^{125} - 155
exporter_context	2^{61} - 120	2^{125} - 200	2^{125} - 216
ikm (DeriveKeyPair)	2^{61} - 84	2^{125} - 148	2^{125} - 148

Table 4: Application Input Limits

This shows that the limits are only marginally smaller than the maximum input length of the underlying hash function; these limits are large and unlikely to be reached in practical applications. Future specifications which define new KDFs MUST specify bounds for these variable-length parameters.

The RECOMMENDED limit for these values is 64 bytes. This would enable interoperability with implementations that statically allocate memory for these inputs to avoid memory allocations.

The values for psk, psk_id, info, and ikm which are inputs to LabeledExtract() were computed with the following expression:

```
max_size_hash_input - Nb - size_version_label -
    size_suite_id - size_input_label
```

The value for exporter_context which is an input to LabeledExpand() was computed with the following expression:

```
max_size_hash_input - Nb - Nh - size_version_label -
    size_suite_id - size_input_label - 2 - 1
```

In these equations, max_size_hash_input is the maximum input length of the underlying hash function in bytes, Nb is the block size of the underlying hash function in bytes, size_version_label is the size of "HPKE-v1" in bytes and equals 7, size_suite_id is the size of the suite_id in bytes and equals 5 for DHKEM (relevant for ikm) and 10 for the remainder of HPKE (relevant for psk, psk_id, info, exporter_context), and size_input_label is the size in bytes of the label used as parameter to LabeledExtract() or LabeledExpand(), the maximum of which is 13 across all labels in this document.

7.3. Authenticated Encryption with Associated Data (AEAD) Functions

Value	AEAD	Nk	Nn	Reference
0x0000	(reserved)	N/A	N/A	N/A
0x0001	AES-128-GCM	16	12	[GCM]
0x0002	AES-256-GCM	32	12	[GCM]
0x0003	ChaCha20Poly1305	32	12	[RFC8439]
0xFFFF	Export-only	N/A	N/A	[[RFCXXXX]]

Table 5: AEAD IDs

The 0xFFFF AEAD ID is reserved for applications which only use the Export interface; see Section 5.3 for more details.

8. API Considerations

The high-level HPKE APIs specified in this document are all fallible. For example, Decap() can fail if the encapsulated key enc is invalid, and Open() may fail if ciphertext decryption fails. The explicit errors generated throughout this specification, along with the conditions that lead to each error, are as follows:

- *ValidationError: KEM input or output validation failure; Section 4.1.

- *DeserializeError: Public or private key deserialization failure; Section 4.

- DecapError: Decap() failure; Section 4.

- OpenError: Context AEAD Open() failure; Section 4 and Section 5.2.

- MessageLimitReachedError: Context AEAD sequence number overflow; Section 4 and Section 5.2.

- DeriveKeyPairError: Key pair derivation failure; Section 7.1.3.

Implicit errors may also occur. As an example, certain classes of failures, e.g., malformed recipient public keys, may not yield explicit errors. For example, for the DHKEM variant described in this specification, the Encap() algorithm fails when given an invalid recipient public key. However, other KEM algorithms may not have an efficient algorithm for verifying the validity of public keys. As a result, an equivalent error may not manifest until AEAD decryption at the recipient. As another example, DHKEM's AuthDecap() function will produce invalid output if given the wrong sender public key. This error is not detectable until subsequent AEAD decryption.

The errors in this document are meant as a guide to implementors. They are not an exhaustive list of all the errors an implementation might emit. For example, future KEMs might have internal failure cases, or an implementation might run out of memory.

Applications using HPKE APIs should not assume that the errors here are complete, nor should they assume certain classes of errors will always manifest the same way for all ciphersuites. For example, the DHKEM specified in this document will emit a DeserializationError or ValidationError if a KEM public key is invalid. However, a new KEM might not have an efficient algorithm for determining whether or not a public key is valid. In this case, an invalid public key might instead yield an OpenError when trying to decrypt a ciphertext.

9. Security Considerations

9.1. Security Properties

HPKE has several security goals, depending on the mode of operation, against active and adaptive attackers that can compromise partial secrets of senders and recipients. The desired security goals are detailed below:

- Message secrecy: Confidentiality of the sender's messages against chosen ciphertext attacks

* Export key secrecy: Indistinguishability of each export secret from a uniformly random bitstring of equal length, i.e., Context.Export is a variable-length PRF

* Sender authentication: Proof of sender origin for PSK, Auth, and AuthPSK modes

These security goals are expected to hold for any honest sender and honest recipient keys, as well as if the honest sender and honest recipient keys are the same.

As noted in Section 9.6.2, HPKE does not provide forward secrecy. In the Base and Auth modes, the secrecy properties are only expected to hold if the recipient private key skR is not compromised at any point in time. In the PSK and AuthPSK modes, the secrecy properties are expected to hold if the recipient private key skR and the pre-shared key are not both compromised at any point in time.

In the Auth mode, sender authentication is generally expected to hold if the sender private key skS is not compromised at the time of message reception. In the AuthPSK mode, sender authentication is generally expected to hold if at the time of message reception, the sender private key skS and the pre-shared key are not both compromised.

HPKE mitigates malleability problems (called benign malleability [SECG]) in prior public key encryption standards based on ECIES by including all public keys in the context key schedule.

9.1.1. Key-Compromise Impersonation

The DHKEM variants defined in this document are vulnerable to key-compromise impersonation attacks [BJM97], which means that sender authentication cannot be expected to hold in the Auth mode if the recipient private key skR is compromised, and in the AuthPSK mode if the pre-shared key and the recipient private key skR are both compromised. NaCl's box interface [NaCl] has the same issue. At the same time, this enables repudiability.

As shown by [ABHKLR20], key-compromise impersonation attacks are generally possible on HPKE because KEM ciphertexts are not bound to HPKE messages. An adversary who knows a recipient's private key can decapsulate an observed KEM ciphertext, compute the key schedule, and encrypt an arbitrary message that the recipient will accept as coming from the original sender. Importantly, this is possible even with a KEM that is resistant to key-compromise impersonation attacks. As a result, mitigating this issue requires fundamental changes that are out-of-scope of this specification.

Applications that require resistance against key-compromise impersonation SHOULD take extra steps to prevent this attack. One possibility is to produce a digital signature over (enc, ct) tuples using a sender's private key - where ct is an AEAD ciphertext produced by the single-shot or multi-shot API, and enc the corresponding KEM encapsulated key.

Given these properties, pre-shared keys strengthen both the authentication and the secrecy properties in certain adversary models. One particular example in which this can be useful is a hybrid quantum setting: if a non-quantum-resistant KEM used with HPKE is broken by a quantum computer, the security properties are preserved through the use of a pre-shared key. This assumes that the pre-shared key has not been compromised, as described in [WireGuard].

9.1.2. Computational Analysis

It is shown in [CS01] that a hybrid public-key encryption scheme of essentially the same form as the Base mode described here is IND-CCA2-secure as long as the underlying KEM and AEAD schemes are IND-CCA2-secure. Moreover, it is shown in [HHK06] that IND-CCA2 security of the KEM and the data encapsulation mechanism are necessary conditions to achieve IND-CCA2 security for hybrid public-key encryption. The main difference between the scheme proposed in [CS01] and the Base mode in this document (both named HPKE) is that we interpose some KDF calls between the KEM and the AEAD. Analyzing the HPKE Base mode instantiation in this document therefore requires verifying that the additional KDF calls do not cause the IND-CCA2 property to fail, as well as verifying the additional export key secrecy property.

Analysis of the PSK, Auth, and AuthPSK modes defined in this document additionally requires verifying the sender authentication property. While the PSK mode just adds supplementary keying material to the key schedule, the Auth and AuthPSK modes make use of a non-standard authenticated KEM construction. Generally, the authenticated modes of HPKE can be viewed and analyzed as flavors of signcryption [SigncryptionDZ10].

A preliminary computational analysis of all HPKE modes has been done in [HPKEAnalysis], indicating asymptotic security for the case where the KEM is DHKEM, the AEAD is any IND-CPA and INT-CTXT-secure scheme, and the DH group and KDF satisfy the following conditions:

* DH group: The gap Diffie-Hellman (GDH) problem is hard in the appropriate subgroup [GAP].

* Extract() and Expand() (in DHKEM): Extract() can be modeled as a random oracle. Expand() can be modeled as a pseudorandom function, wherein the first argument is the key.

* Extract() and Expand() (elsewhere): Extract() can be modeled as a random oracle. Expand() can be modeled as a pseudorandom function, wherein the first argument is the key.

In particular, the KDFs and DH groups defined in this document (see Section 7.2 and Section 7.1) satisfy these properties when used as specified. The analysis in [HPKEAnalysis] demonstrates that under these constraints, HPKE continues to provide IND-CCA2 security, and provides the additional properties noted above. Also, the analysis confirms the expected properties hold under the different key compromise cases mentioned above. The analysis considers a sender that sends one message using the encryption context, and additionally exports two independent secrets using the secret export interface.

The table below summarizes the main results from [HPKEAnalysis]. N/A means that a property does not apply for the given mode, whereas y means the given mode satisfies the property.

Variant	Message Sec.	Export Sec.	Sender Auth.
Base	y	y	N/A
PSK	y	y	y
Auth	y	y	y
AuthPSK	y	y	y

Table 6

If non-DH-based KEMs are to be used with HPKE, further analysis will be necessary to prove their security. The results from [CS01] provide some indication that any IND-CCA2-secure KEM will suffice here, but are not conclusive given the differences in the schemes.

A detailed computational analysis of HPKE's Auth mode single-shot encryption API has been done in [ABHKLR20]. The paper defines security notions for authenticated KEMs and for authenticated public key encryption, using the outsider and insider security terminology known from signcryption [SigncryptionDZ10]. The analysis proves that DHKEM's AuthEncap()/AuthDecap() interface fulfills these notions for all Diffie-Hellman groups specified in this document, and indicates exact security bounds, under the assumption that the gap Diffie-Hellman (GDH) problem is hard in the appropriate subgroup [GAP], and that HKDF can be modeled as a random oracle.

Further, [ABHKLR20] proves composition theorems, showing that HPKE's Auth mode fulfills the security notions of authenticated public key encryption for all KDFs and AEAD schemes specified in this document,

given any authenticated KEM satisfying the previously defined security notions for authenticated KEMs. The theorems assume that the KEM is perfectly correct; they could easily be adapted to work with KEMs that have a non-zero but negligible probability for decryption failure. The assumptions on the KDF are that Extract() and Expand() can be modeled as pseudorandom functions wherein the first argument is the key, respectively. The assumption for the AEAD is IND-CPA and IND-CTXT security.

In summary, the analysis in [ABHKLR20] proves that the single-shot encryption API of HPKE's Auth mode satisfies the desired message confidentiality and sender authentication properties listed at the beginning of this section; it does not consider multiple messages, nor the secret export API.

9.1.3. Post-Quantum Security

All of [CS01], [HPKEAnalysis], and [ABHKLR20] are premised on classical security models and assumptions, and do not consider adversaries capable of quantum computation. A full proof of post-quantum security would need to take appropriate security models and assumptions into account, in addition to simply using a post-quantum KEM. However, the composition theorems from [ABHKLR20] for HPKE's Auth mode only make standard assumptions (i.e., no random oracle assumption) that are expected to hold against quantum adversaries (although with slightly worse bounds). Thus, these composition theorems, in combination with a post-quantum-secure authenticated KEM, guarantee the post-quantum security of HPKE's Auth mode.

In future work, the analysis from [ABHKLR20] can be extended to cover HPKE's other modes and desired security properties. The hybrid quantum-resistance property described above, which is achieved by using the PSK or AuthPSK mode, is not proven in [HPKEAnalysis] because this analysis requires the random oracle model; in a quantum setting, this model needs adaption to, for example, the quantum random oracle model.

9.2. Security Requirements on a KEM used within HPKE

A KEM used within HPKE MUST allow HPKE to satisfy its desired security properties described in Section 9.1. Section 9.5 lists requirements concerning domain separation.

In particular, the KEM shared secret MUST be a uniformly random byte string of length Nsecret. This means, for instance, that it would not be sufficient if the KEM shared secret is only uniformly random as an element of some set prior to its encoding as byte string.

9.2.1. Encap/Decap Interface

As mentioned in Section 9, [CS01] provides some indications that if the KEM's Encap()/Decap() interface (which is used in the Base and PSK modes), is IND-CCA2-secure, HPKE is able to satisfy its desired security properties. An appropriate definition of IND-CCA2-security for KEMs can be found in [CS01] and [BHK09].

9.2.2. AuthEncap/AuthDecap Interface

The analysis of HPKE's Auth mode single-shot encryption API in [ABHKLR20] provides composition theorems that guarantee that HPKE's Auth mode achieves its desired security properties if the KEM's AuthEncap()/AuthDecap() interface satisfies multi-user Outsider-CCA, Outsider-Auth, and Insider-CCA security as defined in the same paper.

Intuitively, Outsider-CCA security formalizes confidentiality, and Outsider-Auth security formalizes authentication of the KEM shared secret in case none of the sender or recipient private keys are compromised. Insider-CCA security formalizes confidentiality of the KEM shared secret in case the sender private key is known or chosen by the adversary. (If the recipient private key is known or chosen by the adversary, confidentiality is trivially broken, because then the adversary knows all secrets on the recipient's side).

An Insider-Auth security notion would formalize authentication of the KEM shared secret in case the recipient private key is known or chosen by the adversary. (If the sender private key is known or chosen by the adversary, it can create KEM ciphertexts in the name of the sender). Because of the generic attack on an analogous Insider-Auth security notion of HPKE described in Section 9.1, a definition of Insider-Auth security for KEMs used within HPKE is not useful.

9.3. Security Requirements on a KDF

The choice of the KDF for the remainder of HPKE SHOULD be made based on the security level provided by the KEM and, if applicable, by the PSK. The KDF SHOULD have at least have the security level of the KEM and SHOULD at least have the security level provided by the PSK.

9.4. Pre-Shared Key Recommendations

In the PSK and AuthPSK modes, the PSK MUST have at least 32 bytes of entropy and SHOULD be of length Nh bytes or longer. Using a PSK longer than 32 bytes but shorter than Nh bytes is permitted.

HPKE is specified to use HKDF as key derivation function. HKDF is not designed to slow down dictionary attacks, see [RFC5869]. Thus,

HPKE's PSK mechanism is not suitable for use with a low-entropy password as the PSK: in scenarios in which the adversary knows the KEM shared secret shared_secret and has access to an oracle that allows to distinguish between a good and a wrong PSK, it can perform PSK-recovering attacks. This oracle can be the decryption operation on a captured HPKE ciphertext or any other recipient behavior which is observably different when using a wrong PSK. The adversary knows the KEM shared secret shared_secret if it knows all KEM private keys of one participant. In the PSK mode this is trivially the case if the adversary acts as sender.

To recover a lower entropy PSK, an attacker in this scenario can trivially perform a dictionary attack. Given a set S of possible PSK values, the attacker generates an HPKE ciphertext for each value in S, and submits the resulting ciphertexts to the oracle to learn which PSK is being used by the recipient. Further, because HPKE uses AEAD schemes that are not key-committing, an attacker can mount a partitioning oracle attack [LGR20] which can recover the PSK from a set of S possible PSK values, with |S| = m*k, in roughly m + log k queries to the oracle using ciphertexts of length proportional to k, the maximum message length in blocks. The PSK must therefore be chosen with sufficient entropy so that m + log k is prohibitive for attackers (e.g., 2^128). Future specifications can define new AEAD algorithms which are key-committing.

9.5. Domain Separation

HPKE allows combining a DHKEM variant DHKEM(Group, KDF') and a KDF such that both KDFs are instantiated by the same KDF. By design, the calls to Extract() and Expand() inside DHKEM and the remainder of HPKE use separate input domains. This justifies modeling them as independent functions even if instantiated by the same KDF. This domain separation between DHKEM and the remainder of HPKE is achieved by the suite_id values in LabeledExtract() and LabeledExpand(): The values used (KEM... in DHKEM and HPKE... in the remainder of HPKE) are prefix-free (a set is prefix-free if no element is a prefix of another within the set).

Future KEM instantiations MUST ensure, should Extract() and Expand() be used internally, that they can be modeled as functions independent from the invocations of Extract() and Expand() in the remainder of HPKE. One way to ensure this is by using LabeledExtract() and LabeledExpand() with a suite_id as defined in Section 4, which will ensure input domain separation as outlined above. Particular attention needs to be paid if the KEM directly invokes functions that are used internally in HPKE's Extract() or Expand(), such as Hash() and HMAC() in the case of HKDF. It MUST be ensured that inputs to these invocations cannot collide with inputs to the internal invocations of these functions inside Extract() or

Expand(). In HPKE's KeySchedule() this is avoided by using Extract() instead of Hash() on the arbitrary-length inputs info and psk_id.

The string literal "HPKE-v1" used in LabeledExtract() and LabeledExpand() ensures that any secrets derived in HPKE are bound to the scheme's name and version, even when possibly derived from the same Diffie-Hellman or KEM shared secret as in another scheme or version.

9.6. Application Embedding

HPKE is designed to be a fairly low-level mechanism. As a result, it assumes that certain properties are provided by the application in which HPKE is embedded, and leaves certain security properties to be provided by other mechanisms.

9.6.1. External Requirements

The primary requirement that HPKE imposes on applications is the requirement that ciphertexts MUST be presented to ContextR.Open() in the same order in which they were generated by ContextS.Seal(). When the single-shot API is used (see Section 6), this is trivially true (since there is only ever one ciphertext. Applications that allow for multiple invocations of Open() / Seal() on the same context MUST enforce the ordering property described above.

Ordering requirements of this character are usually fulfilled by providing a sequence number in the framing of encrypted messages. Whatever information is used to determine the ordering of HPKE-encrypted messages SHOULD be included in the AAD passed to ContextS.Seal() and ContextR.Open(). The specifics of this scheme are up to the application.

HPKE is not tolerant of lost messages. Applications MUST be able to detect when a message has been lost. When an unrecoverable loss is detected, the application MUST discard any associated HPKE context.

9.6.2. Non-Goals

HPKE does not provide several features that a more high-level protocol might provide, for example:

* Downgrade prevention - HPKE assumes that the sender and recipient agree on what algorithms to use. Depending on how these algorithms are negotiated, it may be possible for an intermediary to force the two parties to use suboptimal algorithms.

* Replay protection - The requirement that ciphertexts be presented to the ContextR.Open() function in the same order they were generated by ContextS.Seal() provides a degree of replay

protection within a stream of ciphertexts resulting from a given context. HPKE provides no other replay protection.

* Forward secrecy - HPKE ciphertexts are not forward-secure. In the Base and Auth modes, a given ciphertext can be decrypted if the recipient's public encryption key is compromised. In the PSK and AuthPSK modes, a given ciphertext can be decrypted if the recipient's private key and the PSK are compromised.

* Hiding plaintext length - AEAD ciphertexts produced by HPKE do not hide the plaintext length. Applications requiring this level of privacy should use a suitable padding mechanism. See [I-D.ietf-tls-esni] and [RFC8467] for examples of protocol-specific padding policies.

9.7. Bidirectional Encryption

As discussed in Section 5.2, HPKE encryption is unidirectional from sender to recipient. Applications that require bidirectional encryption can derive necessary keying material with the Secret Export interface Section 5.3. The type and length of such keying material depends on the application use case.

As an example, if an application needs AEAD encryption from recipient to sender, it can derive a key and nonce from the corresponding HPKE context as follows:

```
key = context.Export("response key", Nk)
nonce = context.Export("response nonce", Nn)
```

In this example, the length of each secret is based on the AEAD algorithm used for the corresponding HPKE context.

Note that HPKE's limitations with regard to sender authentication become limits on recipient authentication in this context. In particular, in the Base mode, there is no authentication of the remote party at all. Even in the Auth mode, where the remote party has proven that they hold a specific private key, this authentication is still subject to Key-Compromise Impersonation, as discussed in Section 9.1.1.

9.8. Metadata Protection

The authenticated modes of HPKE (PSK, Auth, AuthPSK) require that the recipient know what key material to use for the sender. This can be signaled in applications by sending the PSK ID (psk_id above) and/or the sender's public key (pkS). However, these values themselves might be considered sensitive, since in a given application context, they might identify the sender.

An application that wishes to protect these metadata values without requiring further provisioning of keys can use an additional instance of HPKE, using the unauthenticated Base mode. Where the application might have sent (psk_id, pkS, enc, ciphertext) before, it would now send (enc2, ciphertext2, enc, ciphertext), where (enc2, ciphertext2) represent the encryption of the psk_id and pkS values.

The cost of this approach is an additional KEM operation each for the sender and the recipient. A potential lower-cost approach (involving only symmetric operations) would be available if the nonce-protection schemes in [BNT19] could be extended to cover other metadata. However, this construction would require further analysis.

10. Message Encoding

This document does not specify a wire format encoding for HPKE messages. Applications that adopt HPKE must therefore specify an unambiguous encoding mechanism which includes, minimally: the encapsulated value enc, ciphertext value(s) (and order if there are multiple), and any info values that are not implicit. One example of a non-implicit value is the recipient public key used for encapsulation, which may be needed if a recipient has more than one public key.

11. IANA Considerations

This document requests the creation of three new IANA registries:

- *HPKE KEM Identifiers
- *HPKE KDF Identifiers
- *HPKE AEAD Identifiers

All these registries should be under a heading of "Hybrid Public Key Encryption", and administered under a Specification Required policy [RFC8126]

11.1. KEM Identifiers

The "HPKE KEM Identifiers" registry lists identifiers for key encapsulation algorithms defined for use with HPKE. These are two-byte values, so the maximum possible value is 0xFFFF = 65535.

Template:

- *Value: The two-byte identifier for the algorithm
- *KEM: The name of the algorithm

*Nsecret: The length in bytes of a KEM shared secret produced by the algorithm

*Nenc: The length in bytes of an encoded encapsulated key produced by the algorithm

*Npk: The length in bytes of an encoded public key for the algorithm

*Nsk: The length in bytes of an encoded private key for the algorithm

*Auth: A boolean indicating if this algorithm provides the AuthEncap()/AuthDecap() interface

*Reference: Where this algorithm is defined

Initial contents: Provided in Table 2

11.2. KDF Identifiers

The "HPKE KDF Identifiers" registry lists identifiers for key derivation functions defined for use with HPKE. These are two-byte values, so the maximum possible value is 0xFFFF = 65535.

Template:

*Value: The two-byte identifier for the algorithm

*KDF: The name of the algorithm

*Nh: The output size of the Extract function in bytes

*Reference: Where this algorithm is defined

Initial contents: Provided in Table 3

11.3. AEAD Identifiers

The "HPKE AEAD Identifiers" registry lists identifiers for authenticated encryption with associated data (AEAD) algorithms defined for use with HPKE. These are two-byte values, so the maximum possible value is 0xFFFF = 65535.

Template:

*Value: The two-byte identifier for the algorithm

*AEAD: The name of the algorithm

*Nk: The length in bytes of a key for this algorithm

*Nn: The length in bytes of a nonce for this algorithm

*Reference: Where this algorithm is defined

Initial contents: Provided in Table 5

12. Acknowledgements

The authors would like to thank Joel Alwen, Jean-Philippe Aumasson, David Benjamin, Benjamin Beurdouche, Bruno Blanchet, Frank Denis, Stephen Farrell, Scott Fluhrer, Eduard Hauck, Scott Hollenbeck, Kevin Jacobs, Burt Kaliski, Eike Kiltz, Julia Len, John Mattsson, Christopher Patton, Doreen Riepel, Raphael Robert, Michael Rosenberg, Michael Scott, Martin Thomson, Steven Valdez, Riad Wahby, and other contributors in the CFRG for helpful feedback that greatly improved this document.

13. References

13.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC5116] McGrew, D., "An Interface and Algorithms for Authenticated Encryption", RFC 5116, DOI 10.17487/RFC5116, January 2008, <https://www.rfc-editor.org/info/rfc5116>.

[RFC8017] Moriarty, K., Ed., Kaliski, B., Jonsson, J., and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2", RFC 8017, DOI 10.17487/RFC8017, November 2016, <https://www.rfc-editor.org/info/rfc8017>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

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

13.2. Informative References

[ABHKLR20] Alwen, J., Blanchet, B., Hauck, E., Kiltz, E., Lipp, B., and D. Riepel, "Analysing the HPKE Standard", 2020, <https://eprint.iacr.org/2020/1499>.

[ANSI] American National Standards Institute, "ANSI X9.63 Public Key Cryptography for the Financial Services Industry -- Key Agreement and Key Transport Using Elliptic Curve Cryptography", 2001.

[BHK09] Mihir Bellare, ., Dennis Hofheinz, ., and . Eike Kiltz, "Subtleties in the Definition of IND-CCA: When and How Should Challenge-Decryption be Disallowed?", 2009, <https://eprint.iacr.org/2009/418>.

[BJM97] Blake-Wilson, S., Johnson, D., and A. Menezes, "Key agreement protocols and their security analysis: Extended Abstract", DOI 10.1007/bfb0024447, Crytography and Coding pp. 30-45, 1997, <https://doi.org/10.1007/bfb0024447>.

[BNT19] Bellare, M., Ng, R., and B. Tackmann, "Nonces Are Noticed: AEAD Revisited", 2019, <http://dx.doi.org/10.1007/978-3-030-26948-7_9>.

[CS01] Cramer, R. and V. Shoup, "Design and Analysis of Practical Public-Key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack", 2001, <https://eprint.iacr.org/2001/108>.

[GAP] Okamoto, T. and D. Pointcheval, "The Gap-Problems - a New Class of Problems for the Security of Cryptographic Schemes", ISBN 978-3-540-44586-9, 2001, <https://link.springer.com/content/pdf/10.1007/3-540-44586-2_8.pdf>.

[GCM] Dworkin, M., "Recommendation for block cipher modes of operation :: GaloisCounter Mode (GCM) and GMAC", DOI 10.6028/nist.sp.800-38d, National Institute of Standards and Technology report, 2007, <https://doi.org/10.6028/nist.sp.800-38d>.

[HHK06] Herranz, J., Hofheinz, D., and E. Kiltz, "Some (in)sufficient conditions for secure hybrid encryption", 2006, <https://eprint.iacr.org/2006/265>.

[HPKEAnalysis] Lipp, B., "An Analysis of Hybrid Public Key Encryption", 2020, <https://eprint.iacr.org/2020/243.pdf>.

[I-D.ietf-mls-protocol] Barnes, R., Beurdouche, B., Millican, J., Omara, E., Cohn-Gordon, K., and R. Robert, "The Messaging Layer Security (MLS) Protocol", Work in Progress, Internet-Draft, draft-ietf-mls-protocol-11, 22 December

2020, <https://www.ietf.org/archive/id/draft-ietf-mls-protocol-11.txt>.

[I-D.ietf-tls-esni] Rescorla, E., Oku, K., Sullivan, N., and C. A. Wood, "TLS Encrypted Client Hello", Work in Progress, Internet-Draft, draft-ietf-tls-esni-10, 8 March 2021, <https://www.ietf.org/archive/id/draft-ietf-tls-esni-10.txt>.

[IEEE1363] Institute of Electrical and Electronics Engineers, "IEEE 1363a, Standard Specifications for Public Key Cryptography - Amendment 1 -- Additional Techniques"", 2004.

[IMB] Diffie, W., Van Oorschot, P., and M. Wiener, "Authentication and authenticated key exchanges", DOI 10.1007/bf00124891, Designs, Codes and Cryptography Vol. 2, pp. 107-125, June 1992, <https://doi.org/10.1007/bf00124891>.

[ISO] International Organization for Standardization / International Electrotechnical Commission, "ISO/IEC 18033-2, Information Technology - Security Techniques - Encryption Algorithms - Part 2 -- Asymmetric Ciphers", 2006.

[keyagreement] Barker, E., Chen, L., Roginsky, A., Vassilev, A., and R. Davis, "Recommendation for pair-wise key-establishment schemes using discrete logarithm cryptography", DOI 10.6028/nist.sp.800-56ar3, National Institute of Standards and Technology report, April 2018, <https://doi.org/10.6028/nist.sp.800-56ar3>.

[LGR20] Len, J., Grubbs, P., and T. Ristenpart, "Partitioning Oracle Attacks".

[MAEA10] Gayoso Martinez, V., Hernandez Alvarez, F., Hernandez Encinas, L., and C. Sanchez Avila, "A Comparison of the Standardized Versions of ECIES", 2010, <https://ieeexplore.ieee.org/abstract/document/5604194/>.

[NaCl] "Public-key authenticated encryption: crypto_box", 2019, <https://nacl.cr.yp.to/box.html>.

[NISTCurves] "Digital Signature Standard (DSS)", DOI 10.6028/nist.fips.186-4, National Institute of Standards and

Technology report, July 2013, <https://doi.org/10.6028/nist.fips.186-4>.

[RFC1421] Linn, J., "Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures", RFC 1421, DOI 10.17487/RFC1421, February 1993, <https://www.rfc-editor.org/info/rfc1421>.

[RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <https://www.rfc-editor.org/info/rfc5869>.

[RFC7748] Langley, A., Hamburg, M., and S. Turner, "Elliptic Curves for Security", RFC 7748, DOI 10.17487/RFC7748, January 2016, <https://www.rfc-editor.org/info/rfc7748>.

[RFC8439] Nir, Y. and A. Langley, "ChaCha20 and Poly1305 for IETF Protocols", RFC 8439, DOI 10.17487/RFC8439, June 2018, <https://www.rfc-editor.org/info/rfc8439>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8467] Mayrhofer, A., "Padding Policies for Extension Mechanisms for DNS (EDNS(0))", RFC 8467, DOI 10.17487/RFC8467, October 2018, <https://www.rfc-editor.org/info/rfc8467>.

[SECG] "Elliptic Curve Cryptography, Standards for Efficient Cryptography Group, ver. 2", 2009, <https://secg.org/sec1-v2.pdf>.

[SigncryptionDZ10] "Practical Signcryption", DOI 10.1007/978-3-540-89411-7, Information Security and Cryptography, 2010, <https://doi.org/10.1007/978-3-540-89411-7>.

[TestVectors] "HPKE Test Vectors", 2021, <https://github.com/cfrg/draft-irtf-cfrg-hpke/blob/5f503c564da00b0687b3de75f1dfbdfc4079ad31/test-vectors.json>.

[WireGuard] Donenfeld, J.A., "WireGuard: Next Generation Kernel Network Tunnel", 2020, <https://www.wireguard.com/papers/wireguard.pdf>.

Appendix A. Test Vectors

Each section below contains test vectors for a single HPKE ciphersuite and contains the following values:

1. Configuration information and private key material: This includes the mode, info string, HPKE ciphersuite identifiers (kem_id, kdf_id, aead_id), and all sender, recipient, and ephemeral key material. For each role X, where X is one of S, R, or E as sender, recipient, and ephemeral, respectively, key pairs are generated as (skX, pkX) = DeriveKeyPair(ikmX). Each key pair (skX, pkX) is written in its serialized form, where skXm = SerializePrivateKey(skX) and pkXm = SerializePublicKey(pkX). For applicable modes, the shared PSK and PSK identifier are also included.

2. Context creation intermediate values and outputs: This includes the KEM outputs enc and shared_secret used to create the context, along with intermediate values key_schedule_context and secret computed in the KeySchedule function in Section 5.1. The outputs include the context values key, base_nonce, and exporter_secret.

3. Encryption test vectors: A fixed plaintext message is encrypted using different sequence numbers and AAD values using the context computed in (2). Each test vector lists the sequence number and corresponding nonce computed with base_nonce, the plaintext message pt, AAD aad, and output ciphertext ct.

4. Export test vectors: Several exported values of the same length with differing context parameters are computed using the context computed in (2). Each test vector lists the exporter_context, output length L, and resulting export value.

These test vectors are also available in JSON format at [TestVectors].

A.1. DHKEM(X25519, HKDF-SHA256), HKDF-SHA256, AES-128-GCM

A.1.1. Base Setup Information

```
mode: 0
kem_id: 32
kdf_id: 1
aead_id: 1
info: 4f6465206f6e20612047726563696616e2055726e
ikmE: 7268600d403fce431561aef583ee1613527cff655c1343f29812e66706df3234
pkEm: 37fda3567bdbd628e88668c3c8d7e97d1d1253b6d4ea6d44c150f741f1bf4431
skEm: 52c4a758a802cd8b936eceea314432798d5baf2d7e9235dc084ab1b9cfa2f736
ikmR: 6db9df30aa07dd42ee5e8181afdb977e538f5e1fec8a06223f33f7013e525037
pkRm: 3948cfe0ad1ddb695d780e59077195da6c56506b027329794ab02bca80815c4d
skRm: 4612c550263fc8ad58375df3f557aac531d26850903e55a9f23f21d8534e8ac8
enc: 37fda3567bdbd628e88668c3c8d7e97d1d1253b6d4ea6d44c150f741f1bf4431
shared_secret:
fe0e18c9f024ce43799ae393c7e8fe8fce9d218875e8227b0187c04e7d2ea1fc
key_schedule_context: 00725611c9d98c07c03f60095cd32d400d8347d45ed67097bb
ad50fc56da742d07cb6cffde367bb0565ba28bb02c90744a20f5ef37f30523526106f637
abb05449
secret: 12fff91991e93b48de37e7daddb52981084bd8aa64289c3788471d9a9712f397
key: 4531685d41d65f03dc48f6b8302c05b0
base_nonce: 56d890e5accaaf011cff4b7d
exporter_secret:
45ff1c2e220db587171952c0592d5f5ebe103f1561a2614e38f2ffd47e99e3f8
```

A.1.1.1. Encryptions

sequence number: 0
pt: 4265617574792069732074727574682c2074727574682062656175747
9
aad: 436f756e742d30
nonce: 56d890e5accaaf011cff4b7d
ct: f938558b5d72f1a23810b4be2ab4f84331acc02fc97babc53a52ae8218a355a96d87
70ac83d07bea87e13c512a

sequence number: 1
pt: 4265617574792069732074727574682c2074727574682062656175747
9
aad: 436f756e742d31
nonce: 56d890e5accaaf011cff4b7c
ct: af2d7e9ac9ae7e270f46ba1f975be53c09f8d875bdc8535458c2494e8a6eab251c03
d0c22a56b8ca42c2063b84

sequence number: 2
pt: 4265617574792069732074727574682c2074727574682062656175747
9
aad: 436f756e742d32
nonce: 56d890e5accaaf011cff4b7f
ct: 498dfcabd92e8acedc281e85af1cb4e3e31c7dc394a1ca20e173cb72516491588d96
a19ad4a683518973dcc180

sequence number: 4
pt: 4265617574792069732074727574682c2074727574682062656175747
9
aad: 436f756e742d34
nonce: 56d890e5accaaf011cff4b79
ct: 583bd32bc67a5994bb8ceaca813d369bca7b2a42408cddef5e22f880b631215a09fc
0012bc69fccaa251c0246d

sequence number: 255
pt: 4265617574792069732074727574682c2074727574682062656175747
9
aad: 436f756e742d323535
nonce: 56d890e5accaaf011cff4b82
ct: 7175db9717964058640a3a11fb9007941a5d1757fda1a6935c805c21af32505bf106
deefec4a49ac38d71c9e0a

sequence number: 256
pt: 4265617574792069732074727574682c2074727574682062656175747
9
aad: 436f756e742d323536
nonce: 56d890e5accaaf011cff4a7d
ct: 957f9800542b0b8891badb026d79cc54597cb2d225b54c00c5238c25d05c30e3fbed
a97d2e0e1aba483a2df9f2

A.1.1.2. Exported Values

```
exporter_context:
L: 32
exported_value:
3853fe2b4035195a573ffc53856e77058e15d9ea064de3e59f4961d0095250ee

exporter_context: 00
L: 32
exported_value:
2e8f0b54673c7029649d4eb9d5e33bf1872cf76d623ff164ac185da9e88c21a5

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
e9e43065102c3836401bed8c3c3c75ae46be1639869391d62c61f1ec7af54931
```

A.1.2. PSK Setup Information

```
mode: 1
kem_id: 32
kdf_id: 1
aead_id: 1
info: 4f6465206f6e2061204772656369616e2055726e
ikmE: 78628c354e46f3e169bd231be7b2ff1c77aa302460a26dbfa15515684c00130b
pkEm: 0ad0950d9fb9588e59690b74f1237ecdf1d775cd60be2eca57af5a4b0471c91b
skEm: 463426a9ffb42bb17dbe6044b9abd1d4e4d95f9041cef0e99d7824eef2b6f588
ikmR: d4a09d09f575fef425905d2ab396c1449141463f698f8efdb7accfaff8995098
pkRm: 9fed7e8c17387560e92cc6462a68049657246a09bfa8ade7aefe589672016366
skRm: c5eb01eb457fe6c6f57577c5413b931550a162c71a03ac8d196babbd4e5ce0fd
psk: 0247fd33b913760fa1fa51e1892d9f307fbe65eb171e8132c2af18555a738b82
psk_id: 456e6e796e20447572696e206172616e204d6f726961
enc: 0ad0950d9fb9588e59690b74f1237ecdf1d775cd60be2eca57af5a4b0471c91b
shared_secret:
727699f009ffe3c076315019c69648366b69171439bd7dd0807743bde76986cd
key_schedule_context: 01e78d5cf6190d275863411ff5edd0dece5d39fa48e04eec1e
d9b71be34729d18ccb6cffde367bb0565ba28bb02c90744a20f5ef37f30523526106f637
abb05449
secret: 3728ab0b024b383b0381e432b47cced1496d2516957a76e2a9f5c8cb947afca4
key: 15026dba546e3ae05836fc7de5a7bb26
base_nonce: 9518635eba129d5ce0914555
exporter_secret:
3d76025dbbedc49448ec3f9080a1abab6b06e91c0b11ad23c912f043a0ee7655
```

A.1.2.1. Encryptions

sequence number: 0
pt: 4265617574792069732074727574682c20747275746820626561757479
aad: 436f756e742d30
nonce: 9518635eba129d5ce0914555
ct: e52c6fed7f758d0cf7145689f21bc1be6ec9ea097fef4e959440012f4feb73fb611b
946199e681f4cfc34db8ea

sequence number: 1
pt: 4265617574792069732074727574682c20747275746820626561757479
aad: 436f756e742d31
nonce: 9518635eba129d5ce0914554
ct: 49f3b19b28a9ea9f43e8c71204c00d4a490ee7f61387b6719db765e948123b45b616
33ef059ba22cd62437c8ba

sequence number: 2
pt: 4265617574792069732074727574682c20747275746820626561757479
aad: 436f756e742d32
nonce: 9518635eba129d5ce0914557
ct: 257ca6a08473dc851fde45afd598cc83e326ddd0abe1ef23baa3baa4dd8cde99fce2
c1e8ce687b0b47ead1adc9

sequence number: 4
pt: 4265617574792069732074727574682c20747275746820626561757479
aad: 436f756e742d34
nonce: 9518635eba129d5ce0914551
ct: a71d73a2cd8128fcccbd328b9684d70096e073b59b40b55e6419c9c68ae21069c847
e2a70f5d8fb821ce3dfb1c

sequence number: 255
pt: 4265617574792069732074727574682c20747275746820626561757479
aad: 436f756e742d323535
nonce: 9518635eba129d5ce09145aa
ct: 55f84b030b7f7197f7d7d552365b6b932df5ec1abacd30241cb4bc4ccea27bd2b518
766adfa0fb1b71170e9392

sequence number: 256
pt: 4265617574792069732074727574682c20747275746820626561757479
aad: 436f756e742d323536
nonce: 9518635eba129d5ce0914455
ct: c5bf246d4a790a12dcc9eed5eae525081e6fb541d5849e9ce8abd92a3bc1551776be
a16b4a518f23e237c14b59

A.1.2.2. Exported Values

exporter_context:
L: 32
exported_value:
dff17af354c8b41673567db6259fd6029967b4e1aad13023c2ae5df8f4f43bf6

exporter_context: 00
L: 32
exported_value:
6a847261d8207fe596befb52928463881ab493da345b10e1dcc645e3b94e2d95

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
8aff52b45a1be3a734bc7a41e20b4e055ad4c4d22104b0c20285a7c4302401cd

A.1.3. Auth Setup Information

mode: 2
kem_id: 32
kdf_id: 1
aead_id: 1
info: 4f6465206f6e2061204772656369616e2055726e
ikmE: 6e6d8f200ea2fb20c30b003a8b4f433d2f4ed4c2658d5bc8ce2fef718059c9f7
pkEm: 23fb952571a14a25e3d678140cd0e5eb47a0961bb18afcf85896e5453c312e76
skEm: ff4442ef24fbc3c1ff86375b0be1e77e88a0de1e79b30896d73411c5ff4c3518
ikmR: f1d4a30a4cef8d6d4e3b016e6fd3799ea057db4f345472ed302a67ce1c20cdec
pkRm: 1632d5c2f71c2b38d0a8fcc359355200caa8b1ffdf28618080466c909cb69b2e
skRm: fdea67cf831f1ca98d8e27b1f6abeb5b7745e9d35348b80fa407ff6958f9137e
ikmS: 94b020ce91d73fca4649006c7e7329a67b40c55e9e93cc907d282bbbff386f58
pkSm: 8b0c70873dc5aecb7f9ee4e62406a397b350e57012be45cf53b7105ae731790b
skSm: dc4a146313cce60a278a5323d321f051c5707e9c45ba21a3479fecdf76fc69dd
enc: 23fb952571a14a25e3d678140cd0e5eb47a0961bb18afcf85896e5453c312e76
shared_secret:
2d6db4cf719dc7293fcbf3fa64690708e44e2bebc81f84608677958c0d4448a7
key_schedule_context: 02725611c9d98c07c03f60095cd32d400d8347d45ed67097bb
ad50fc56da742d07cb6cffde367bb0565ba28bb02c90744a20f5ef37f30523526106f637
abb05449
secret: 56c62333d9d9f7767f5b083fdfce0aa7e57e301b74029bb0cffa7331385f1dda
key: b062cb2c4dd4bca0ad7c7a12bbc341e6
base_nonce: a1bc314c1942ade7051ffed0
exporter_secret:
ee1a093e6e1c393c162ea98fdf20560c75909653550540a2700511b65c88c6f1

A.1.3.1. Encryptions

```
sequence number: 0
pt: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d30
nonce: a1bc314c1942ade7051ffed0
ct: 5fd92cc9d46dbf8943e72a07e42f363ed5f721212cd90bcfd072bfd9f44e06b80fd1
7824947496e21b680c141b

sequence number: 1
pt: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d31
nonce: a1bc314c1942ade7051ffed1
ct: d3736bb256c19bfa93d79e8f80b7971262cb7c887e35c26370cfed62254369a1b52e
3d505b79dd699f002bc8ed

sequence number: 2
pt: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d32
nonce: a1bc314c1942ade7051ffed2
ct: 122175cfd5678e04894e4ff8789e85dd381df48dcaf970d52057df2c9acc3b121313
a2bfeaa986050f82d93645

sequence number: 4
pt: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d34
nonce: a1bc314c1942ade7051ffed4
ct: dae12318660cf963c7bcbef0f39d64de3bf178cf9e585e756654043cc5059873bc8a
f190b72afc43d1e0135ada

sequence number: 255
pt: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d323535
nonce: a1bc314c1942ade7051ffe2f
ct: 55d53d85fe4d9e1e97903101eab0b4865ef20cef28765a47f840ff99625b7d69dee9
27df1defa66a036fc58ff2

sequence number: 256
pt: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d323536
nonce: a1bc314c1942ade7051fffd0
ct: 42fa248a0e67ccca688f2b1d13ba4ba84755acf764bd797c8f7ba3b9b1dc3330326f
8d172fef6003c79ec72319
```

A.1.3.2. Exported Values

```
exporter_context:
L: 32
exported_value:
28c70088017d70c896a8420f04702c5a321d9cbf0279fba899b59e51bac72c85

exporter_context: 00
L: 32
exported_value:
25dfc004b0892be1888c3914977aa9c9bbaf2c7471708a49e1195af48a6f29ce

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
5a0131813abc9a522cad678eb6bafaabc43389934adb8097d23c5ff68059eb64
```

A.1.4. AuthPSK Setup Information

```
mode: 3
kem_id: 32
kdf_id: 1
aead_id: 1
info: 4f6465206f6e20612047726563696616e2055726e
ikmE: 4303619085a20ebcf18edd22782952b8a7161e1dbae6e46e143a52a96127cf84
pkEm: 820818d3c23993492cc5623ab437a48a0a7ca3e9639c140fe1e33811eb844b7c
skEm: 14de82a5897b613616a00c39b87429df35bc2b426bcfd73febcb45e903490768
ikmR: 4b16221f3b269a88e207270b5e1de28cb01f847841b344b8314d6a622fe5ee90
pkRm: 1d11a3cd247ae48e901939659bd4d79b6b959e1f3e7d66663fbc9412dd4e0976
skRm: cb29a95649dc5656c2d054c1aa0d3df0493155e9d5da6d7e344ed8b6a64a9423
ikmS: 62f77dcf5df0dd7eac54eac9f654f426d4161ec850cc65c54f8b65d2e0b4e345
pkSm: 2bfb2eb18fcad1af0e4f99142a1c474ae74e21b9425fc5c589382c69b50cc57e
skSm: fc1c87d2f3832adb178b431fce2ac77c7ca2fd680f3406c77b5ecdf818b119f4
psk: 0247fd33b913760fa1fa51e1892d9f307fbe65eb171e8132c2af18555a738b82
psk_id: 456e6e796e20447572696e206172616e204d6f726961
enc: 820818d3c23993492cc5623ab437a48a0a7ca3e9639c140fe1e33811eb844b7c
shared_secret:
f9d0e870aba28d04709b2680cb8185466c6a6ff1d6e9d1091d5bf5e10ce3a577
key_schedule_context: 03e78d5cf6190d275863411ff5edd0dece5d39fa48e04eec1e
d9b71be34729d18ccb6cffde367bb0565ba28bb02c90744a20f5ef37f30523526106f637
abb05449
secret: 5f96c55e4108c6691829aaabaa7d539c0b41d7c72aae94ae289752f056b6cec4
key: 1364ead92c47aa7becfa95203037b19a
base_nonce: 99d8b5c54669807e9fc70df1
exporter_secret:
f048d55eacbf60f9c6154bd4021774d1075ebf963c6adc71fa846f183ab2dde6
```

A.1.4.1. Encryptions

sequence number: 0
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d30
nonce: 99d8b5c54669807e9fc70df1
ct: a84c64df1e11d8fd11450039d4fe64ff0c8a99fca0bd72c2d4c3e0400bc14a40f27e
45e141a24001697737533e

sequence number: 1
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d31
nonce: 99d8b5c54669807e9fc70df0
ct: 4d19303b848f424fc3c3beca249b2c6de0a34083b8e909b6aa4c3688505c05ffe0c8
f57a0a4c5ab9da127435d9

sequence number: 2
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d32
nonce: 99d8b5c54669807e9fc70df3
ct: 0c085a365fbfa63409943b00a3127abce6e45991bc653f182a80120868fc507e9e4d
5e37bcc384fc8f14153b24

sequence number: 4
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d34
nonce: 99d8b5c54669807e9fc70df5
ct: 000a3cd3a3523bf7d9796830b1cd987e841a8bae6561ebb6791a3f0e34e89a4fb539
faeee3428b8bbc082d2c1a

sequence number: 255
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323535
nonce: 99d8b5c54669807e9fc70d0e
ct: 576d39dd2d4cc77d1a14a51d5c5f9d5e77586c3d8d2ab33bdec6379e28ce5c502f0b
1cbd09047cf9eb9269bb52

sequence number: 256
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323536
nonce: 99d8b5c54669807e9fc70cf1
ct: 13239bab72e25e9fd5bb09695d23c90a24595158b99127505c8a9ff9f127e0d657f7
1af59d67d4f4971da028f9

A.1.4.2. Exported Values

```
exporter_context:
L: 32
exported_value:
08f7e20644bb9b8af54ad66d2067457c5f9fcb2a23d9f6cb4445c0797b330067

exporter_context: 00
L: 32
exported_value:
52e51ff7d436557ced5265ff8b94ce69cf7583f49cdb374e6aad801fc063b010

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
a30c20370c026bbea4dca51cb63761695132d342bae33a6a11527d3e7679436d
```

A.2. DHKEM(X25519, HKDF-SHA256), HKDF-SHA256, ChaCha20Poly1305

A.2.1. Base Setup Information

```
mode: 0
kem_id: 32
kdf_id: 1
aead_id: 3
info: 4f6465206f6e20612047726563696616e2055726e
ikmE: 909a9b35d3dc4713a5e72a4da274b55d3d3821a37e5d099e74a647db583a904b
pkEm: 1afa08d3dec047a643885163f1180476fa7ddb54c6a8029ea33f95796bf2ac4a
skEm: f4ec9b33b792c372c1d2c2063507b684ef925b8c75a42dbcbf57d63ccd381600
ikmR: 1ac01f181fdf9f352797655161c58b75c656a6cc2716dcb66372da835542e1df
pkRm: 4310ee97d88cc1f088a5576c77ab0cf5c3ac797f3d95139c6c84b5429c59662a
skRm: 8057991eef8f1f1af18f4a9491d16a1ce333f695d4db8e38da75975c4478e0fb
enc: 1afa08d3dec047a643885163f1180476fa7ddb54c6a8029ea33f95796bf2ac4a
shared_secret:
0bbe78490412b4bbea4812666f7916932b828bba79942424abb65244930d69a7
key_schedule_context: 00431df6cd95e11ff49d7013563baf7f11588c75a6611ee2a4
404a49306ae4cfc5b69c5718a60cc5876c358d3f7fc31ddb598503f67be58ea1e798c0bb
19eb9796
secret: 5b9cd775e64b437a2335cf499361b2e0d5e444d5cb41a8a53336d8fe402282c6
key: ad2744de8e17f4ebba575b3f5f5a8fa1f69c2a07f6e7500bc60ca6e3e3ec1c91
base_nonce: 5c4d98150661b848853b547f
exporter_secret:
a3b010d4994890e2c6968a36f64470d3c824c8f5029942feb11e7a74b2921922
```

A.2.1.1. Encryptions

```
sequence number: 0
pt: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d30
nonce: 5c4d98150661b848853b547f
ct: 1c5250d8034ec2b784ba2cfd69dbdb8af406cfe3ff938e131f0def8c8b60b4db2199
3c62ce81883d2dd1b51a28

sequence number: 1
pt: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d31
nonce: 5c4d98150661b848853b547e
ct: 6b53c051e4199c518de79594e1c4ab18b96f081549d45ce015be002090bb119e8528
5337cc95ba5f59992dc98c

sequence number: 2
pt: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d32
nonce: 5c4d98150661b848853b547d
ct: 71146bd6795ccc9c49ce25dda112a48f202ad220559502cef1f34271e0cb4b02b4f1
0ecac6f48c32f878fae86b

sequence number: 4
pt: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d34
nonce: 5c4d98150661b848853b547b
ct: 63357a2aa291f5a4e5f27db6baa2af8cf77427c7c1a909e0b37214dd47db122bb153
495ff0b02e9e54a50dbe16

sequence number: 255
pt: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d323535
nonce: 5c4d98150661b848853b5480
ct: 18ab939d63ddec9f6ac2b60d61d36a7375d2070c9b6838611107570 62c52b8880a5f
6b3936da9cd6c23ef2a95c

sequence number: 256
pt: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d323536
nonce: 5c4d98150661b848853b557f
ct: 7a4a13e9ef23978e2c520fd4d2e757514ae160cd0cd05e556ef692370ca53076214c
0c40d4c728d6ed9e727a5b
```

A.2.1.2. Exported Values

exporter_context:
L: 32
exported_value:
4bbd6243b8bb54cec311fac9df81841b6fd61f56538a775e7c80a9f40160606e

exporter_context: 00
L: 32
exported_value:
8c1df14732580e5501b00f82b10a1647b40713191b7c1240ac80e2b68808ba69

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
5acb09211139c43b3090489a9da433e8a30ee7188ba8b0a9a1ccf0c229283e53

A.2.2. PSK Setup Information

mode: 1
kem_id: 32
kdf_id: 1
aead_id: 3
info: 4f6465206f6e2061204772656369616e2055726e
ikmE: 35706a0b09fb26fb45c39c2f5079c709c7cf98e43afa973f14d88ece7e29c2e3
pkEm: 2261299c3f40a9afc133b969a97f05e95be2c514e54f3de26cbe5644ac735b04
skEm: 0c35fdf49df7aa01cd330049332c40411ebba36e0c718ebc3edf5845795f6321
ikmR: 26b923eade72941c8a85b09986cdfa3f1296852261adedc52d58d2930269812b
pkRm: 13640af826b722fc04feaa4de2f28fbd5ecc03623b317834e7ff4120dbe73062
skRm: 77d114e0212be51cb1d76fa99dd41cfd4d0166b08caa09074430a6c59ef17879
psk: 0247fd33b913760fa1fa51e1892d9f307fbe65eb171e8132c2af18555a738b82
psk_id: 456e6e796e20447572696e206172616e204d6f726961
enc: 2261299c3f40a9afc133b969a97f05e95be2c514e54f3de26cbe5644ac735b04
shared_secret:
4be079c5e77779d0215b3f689595d59e3e9b0455d55662d1f3666ec606e50ea7
key_schedule_context: 016870c4c76ca38ae43efbec0f2377d109499d7ce73f4a9e1e
c37f21d3d063b97cb69c5718a60cc5876c358d3f7fc31ddb598503f67be58ea1e798c0bb
19eb9796
secret: 16974354c497c9bd24c000ceed693779b604f1944975b18c442d373663f4a8cc
key: 600d2fdb0313a7e5c86a9ce9221cd95bed069862421744cfb4ab9d7203a9c019
base_nonce: 112e0465562045b7368653e7
exporter_secret:
73b506dc8b6b4269027f80b0362def5cbb57ee50eed0c2873dac9181f453c5ac

A.2.2.1. Encryptions

```
sequence number: 0
pt: 4265617574792069732074727574682c207472757468206265617574792
aad: 436f756e742d30
nonce: 112e0465562045b7368653e7
ct: 4a177f9c0d6f15cfdf533fb65bf84aecdc6ab16b8b85b4cf65a370e07fc1d78d28fb
073214525276f4a89608ff

sequence number: 1
pt: 4265617574792069732074727574682c207472757468206265617574792
aad: 436f756e742d31
nonce: 112e0465562045b7368653e6
ct: 5c3cabae2f0b3e124d8d864c116fd8f20f3f56fda988c3573b40b09997fd6c769e77
c8eda6cda4f947f5b704a8

sequence number: 2
pt: 4265617574792069732074727574682c207472757468206265617574792
aad: 436f756e742d32
nonce: 112e0465562045b7368653e5
ct: 14958900b44bdae9cbe5a528bf933c5c990dbb8e282e6e495adf8205d19da9eb270e
3a6f1e0613ab7e757962a4

sequence number: 4
pt: 4265617574792069732074727574682c207472757468206265617574792
aad: 436f756e742d34
nonce: 112e0465562045b7368653e3
ct: c2a7bc09ddb853cf2effb6e8d058e346f7fe0fb3476528c80db6b698415c5f8c50b6
8a9a355609e96d2117f8d3

sequence number: 255
pt: 4265617574792069732074727574682c207472757468206265617574792
aad: 436f756e742d323535
nonce: 112e0465562045b736865318
ct: 2414d0788e4bc39a59a26d7bd5d78e111c317d44c37bd5a4c2a1235f2ddc2085c487
d406490e75210c958724a7

sequence number: 256
pt: 4265617574792069732074727574682c207472757468206265617574792
aad: 436f756e742d323536
nonce: 112e0465562045b7368652e7
ct: c567ae1c3f0f75abe1dd9e4532b422600ed4a6e5b9484dafb1e43ab9f5fd662b28c0
0e2e81d3cde955dae7e218
```

A.2.2.2. Exported Values

```
exporter_context:
L: 32
exported_value:
813c1bfc516c99076ae0f466671f0ba5ff244a41699f7b2417e4c59d46d39f40

exporter_context: 00
L: 32
exported_value:
2745cf3d5bb65c333658732954ee7af49eb895ce77f8022873a62a13c94cb4e1

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
ad40e3ae14f21c99bfdebc20ae14ab86f4ca2dc9a4799d200f43a25f99fa78ae
```

A.2.3. Auth Setup Information

```
mode: 2
kem_id: 32
kdf_id: 1
aead_id: 3
info: 4f6465206f6e2061204772656369616e2055726e
ikmE: 938d3daa5a8904540bc24f48ae90eed3f4f7f11839560597b55e7c9598c996c0
pkEm: f7674cc8cd7baa5872d1f33dbaffe3314239f6197ddf5ded1746760bfc847e0e
skEm: c94619e1af28971c8fa7957192b7e62a71ca2dcdde0a7cc4a8a9e741d600ab13
ikmR: 64835d5ee64aa7aad57c6f2e4f758f7696617f8829e70bc9ac7a5ef95d1c756c
pkRm: 1a478716d63cb2e16786ee93004486dc151e988b34b475043d3e0175bdb01c44
skRm: 3ca22a6d1cda1bb9480949ec5329d3bf0b080ca4c45879c95eddb55c70b80b82
ikmS: 9d8f94537d5a3ddef71234c0baedfad4ca6861634d0b94c3007fed557ad17df6
pkSm: f0f4f9e96c54aeed3f323de8534fffd7e0577e4ce269896716bcb95643c8712b
skSm: 2def0cb58ffcf83d1062dd085c8aceca7f4c0c3fd05912d847b61f3e54121f05
enc: f7674cc8cd7baa5872d1f33dbaffe3314239f6197ddf5ded1746760bfc847e0e
shared_secret:
d2d67828c8bc9fa661cf15a31b3ebf1febe0cafef7abfaaca580aaf6d471e3eb
key_schedule_context: 02431df6cd95e11ff49d7013563baf7f11588c75a6611ee2a4
404a49306ae4cfc5b69c5718a60cc5876c358d3f7fc31ddb598503f67be58ea1e798c0bb
19eb9796
secret: 3022dfc0a81d6e09a2e6daeeb605bb1ebb9ac49535540d9a4c6560064a6c6da8
key: b071fd1136680600eb447a845a967d35e9db20749cdf9ce098bcc4deef4b1356
base_nonce: d20577dff16d7cea2c4bf780
exporter_secret:
be2d93b82071318cdb88510037cf504344151f2f9b9da8ab48974d40a2251dd7
```

A.2.3.1. Encryptions

sequence number: 0
pt: 4265617574792069732074727574682c2074727574682062656175747
9
aad: 436f756e742d30
nonce: d20577dff16d7cea2c4bf780
ct: ab1a13c9d4f01a87ec3440dbd756e2677bd2ecf9df0ce7ed73869b98e00c09be111c
b9fdf077347aeb88e61bdf

sequence number: 1
pt: 4265617574792069732074727574682c2074727574682062656175747
9
aad: 436f756e742d31
nonce: d20577dff16d7cea2c4bf781
ct: 3265c7807ffff7fdace21659a2c6ccffee52a26d270c76468ed74202a65478bfaedf
ff9c2b7634e24f10b71016

sequence number: 2
pt: 4265617574792069732074727574682c2074727574682062656175747
9
aad: 436f756e742d32
nonce: d20577dff16d7cea2c4bf782
ct: 3aadee86ad2a05081ea860033a9d09dbccb4acac2ded0891da40f51d4df19925f7a7
67b076a5cbc9355c8fd35e

sequence number: 4
pt: 4265617574792069732074727574682c2074727574682062656175747
9
aad: 436f756e742d34
nonce: d20577dff16d7cea2c4bf784
ct: 502ecccd5c2be3506a081809cc58b43b94f77cbe37b8b31712d9e21c9e61aa6946a8
e922f54eae630f88eb8033

sequence number: 255
pt: 4265617574792069732074727574682c2074727574682062656175747
9
aad: 436f756e742d323535
nonce: d20577dff16d7cea2c4bf77f
ct: 652e597ba20f3d9241cda61f33937298b1169e6adf72974bbe454297502eb4be132e
1c5064702fc165c2ddbde8

sequence number: 256
pt: 4265617574792069732074727574682c2074727574682062656175747
9
aad: 436f756e742d323536
nonce: d20577dff16d7cea2c4bf680
ct: 3be14e8b3bbd1028cf2b7d0a691dbbeff71321e7dec92d3c2cfb30a0994ab246af76
168480285a60037b4ba13a

A.2.3.2. Exported Values

```
exporter_context:
L: 32
exported_value:
070cffafd89b67b7f0eeb800235303a223e6ff9d1e774dce8eac585c8688c872

exporter_context: 00
L: 32
exported_value:
2852e728568d40ddb0edde284d36a4359c56558bb2fb8837cd3d92e46a3a14a8

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
1df39dc5dd60edcbf5f9ae804e15ada66e885b28ed7929116f768369a3f950ee
```

A.2.4. AuthPSK Setup Information

```
mode: 3
kem_id: 32
kdf_id: 1
aead_id: 3
info: 4f6465206f6e2061204772656369616e2055726e
ikmE: 49d6eac8c6c558c953a0a252929a818745bb08cd3d29e15f9f5db5eb2e7d4b84
pkEm: 656a2e00dc9990fd189e6e473459392df556e9a2758754a09db3f51179a3fc02
skEm: 5e6dd73e82b856339572b7245d3cbb073a7561c0bee52873490e305cbb710410
ikmR: f3304ddcf15848488271f12b75ecaf72301faabf6ad283654a14c398832eb184
pkRm: a5099431c35c491ec62ca91df1525d6349cb8aa170c51f9581f8627be6334851
skRm: 7b36a42822e75bf3362dfabbe474b3016236408becb83b859a6909e22803cb0c
ikmS: 20ade1d5203de1aadfb261c4700b6432e260d0d317be6ebbb8d7fffb1f86ad9d
pkSm: 3ac5bd4dd66ff9f2740bef0d6ccb66daa77bff7849d7895182b07fb74d087c45
skSm: 90761c5b0a7ef0985ed66687ad708b921d9803d51637c8d1cb72d03ed0f64418
psk: 0247fd33b913760fa1fa51e1892d9f307fbe65eb171e8132c2af18555a738b82
psk_id: 456e6e796e20447572696e206172616e204d6f726961
enc: 656a2e00dc9990fd189e6e473459392df556e9a2758754a09db3f51179a3fc02
shared_secret:
86a6c0ed17714f11d2951747e660857a5fd7616c933ef03207808b7a7123fe67
key_schedule_context: 036870c4c76ca38ae43efbec0f2377d109499d7ce73f4a9e1e
c37f21d3d063b97cb69c5718a60cc5876c358d3f7fc31ddb598503f67be58ea1e798c0bb
19eb9796
secret: 22670daee17530c9564001d0a7e740e80d0bcc7ae15349f472fcc9e057cbc259
key: 49c7e6d7d2d257aded2a746fe6a9bf12d4de8007c4862b1fdffe8c35fb65054c
base_nonce: abac79931e8c1bcb8a23960a
exporter_secret:
7c6cc1bb98993cd93e2599322247a58fd41fdecd3db895fb4c5fd8d6bbe606b5
```

A.2.4.1. Encryptions

```
sequence number: 0
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d30
nonce: abac79931e8c1bcb8a23960a
ct: 9aa52e29274fc6172e38a4461361d2342585d3aeec67fb3b721ecd63f059577c7fe8
86be0ede01456ebc67d597

sequence number: 1
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d31
nonce: abac79931e8c1bcb8a23960b
ct: 59460bacdbe7a920ef2806a74937d5a691d6d5062d7daafcad7db7e4d8c649adffe5
75c1889c5c2e3a49af8e3e

sequence number: 2
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d32
nonce: abac79931e8c1bcb8a239608
ct: 5688ff6a03ba26ae936044a5c800f286fb5d1eccdd2a0f268f6ff9773b51169318d1
a1466bb36263415071db00

sequence number: 4
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d34
nonce: abac79931e8c1bcb8a23960e
ct: d936b7a01f5c7dc4c3dc04e322cc694684ee18dd71719196874e5235aed3cfb06cad
cd3bc7da0877488d7c551d

sequence number: 255
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323535
nonce: abac79931e8c1bcb8a2396f5
ct: 4d4c462f7b9b637eaf1f4e15e325b7bc629c0af6e3073422c86064cc3c98cff87300
f054fd56dd57dc34358beb

sequence number: 256
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323536
nonce: abac79931e8c1bcb8a23970a
ct: 9b7f84224922d2a9edd7b2c2057f3bcf3a547f17570575e626202e593bfdd99e9878
a1af9e41ded58c7fb77d2f
```

A.2.4.2. Exported Values

```
exporter_context:
L: 32
exported_value:
c23ebd4e7a0ad06a5dddf779f65004ce9481069ce0f0e6dd51a04539ddcbd5cd

exporter_context: 00
L: 32
exported_value:
ed7ff5ca40a3d84561067ebc8e01702bc36cf1eb99d42a92004642b9dfaadd37

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
d3bae066aa8da27d527d85c040f7dd6ccb60221c902ee36a82f70bcd62a60ee4
```

A.3. DHKEM(P-256, HKDF-SHA256), HKDF-SHA256, AES-128-GCM

A.3.1. Base Setup Information

```
mode: 0
kem_id: 16
kdf_id: 1
aead_id: 1
info: 4f6465206f6e20612047726563696616e2055726e
ikmE: 4270e54ffd08d79d5928020af4686d8f6b7d35dbe470265f1f5aa22816ce860e
pkEm: 04a92719c6195d5085104f469a8b9814d5838ff72b60501e2c4466e5e67b325ac9
8536d7b61a1af4b78e5b7f951c0900be863c403ce65c9bfcb9382657222d18c4
skEm: 4995788ef4b9d6132b249ce59a77281493eb39af373d236a1fe415cb0c2d7beb
ikmR: 668b37171f1072f3cf12ea8a236a45df23fc13b82af3609ad1e354f6ef817550
pkRm: 04fe8c19ce0905191ebc298a9245792531f26f0cece2460639e8bc39cb7f706a82
6a779b4cf969b8a0e539c7f62fb3d30ad6aa8f80e30f1d128aafd68a2ce72ea0
skRm: f3ce7fdae57e1a310d87f1ebbde6f328be0a99cdbcadf4d6589cf29de4b8ffd2
enc: 04a92719c6195d5085104f469a8b9814d5838ff72b60501e2c4466e5e67b325ac98
536d7b61a1af4b78e5b7f951c0900be863c403ce65c9bfcb9382657222d18c4
shared_secret:
c0d26aeab536609a572b07695d933b589dcf363ff9d93c93adea537aeabb8cb8
key_schedule_context: 00b88d4e6d91759e65e87c470e8b9141113e9ad5f0c8ceefc1
e088c82e6980500798e486f9c9c09c9b5c753ac72d6005de254c607d1b534ed11d493ae1
c1d9ac85
secret: 2eb7b6bf138f6b5aff857414a058a3f1750054a9ba1f72c2cf0684a6f20b10e1
key: 868c066ef58aae6dc589b6cfdd18f97e
base_nonce: 4e0bc5018beba4bf004cca59
exporter_secret:
14ad94af484a7ad3ef40e9f3be99ecc6fa9036df9d4920548424df127ee0d99f
```

A.3.1.1. Encryptions

```
sequence number: 0
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d30
nonce: 4e0bc5018beba4bf004cca59
ct: 5ad590bb8baa577f8619db35a36311226a896e7342a6d836d8b7bcd2f20b6c7f9076
ac232e3ab2523f39513434

sequence number: 1
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d31
nonce: 4e0bc5018beba4bf004cca58
ct: fa6f037b47fc21826b610172ca9637e82d6e5801eb31cbd3748271affd4ecb06646e
0329cbdf3c3cd655b28e82

sequence number: 2
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d32
nonce: 4e0bc5018beba4bf004cca5b
ct: 895cabfac50ce6c6eb02ffe6c048bf53b7f7be9a91fc559402cbc5b8dcaeb52b2ccc
93e466c28fb55fed7a7fec

sequence number: 4
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d34
nonce: 4e0bc5018beba4bf004cca5d
ct: 8787491ee8df99bc99a246c4b3216d3d57ab5076e18fa27133f520703bc70ec999dd
36ce042e44f0c3169a6a8f

sequence number: 255
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323535
nonce: 4e0bc5018beba4bf004ccaa6
ct: 2ad71c85bf3f45c6eca301426289854b31448bcf8a8ccb1deef3ebd87f60848aa53c
538c30a4dac71d619ee2cd

sequence number: 256
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323536
nonce: 4e0bc5018beba4bf004ccb59
ct: 10f179686aa2caec1758c8e554513f16472bd0a11e2a907dde0b212cbe87d74f367f
8ffe5e41cd3e9962a6afb2
```

A.3.1.2. Exported Values

```
exporter_context:
L: 32
exported_value:
5e9bc3d236e1911d95e65b576a8a86d478fb827e8bdfe77b741b289890490d4d

exporter_context: 00
L: 32
exported_value:
6cff87658931bda83dc857e6353efe4987a201b849658d9b047aab4cf216e796

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
d8f1ea7942adbba7412c6d431c62d01371ea476b823eb697e1f6e6cae1dab85a
```

A.3.2. PSK Setup Information

```
mode: 1
kem_id: 16
kdf_id: 1
aead_id: 1
info: 4f6465206f6e20612047726563696616e2055726e
ikmE: 2afa611d8b1a7b321c761b483b6a053579afa4f767450d3ad0f84a39fda587a6
pkEm: 04305d35563527bce037773d79a13deabed0e8e7cde61eecee403496959e89e4d0
ca701726696d1485137ccb5341b3c1c7aaee90a4a02449725e744b1193b53b5f
skEm: 57427244f6cc016cddf1c19c8973b4060aa13579b4c067fd5d93a5d74e32a90f
ikmR: d42ef874c1913d9568c9405407c805baddaffd0898a00f1e84e154fa787b2429
pkRm: 040d97419ae99f13007a93996648b2674e5260a8ebd2b822e84899cd52d87446ea
394ca76223b76639eccdf00e1967db10ade37db4e7db476261fcc8df97c5ffd1
skRm: 438d8bcef33b89e0e9ae5eb0957c353c25a94584b0dd59c991372a75b43cb661
psk: 0247fd33b913760fa1fa51e1892d9f307fbe65eb171e8132c2af18555a738b82
psk_id: 456e6e796e20447572696e206172616e204d6f726961
enc: 04305d35563527bce037773d79a13deabed0e8e7cde61eecee403496959e89e4d0c
a701726696d1485137ccb5341b3c1c7aaee90a4a02449725e744b1193b53b5f
shared_secret:
2e783ad86a1beae03b5749e0f3f5e9bb19cb7eb382f2fb2dd64c99f15ae0661b
key_schedule_context: 01b873cdf2dff4c1434988053b7a775e980dd2039ea24f950b
26b056ccedcb933198e486f9c9c09c9b5c753ac72d6005de254c607d1b534ed11d493ae1
c1d9ac85
secret: f2f534e55931c62eeb2188c1f53450354a725183937e68c85e68d6b267504d26
key: 55d9eb9d26911d4c514a990fa8d57048
base_nonce: b595dc6b2d7e2ed23af529b1
exporter_secret:
895a723a1eab809804973a53c0ee18ece29b25a7555a4808277ad2651d66d705
```

A.3.2.1. Encryptions

sequence number: 0
pt: 426561757479206973207472757468 2c2074727574682062656175747920
aad: 436f756e742d30
nonce: b595dc6b2d7e2ed23af529b1
ct: 90c4deb5b75318530194e4bb62f890b019b1397bbf9d0d6eb918890e1fb2be1ac260
3193b60a49c2126b75d0eb

sequence number: 1
pt: 426561757479206973207472757468 2c2074727574682062656175747920
aad: 436f756e742d31
nonce: b595dc6b2d7e2ed23af529b0
ct: 9e223384a3620f4a75b5a52f546b7262d8826dea18db5a365feb8b997180b22d72dc
1287f7089a1073a7102c27

sequence number: 2
pt: 426561757479206973207472757468 2c2074727574682062656175747920
aad: 436f756e742d32
nonce: b595dc6b2d7e2ed23af529b3
ct: adf9f6000773035023be7d415e13f84c1cb32a24339a32eb81df02be9ddc6abc880d
d81cceb7c1d0c7781465b2

sequence number: 4
pt: 426561757479206973207472757468 2c2074727574682062656175747920
aad: 436f756e742d34
nonce: b595dc6b2d7e2ed23af529b5
ct: 1f4cc9b7013d65511b1f69c050b7bd8bbd5a5c16ece82b238fec4f30ba2400e7ca8e
e482ac5253cffb5c3dc577

sequence number: 255
pt: 426561757479206973207472757468 2c2074727574682062656175747920
aad: 436f756e742d323535
nonce: b595dc6b2d7e2ed23af5294e
ct: cdc541253111ed7a424eea5134dc14fc5e8293ab3b537668b8656789628e45894e5b
b873c968e3b7cdcbb654a4

sequence number: 256
pt: 426561757479206973207472757468 2c2074727574682062656175747920
aad: 436f756e742d323536
nonce: b595dc6b2d7e2ed23af528b1
ct: faf985208858b1253b97b60aecd28bc18737b58d1242370e7703ec33b73a4c31a1af
ee300e349adef9015bbbfd

A.3.2.2. Exported Values

```
exporter_context:
L: 32
exported_value:
a115a59bf4dd8dc49332d6a0093af8efca1bcbfd3627d850173f5c4a55d0c185

exporter_context: 00
L: 32
exported_value:
4517eaede0669b16aac7c92d5762dd459c301fa10e02237cd5aeb9be969430c4

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
164e02144d44b607a7722e58b0f4156e67c0c2874d74cf71da6ca48a4cbdc5e0
```

A.3.3. Auth Setup Information

```
mode: 2
kem_id: 16
kdf_id: 1
aead_id: 1
info: 4f6465206f6e20612047726563696616e2055726e
ikmE: 798d82a8d9ea19dbc7f2c6dfa54e8a6706f7cdc119db0813dacf8440ab37c857
pkEm: 042224f3ea800f7ec55c03f29fc9865f6ee27004f818fcbdc6dc68932c1e52e15b
79e264a98f2c535ef06745f3d308624414153b22c7332bc1e691cb4af4d53454
skEm: 6b8de0873aed0c1b2d09b8c7ed54cbf24fdf1dfc7a47fa501f918810642d7b91
ikmR: 7bc93bde8890d1fb55220e7f3b0c107ae7e6eda35ca4040bb6651284bf0747ee
pkRm: 04423e363e1cd54ce7b7573110ac121399acbc9ed815fae03b72ffbd4c18b01836
835c5a09513f28fc971b7266cfde2e96afe84bb0f266920e82c4f53b36e1a78d
skRm: d929ab4be2e59f6954d6bedd93e638f02d4046cef21115b00cdda2acb2a4440e
ikmS: 874baa0dcf93595a24a45a7f042e0d22d368747daaa7e19f80a802af19204ba8
pkSm: 04a817a0902bf28e036d66add5d544cc3a0457eab150f104285df1e293b5c10eef
8651213e43d9cd9086c80b309df22cf37609f58c1127f7607e85f210b2804f73
skSm: 1120ac99fb1fccc1e8230502d245719d1b217fe20505c7648795139d177f0de9
enc: 042224f3ea800f7ec55c03f29fc9865f6ee27004f818fcbdc6dc68932c1e52e15b7
9e264a98f2c535ef06745f3d308624414153b22c7332bc1e691cb4af4d53454
shared_secret:
d4aea336439aadf68f9348880aa358086f1480e7c167b6ef15453ba69b94b44f
key_schedule_context: 02b88d4e6d91759e65e87c470e8b9141113e9ad5f0c8ceefc1
e088c82e6980500798e486f9c9c09c9b5c753ac72d6005de254c607d1b534ed11d493ae1
c1d9ac85
secret: fd0a93c7c6f6b1b0dd6a822d7b16f6c61c83d98ad88426df4613c3581a2319f1
key: 19aa8472b3fdc530392b0e54ca17c0f5
base_nonce: b390052d26b67a5b8a8fcaa4
exporter_secret:
f152759972660eb0e1db880835abd5de1c39c8e9cd269f6f082ed80e28acb164
```

A.3.3.1. Encryptions

```
sequence number: 0
pt: 426561757479206973207472757468 2c2074727574682062656175747920
aad: 436f756e742d30
nonce: b390052d26b67a5b8a8fcaa4
ct: 82ffc8c44760db691a07c5627e5fc2c08e7a86979ee79b494a17cc3405446ac2bdb8
f265db4a099ed3289ffe19

sequence number: 1
pt: 426561757479206973207472757468 2c2074727574682062656175747920
aad: 436f756e742d31
nonce: b390052d26b67a5b8a8fcaa5
ct: b0a705a54532c7b4f5907de51c13dffe1e08d55ee9ba59686114b05945494d96725b
239468f1229e3966aa1250

sequence number: 2
pt: 426561757479206973207472757468 2c2074727574682062656175747920
aad: 436f756e742d32
nonce: b390052d26b67a5b8a8fcaa6
ct: 8dc805680e3271a801790833ed74473710157645584f06d1b53ad439078d880b23e2
5256663178271c80ee8b7c

sequence number: 4
pt: 426561757479206973207472757468 2c2074727574682062656175747920
aad: 436f756e742d34
nonce: b390052d26b67a5b8a8fcaa0
ct: 04c8f7aae1584b61aa5816382cb0b834a5d744f420e6dffb5ddcec633a21b8b34728
20930c1ea9258b035937a2

sequence number: 255
pt: 426561757479206973207472757468 2c2074727574682062656175747920
aad: 436f756e742d323535
nonce: b390052d26b67a5b8a8fca5b
ct: 4a319462eaedee37248b4d985f64f4f863d31913fe9e30b6e13136053b69fe5d7085
3c84c60a84bb5495d5a678

sequence number: 256
pt: 426561757479206973207472757468 2c2074727574682062656175747920
aad: 436f756e742d323536
nonce: b390052d26b67a5b8a8fcba4
ct: 28e874512f8940fafc7d06135e7589f6b4198bc0f3a1c64702e72c9e6abaf9f05cb0
d2f11b03a517898815c934
```

A.3.3.2. Exported Values

```
exporter_context:
L: 32
exported_value:
837e49c3ff629250c8d80d3c3fb957725ed481e59e2feb57afd9fe9a8c7c4497

exporter_context: 00
L: 32
exported_value:
594213f9018d614b82007a7021c3135bda7b380da4acd9ab27165c508640dbda

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
14fe634f95ca0d86e15247cca7de7ba9b73c9b9deb6437e1c832daf7291b79d5
```

A.3.4. AuthPSK Setup Information

```
mode: 3
kem_id: 16
kdf_id: 1
aead_id: 1
info: 4f6465206f6e20612047726563696616e2055726e
ikmE: 3c1fceb477ec954c8d58ef3249e4bb4c38241b5925b95f7486e4d9f1d0d35fbb
pkEm: 046a1de3fc26a3d43f4e4ba97dbe24f7e99181136129c48fbe872d4743e2b13135
7ed4f29a7b317dc22509c7b00991ae990bf65f8b236700c82ab7c11a84511401
skEm: 36f771e411cf9cf72f0701ef2b991ce9743645b472e835fe234fb4d6eb2ff5a0
ikmR: abcc2da5b3fa81d8aabd91f7f800a8ccf60ec37b1b585a5d1d1ac77f258b6cca
pkRm: 04d824d7e897897c172ac8a9e862e4bd820133b8d090a9b188b8233a64dfbc5f72
5aa0aa52c8462ab7c9188f1c4872f0c99087a867e8a773a13df48a627058e1b3
skRm: bdf4e2e587afdf0930644a0c45053889ebcadeca662d7c755a353d5b4e2a8394
ikmS: 6262031f040a9db853edd6f91d2272596eabbc78a2ed2bd643f770ecd0f19b82
pkSm: 049f158c750e55d8d5ad13ede66cf6e79801634b7acadcad72044eac2ae1d04800
69133d6488bf73863fa988c4ba8bde1c2e948b761274802b4d8012af4f13af9e
skSm: b0ed8721db6185435898650f7a677affce925aba7975a582653c4cb13c72d240
psk: 0247fd33b913760fa1fa51e1892d9f307fbe65eb171e8132c2af18555a738b82
psk_id: 456e6e796e20447572696e206172616e204d6f726961
enc: 046a1de3fc26a3d43f4e4ba97dbe24f7e99181136129c48fbe872d4743e2b131357
ed4f29a7b317dc22509c7b00991ae990bf65f8b236700c82ab7c11a84511401
shared_secret:
d4c27698391db126f1612d9e91a767f10b9b19aa17e1695549203f0df7d9aebe
key_schedule_context: 03b873cdf2dff4c1434988053b7a775e980dd2039ea24f950b
26b056ccedcb933198e486f9c9c09c9b5c753ac72d6005de254c607d1b534ed11d493ae1
c1d9ac85
secret: 3bf9d4c7955da2740414e73081fa74d6f6f2b4b9645d0685219813ce99a2f270
key: 4d567121d67fae1227d90e11585988fb
base_nonce: 67c9d05330ca21e5116ecda6
exporter_secret:
3f479020ae186788e4dfd4a42a21d24f3faabb224dd4f91c2b2e5e9524ca27b2
```

A.3.4.1. Encryptions

sequence number: 0
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d30
nonce: 67c9d05330ca21e5116ecda6
ct: b9f36d58d9eb101629a3e5a7b63d2ee4af42b3644209ab37e0a272d44365407db8e6
55c72e4fa46f4ff81b9246

sequence number: 1
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d31
nonce: 67c9d05330ca21e5116ecda7
ct: 51788c4e5d56276771032749d015d3eea651af0c7bb8e3da669effffed299ea1f641
df621af65579c10fc09736

sequence number: 2
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d32
nonce: 67c9d05330ca21e5116ecda4
ct: 3b5a2be002e7b29927f06442947e1cf709b9f8508b03823127387223d712703471c2
66efc355f1bc2036f3027c

sequence number: 4
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d34
nonce: 67c9d05330ca21e5116ecda2
ct: 8ddbf1242fe5c7d61e1675496f3bfdb4d90205b3dfbc1b12aab41395d71a82118e09
5c484103107cf4face5123

sequence number: 255
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323535
nonce: 67c9d05330ca21e5116ecd59
ct: 6de25ceadeaec572fbaa25eda2558b73c383fe55106abaec24d518ef6724a7ce698f
83ecdc53e640fe214d2f42

sequence number: 256
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323536
nonce: 67c9d05330ca21e5116ecca6
ct: f380e19d291e12c5e378b51feb5cd50f6d00df6cb2af8393794c4df342126c2e2963
3fe7e8ce49587531affd4d

A.3.4.2. Exported Values

```
exporter_context:
L: 32
exported_value:
595ce0eff405d4b3bb1d08308d70a4e77226ce11766e0a94c4fdb5d90025c978

exporter_context: 00
L: 32
exported_value:
110472ee0ae328f57ef7332a9886a1992d2c45b9b8d5abc9424ff68630f7d38d

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
18ee4d001a9d83a4c67e76f88dd747766576cac438723bad0700a910a4d717e6
```

A.4. DHKEM(P-256, HKDF-SHA256), HKDF-SHA512, AES-128-GCM

A.4.1. Base Setup Information

```
mode: 0
kem_id: 16
kdf_id: 3
aead_id: 1
info: 4f6465206f6e20612047726563696616e2055726e
ikmE: 4ab11a9dd78c39668f7038f921ffc0993b368171d3ddde8031501ee1e08c4c9a
pkEm: 0493ed86735bdfb978cc055c98b45695ad7ce61ce748f4dd63c525a3b8d53a1556
5c6897888070070c1579db1f86aaa56deb8297e64db7e8924e72866f9a472580
skEm: 2292bf14bb6e15b8c81a0f45b7a6e93e32d830e48cca702e0affcfb4d07e1b5c
ikmR: ea9ff7cc5b2705b188841c7ace169290ff312a9cb31467784ca92d7a2e6e1be8
pkRm: 04085aa5b665dc3826f9650ccbcc471be268c8ada866422f739e2d531d4a8818a9
466bc6b449357096232919ec4fe9070ccbac4aac30f4a1a53efcf7af90610edd
skRm: 3ac8530ad1b01885960fab38cf3cdc4f7aef121eaa239f222623614b4079fb38
enc: 0493ed86735bdfb978cc055c98b45695ad7ce61ce748f4dd63c525a3b8d53a15565
c6897888070070c1579db1f86aaa56deb8297e64db7e8924e72866f9a472580
shared_secret:
02f584736390fc93f5b4ad039826a3fa08e9911bd1215a3db8e8791ba533cafd
key_schedule_context: 005b8a3617af7789ee716e7911c7e77f84cdc4cc46e60fb7e1
9e4059f9aeadc00585e26874d1ddde76e551a7679cd47168c466f6e1f705cc9374c19277
8a34fcd5ca221d77e229a9d11b654de7942d685069c633b2362ce3b3d8ea4891c9a2a87a
4eb7cdb289ba5e2ecbf8cd2c8498bb4a383dc021454d70d46fcbbad1252ef4f9
secret: 0c7acdab61693f936c4c1256c78e7be30eebfe466812f9cc49f0b58dc970328d
fc03ea359be0250a471b1635a193d2dfa8cb23c90aa2e25025b892a725353eeb
key: 090ca96e5f8aa02b69fac360da50ddf9
base_nonce: 9c995e621bf9a20c5ca45546
exporter_secret: 4a7abb2ac43e6553f129b2c5750a7e82d149a76ed56dc342d7bca61
e26d494f4855dff0d0165f27ce57756f7f16baca006539bb8e4518987ba610480ac03efa
8
```

A.4.1.1. Encryptions

sequence number: 0
pt: 4265617574792069732074727574682c2074727574682062656175747
9
aad: 436f756e742d30
nonce: 9c995e621bf9a20c5ca45546
ct: d3cf4984931484a080f74c1bb2a6782700dc1fef9abe8442e44a6f09044c88907200
b332003543754eb51917ba

sequence number: 1
pt: 4265617574792069732074727574682c2074727574682062656175747
9
aad: 436f756e742d31
nonce: 9c995e621bf9a20c5ca45547
ct: d14414555a47269dfead9fbf26abb303365e40709a4ed16eaefe1f2070f1ddeb1bdd
94d9e41186f124e0acc62d

sequence number: 2
pt: 4265617574792069732074727574682c2074727574682062656175747
9
aad: 436f756e742d32
nonce: 9c995e621bf9a20c5ca45544
ct: 9bba136cade5c4069707ba91a61932e2cbedda2d9c7bdc33515aa01dd0e0f7e9d357
9bf4016dec37da4aafa800

sequence number: 4
pt: 4265617574792069732074727574682c2074727574682062656175747
9
aad: 436f756e742d34
nonce: 9c995e621bf9a20c5ca45542
ct: a531c0655342be013bf32112951f8df1da643602f1866749519f5dcb09cc68432579
de305a77e6864e862a7600

sequence number: 255
pt: 4265617574792069732074727574682c2074727574682062656175747
9
aad: 436f756e742d323535
nonce: 9c995e621bf9a20c5ca455b9
ct: be5da649469efbad0fb950366a82a73fefeda5f652ec7d3731fac6c4ffa21a7004d2
ab8a04e13621bd3629547d

sequence number: 256
pt: 4265617574792069732074727574682c2074727574682062656175747
9
aad: 436f756e742d323536
nonce: 9c995e621bf9a20c5ca45446
ct: 62092672f5328a0dde095e57435edf7457ace60b26ee44c9291110ec135cb0e14b85
594e4fea11247d937deb62

A.4.1.2. Exported Values

```
exporter_context:
L: 32
exported_value:
a32186b8946f61aeead1c093fe614945f85833b165b28c46bf271abf16b57208

exporter_context: 00
L: 32
exported_value:
84998b304a0ea2f11809398755f0abd5f9d2c141d1822def79dd15c194803c2a

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
93fb9411430b2cfa2cf0bed448c46922a5be9beff20e2e621df7e4655852edbc
```

A.4.2. PSK Setup Information

```
mode: 1
kem_id: 16
kdf_id: 3
aead_id: 1
info: 4f6465206f6e20612047726563696616e2055726e
ikmE: c11d883d6587f911d2ddbc2a0859d5b42fb13bf2c8e89ef408a25564893856f5
pkEm: 04a307934180ad5287f95525fe5bc6244285d7273c15e061f0f2efb211c35057f3
079f6e0abae200992610b25f48b63aacfcb669106ddee8aa023feed301901371
skEm: a5901ff7d6931959c2755382ea40a4869b1dec3694ed3b009dda2d77dd488f18
ikmR: 75bfc2a3a3541170a54c0b06444e358d0ee2b4fb78a401fd399a47a33723b700
pkRm: 043f5266fba0742db649e1043102b8a5afd114465156719cea90373229aabdd84d
7f45dabfc1f55664b888a7e86d594853a6cccdc9b189b57839cbbe3b90b55873
skRm: bc6f0b5e22429e5ff47d5969003f3cae0f4fec50e23602e880038364f33b8522
psk: 0247fd33b913760fa1fa51e1892d9f307fbe65eb171e8132c2af18555a738b82
psk_id: 456e6e796e20447572696e206172616e204d6f726961
enc: 04a307934180ad5287f95525fe5bc6244285d7273c15e061f0f2efb211c35057f30
79f6e0abae200992610b25f48b63aacfcb669106ddee8aa023feed301901371
shared_secret:
2912aacc6eaebd71ff715ea50f6ef3a6637856b2a4c58ea61e0c3fc159e3bc16
key_schedule_context: 01713f73042575cebfd132f0cc4338523f8eae95c80a749f7c
f3eb9436ff1c612ca62c37df27ca46d2cc162445a92c5f5fdc57bcde129ca7b1f284b0c1
2297c037ca221d77e229a9d11b654de7942d685069c633b2362ce3b3d8ea4891c9a2a87a
4eb7cdb289ba5e2ecbf8cd2c8498bb4a383dc021454d70d46fcbbad1252ef4f9
secret: ff2051d2128d5f3078de867143e076262ce1d0aecafc3fff3d607f1eaff05345
c7d5ffcb3202cdecb3d1a2f7da20592a237747b6e855390cbe2109d3e6ac70c2
key: 0b910ba8d9cfa17e5f50c211cb32839a
base_nonce: 0c29e714eb52de5b7415a1b7
exporter_secret: 50c0a182b6f94b4c0bd955c4aa20df01f282cc12c43065a0812fe4d
4352790171ed2b2c4756ad7f5a730ba336c8f1edd0089d8331192058c385bae39c7cc8b5
7
```

A.4.2.1. Encryptions

sequence number: 0
pt: 4265617574792069732074727574682c2074727574682062656175747
9
aad: 436f756e742d30
nonce: 0c29e714eb52de5b7415a1b7
ct: 57624b6e320d4aba0afd11f548780772932f502e2ba2a8068676b2a0d3b5129a45b9
faa88de39e8306da41d4cc

sequence number: 1
pt: 4265617574792069732074727574682c2074727574682062656175747
9
aad: 436f756e742d31
nonce: 0c29e714eb52de5b7415a1b6
ct: 159d6b4c24bacaf2f5049b7863536d8f3ffede76302dace42080820fa51925d4e1c7
2a64f87b14291a3057e00a

sequence number: 2
pt: 4265617574792069732074727574682c2074727574682062656175747
9
aad: 436f756e742d32
nonce: 0c29e714eb52de5b7415a1b5
ct: bd24140859c99bf0055075e9c460032581dd1726d52cf980d308e9b20083ca62e700
b17892bcf7fa82bac751d0

sequence number: 4
pt: 4265617574792069732074727574682c2074727574682062656175747
9
aad: 436f756e742d34
nonce: 0c29e714eb52de5b7415a1b3
ct: 93ddd55f82e9aaaa3cfc06840575f09d80160b20538125c2549932977d1238dde812
6a4a91118faf8632f62cb8

sequence number: 255
pt: 4265617574792069732074727574682c2074727574682062656175747
9
aad: 436f756e742d323535
nonce: 0c29e714eb52de5b7415a148
ct: 377a98a3c34bf716581b05a6b3fdc257f245856384d5f2241c8840571c52f5c85c21
138a4a81655edab8fe227d

sequence number: 256
pt: 4265617574792069732074727574682c2074727574682062656175747
9
aad: 436f756e742d323536
nonce: 0c29e714eb52de5b7415a0b7
ct: cc161f5a179831d456d119d2f2c19a6817289c75d1c61cd37ac8a450acd9efba02e0
ac00d128c17855931ff69a

A.4.2.2. Exported Values

```
exporter_context:
L: 32
exported_value:
8158bea21a6700d37022bb7802866edca30ebf2078273757b656ef7fc2e428cf

exporter_context: 00
L: 32
exported_value:
6a348ba6e0e72bb3ef22479214a139ef8dac57be34509a61087a12565473da8d

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
2f6d4f7a18ec48de1ef4469f596aada4afdf6d79b037ed3c07e0118f8723bffc
```

A.4.3. Auth Setup Information

```
mode: 2
kem_id: 16
kdf_id: 3
aead_id: 1
info: 4f6465206f6e20612047726563696616e2055726e
ikmE: 6bb031aa9197562da0b44e737db2b9e61f6c3ea1138c37de28fc37ac29bc7350
pkEm: 04fec59fa9f76f5d0f6c1660bb179cb314ed97953c53a60ab38f8e6ace60fd5917
8084d0dd66e0f79172992d4ddb2e91172ce24949bcebfff158dcc417f2c6e9c6
skEm: 93cddd5288e7ef4884c8fe321d075df01501b993ff49ffab8184116f39b3c655
ikmR: 649a3f92edbb7a2516a0ade0b7dccc58a37240c4ba06f9726a952227b4adf6ff
pkRm: 04378bad519aab406e04d0e5608bcca809c02d6afd2272d4dd03e9357bd0eee8ad
f84c8deba3155c9cf9506d1d4c8bfefe3cf033a75716cc3cc07295100ec96276
skRm: 1ea4484be482bf25fdb2ed39e6a02ed9156b3e57dfb18dff82e4a048de990236
ikmS: 4d79b8691aab55a7265e8490a04bb3860ed64dece90953ad0dc43a6ea59b4bf2
pkSm: 0404d3c1f9fca22eb4a6d326125f0814c35593b1da8ea0d11a640730b215a259b9
b98a34ad17e21617d19fe1d4fa39a4828bfdb306b729ec51c543caca3b2d9529
skSm: 02b266d66919f7b08f42ae0e7d97af4ca98b2dae3043bb7e0740ccadc1957579
enc: 04fec59fa9f76f5d0f6c1660bb179cb314ed97953c53a60ab38f8e6ace60fd59178
084d0dd66e0f79172992d4ddb2e91172ce24949bcebfff158dcc417f2c6e9c6
shared_secret:
1ed49f6d7ada333d171cd63861a1cb700a1ec4236755a9cd5f9f8f67a2f8e7b3
key_schedule_context: 025b8a3617af7789ee716e7911c7e77f84cdc4cc46e60fb7e1
9e4059f9aeadc00585e26874d1ddde76e551a7679cd47168c466f6e1f705cc9374c19277
8a34fcd5ca221d77e229a9d11b654de7942d685069c633b2362ce3b3d8ea4891c9a2a87a
4eb7cdb289ba5e2ecbf8cd2c8498bb4a383dc021454d70d46fcbbad1252ef4f9
secret: 9c846ba81ddbbd57bc26d99da6cf7ab956bb735ecd47fe21ed14241c70791b74
84c1d06663d21a5d97bf1be70d56ab727f650c4f859c5ed3f71f8928b3c082dd
key: 9d4b1c83129f3de6db95faf3d539dcf1
base_nonce: ea4fd7a485ee5f1f4b62c1b7
exporter_secret: ca2410672369aae1afd6c2639f4fe34ca36d35410c090608d2924f6
0def17f910d7928575434d7f991b1f19d3e8358b8278ff59ced0d5eed4774cec72e12766
e
```

A.4.3.1. Encryptions

```
sequence number: 0
pt: 426561757479206973207472757468 2c20747275746820626561757479
aad: 436f756e742d30
nonce: ea4fd7a485ee5f1f4b62c1b7
ct: 2480179d880b5f458154b8bfe3c7e8732332de84aabf06fc440f6b31f169e154157f
a9eb44f2fa4d7b38a9236e

sequence number: 1
pt: 426561757479206973207472757468 2c20747275746820626561757479
aad: 436f756e742d31
nonce: ea4fd7a485ee5f1f4b62c1b6
ct: 10cd81e3a816d29942b602a92884348171a31cbd0f042c3057c65cd93c540943a5b0
5115bd520c09281061935b

sequence number: 2
pt: 426561757479206973207472757468 2c20747275746820626561757479
aad: 436f756e742d32
nonce: ea4fd7a485ee5f1f4b62c1b5
ct: 920743a88d8cf6a09e1a3098e8be8edd09db136e9d543f215924043af8c7410f68ce
6aa64fd2b1a176e7f6b3fd

sequence number: 4
pt: 426561757479206973207472757468 2c20747275746820626561757479
aad: 436f756e742d34
nonce: ea4fd7a485ee5f1f4b62c1b3
ct: 6b11380fcc708fc8589effb5b5e0394cbd441fa5e240b5500522150ca8265d65ff55
479405af936e2349119dcd

sequence number: 255
pt: 426561757479206973207472757468 2c20747275746820626561757479
aad: 436f756e742d323535
nonce: ea4fd7a485ee5f1f4b62c148
ct: d084eca50e7554bb97ba34c4482dfe32c9a2b7f3ab009c2d1b68ecbf97bee2d28cd9
4b6c829b96361f2701772d

sequence number: 256
pt: 426561757479206973207472757468 2c20747275746820626561757479
aad: 436f756e742d323536
nonce: ea4fd7a485ee5f1f4b62c0b7
ct: 247da592cc4ce834a94de2c79f5730ee49342470a021e4a4bc2bb77c53b17413e94d
94f57b4fdaedcf97cfe7b1
```

A.4.3.2. Exported Values

```
exporter_context:
L: 32
exported_value:
f03fbc82f321a0ab4840e487cb75d07aafd8e6f68485e4f7ff72b2f55ff24ad6

exporter_context: 00
L: 32
exported_value:
1ce0cadec0a8f060f4b5070c8f8888dcdfefc2e35819df0cd559928a11ff0891

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
70c405c707102fd0041ea716090753be47d68d238b111d542846bd0d84ba907c
```

A.4.4. AuthPSK Setup Information

```
mode: 3
kem_id: 16
kdf_id: 3
aead_id: 1
info: 4f6465206f6e20612047726563696616e2055726e
ikmE: 37ae06a521cd555648c928d7af58ad2aa4a85e34b8cabd069e94ad55ab872cc8
pkEm: 04801740f4b1b35823f7fb2930eac2efc8c4893f34ba111c0bb976e3c7d5dc0aef
5a7ef0bf4057949a140285f774f1efc53b3860936b92279a11b68395d898d138
skEm: 778f2254ae5d661d5c7fca8c4a7495a25bd13f26258e459159f3899df0de76c1
ikmR: 7466024b7e2d2366c3914d7833718f13afb9e3e45bcfbb510594d614ddd9b4e7
pkRm: 04a4ca7af2fc2cce48edbf2f1700983e927743a4e85bb5035ad562043e25d9a111
cbf6f7385fac55edc5c9d2ca6ed351a5643de95c36748e11dbec98730f4d43e9
skRm: 00510a70fde67af487c093234fc4215c1cdec09579c4b30cc8e48cb530414d0e
ikmS: ee27aaf99bf5cd8398e9de88ac09a82ac22cdb8d0905ab05c0f5fa12ba1709f3
pkSm: 04b59a4157a9720eb749c95f842a5e3e8acdccbe834426d405509ac3191e23f216
5b5bb1f07a6240dd567703ae75e13182ee0f69fc102145cdb5abf681ff126d60
skSm: d743b20821e6326f7a26684a4beed7088b35e392114480ca9f6c325079dcf10b
psk: 0247fd33b913760fa1fa51e1892d9f307fbe65eb171e8132c2af18555a738b82
psk_id: 456e6e796e20447572696e206172616e204d6f726961
enc: 04801740f4b1b35823f7fb2930eac2efc8c4893f34ba111c0bb976e3c7d5dc0aef5
a7ef0bf4057949a140285f774f1efc53b3860936b92279a11b68395d898d138
shared_secret:
02bee8be0dda755846115db45071c0cf59c25722e015bde1c124de849c0fea52
key_schedule_context: 03713f73042575cebfd132f0cc4338523f8eae95c80a749f7c
f3eb9436ff1c612ca62c37df27ca46d2cc162445a92c5f5fdc57bcde129ca7b1f284b0c1
2297c037ca221d77e229a9d11b654de7942d685069c633b2362ce3b3d8ea4891c9a2a87a
4eb7cdb289ba5e2ecbf8cd2c8498bb4a383dc021454d70d46fcbbad1252ef4f9
secret: 0f9df08908a6a3d06c8e934cd3f5313f9ebccd0986e316c0198bb48bed30dc3d
b2f3baab94fd40c2c285c7288c77e2255401ee2d5884306addf4296b93c238b3
key: b68bb0e2fbf7431cedb46cc3b6f1fe9e
base_nonce: 76af62719d33d39a1cb6be9f
exporter_secret: 7f72308ae68c9a2b3862e686cb547b16d33d00fe482c770c4717d8b
54e9b1e547244c3602bdd86d5a788a8443befea0a7658002b23f1c96a62a64986fffc511
a
```

A.4.4.1. Encryptions

sequence number: 0
pt: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d30
nonce: 76af62719d33d39a1cb6be9f
ct: 840669634db51e28df54f189329c1b727fd303ae413f003020aff5e26276aaa910fc
4296828cb9d862c2fd7d16

sequence number: 1
pt: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d31
nonce: 76af62719d33d39a1cb6be9e
ct: d4680a48158d9a75fd09355878d6e33997a36ee01d4a8f22032b22373b795a941b7b
9c5205ff99e0ff284beef4

sequence number: 2
pt: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d32
nonce: 76af62719d33d39a1cb6be9d
ct: c45eb6597de2bac929a0f5d404ba9d2dc1ea031880930f1fd7a283f0a0cbebb35eac
1a9ee0d1225f5e0f181571

sequence number: 4
pt: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d34
nonce: 76af62719d33d39a1cb6be9b
ct: 4ee2482ad8d7d1e9b7e651c78b6ca26d3c5314d0711710ca62c2fd8bb8996d7d8727
c157538d5493da696b61f8

sequence number: 255
pt: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d323535
nonce: 76af62719d33d39a1cb6be60
ct: 65596b731df010c76a915c6271a438056ce65696459432eeafdae7b4cadb6290dd61
e68edd4e40b659d2a8cbcc

sequence number: 256
pt: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d323536
nonce: 76af62719d33d39a1cb6bf9f
ct: 9f659482ebc52f8303f9eac75656d807ec38ce2e50c72e3078cd13d86b30e3f89069
0a873277620f8a6a42d836

A.4.4.2. Exported Values

exporter_context:
L: 32
exported_value:
c8c917e137a616d3d4e4c9fcd9c50202f366cb0d37862376bc79f9b72e8a8db9

exporter_context: 00
L: 32
exported_value:
33a5d4df232777008a06d0684f23bb891cfaef702f653c8601b6ad4d08dddddf

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
bed80f2e54f1285895c4a3f3b3625e6206f78f1ed329a0cfb5864f7c139b3c6a

A.5. DHKEM(P-256, HKDF-SHA256), HKDF-SHA256, ChaCha20Poly1305

A.5.1. Base Setup Information

mode: 0
kem_id: 16
kdf_id: 1
aead_id: 3
info: 4f6465206f6e20612047726563696616e2055726e
ikmE: f1f1a3bc95416871539ecb51c3a8f0cf608afb40fbbe305c0a72819d35c33f1f
pkEm: 04c07836a0206e04e31d8ae99bfd549380b072a1b1b82e563c935c095827824fc1
559eac6fb9e3c70cd3193968994e7fe9781aa103f5b50e934b5b2f387e381291
skEm: 7550253e1147aae48839c1f8af80d2770fb7a4c763afe7d0afa7e0f42a5b3689
ikmR: 61092f3f56994dd424405899154a9918353e3e008171517ad576b900ddb275e7
pkRm: 04a697bffde9405c992883c5c439d6cc358170b51af72812333b015621dc0f40ba
d9bb726f68a5c013806a790ec716ab8669f84f6b694596c2987cf35baba2a006
skRm: a4d1c55836aa30f9b3fbb6ac98d338c877c2867dd3a77396d13f68d3ab150d3b
enc: 04c07836a0206e04e31d8ae99bfd549380b072a1b1b82e563c935c095827824fc15
59eac6fb9e3c70cd3193968994e7fe9781aa103f5b50e934b5b2f387e381291
shared_secret:
806520f82ef0b03c823b7fc524b6b55a088f566b9751b89551c170f4113bd850
key_schedule_context: 00b738cd703db7b4106e93b4621e9a19c89c838e55964240e5
d3f331aaf8b0d58b2e986ea1c671b61cf45eec134dac0bae58ec6f63e790b1400b47c330
38b0269c
secret: fe891101629aa355aad68eff3cc5170d057eca0c7573f6575e91f9783e1d4506
key: a8f45490a92a3b04d1dbf6cf2c3939ad8bfc9bfcb97c04bffe116730c9dfe3fc
base_nonce: 726b4390ed2209809f58c693
exporter_secret:
4f9bd9b3a8db7d7c3a5b9d44fdc1f6e37d5d77689ade5ec44a7242016e6aa205

A.5.1.1. Encryptions

```
sequence number: 0
pt: 426561757479206973207472757468 2c207472757468 20626561757479
aad: 436f756e742d30
nonce: 726b4390ed2209809f58c693
ct: 6469c41c5c81d3aa85432531ecf6460ec945bde1eb428cb2fedf7a29f5a685b4ccb0
d057f03ea2952a27bb458b

sequence number: 1
pt: 426561757479206973207472757468 2c207472757468 20626561757479
aad: 436f756e742d31
nonce: 726b4390ed2209809f58c692
ct: f1564199f7e0e110ec9c1bcdde332177fc35c1adf6e57f8d1df24022227ffa871686
2dbda2b1dc546c9d114374

sequence number: 2
pt: 426561757479206973207472757468 2c207472757468 20626561757479
aad: 436f756e742d32
nonce: 726b4390ed2209809f58c691
ct: 39de89728bcb774269f882af8dc5369e4f3d6322d986e872b3a8d074c7c18e8549ff
3f85b6d6592ff87c3f310c

sequence number: 4
pt: 426561757479206973207472757468 2c207472757468 20626561757479
aad: 436f756e742d34
nonce: 726b4390ed2209809f58c697
ct: bc104a14fbede0cc79eeb826ea0476ce87b9c928c36e5e34dc9b6905d91473ec369a
08b1a25d305dd45c6c5f80

sequence number: 255
pt: 426561757479206973207472757468 2c207472757468 20626561757479
aad: 436f756e742d323535
nonce: 726b4390ed2209809f58c66c
ct: 8f2814a2c548b3be50259713c6724009e092d37789f6856553d61df23ebc079235f7
10e6af3c3ca6eaba7c7c6c

sequence number: 256
pt: 426561757479206973207472757468 2c207472757468 20626561757479
aad: 436f756e742d323536
nonce: 726b4390ed2209809f58c793
ct: b45b69d419a9be7219d8c94365b89ad6951caf4576ea4774ea40e9b7047a09d6537d
1aa2f7c12d6ae4b729b4d0
```

A.5.1.2. Exported Values

```
exporter_context:
L: 32
exported_value:
9b13c510416ac977b553bf1741018809c246a695f45eff6d3b0356dbefe1e660

exporter_context: 00
L: 32
exported_value:
6c8b7be3a20a5684edecb4253619d9051ce8583baf850e0cb53c402bdcaf8ebb

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
477a50d804c7c51941f69b8e32fe8288386ee1a84905fe4938d58972f24ac938
```

A.5.2. PSK Setup Information

```
mode: 1
kem_id: 16
kdf_id: 1
aead_id: 3
info: 4f6465206f6e20612047726563696616e2055726e
ikmE: e1a4e1d50c4bfcf890f2b4c7d6b2d2aca61368eddc3c84162df2856843e1057a
pkEm: 04f336578b72ad7932fe867cc4d2d44a718a318037a0ec271163699cee653fa805
c1fec955e562663e0c2061bb96a87d78892bff0cc0bad7906c2d998ebe1a7246
skEm: 7d6e4e006cee68af9b3fdd583a0ee8962df9d59fab029997ee3f456cbc857904
ikmR: ee51dec304abf993ef8fd52aacdd3b539108bbf6e491943266c1de89ec596a17
pkRm: 041eb8f4f20ab72661af369ff3231a733672fa26f385ffb959fd1bae46bfda43ad
55e2d573b880831381d9367417f554ce5b2134fbba5235b44db465feffc6189e
skRm: 12ecde2c8bc2d5d7ed2219c71f27e3943d92b344174436af833337c557c300b3
psk: 0247fd33b913760fa1fa51e1892d9f307fbe65eb171e8132c2af18555a738b82
psk_id: 456e6e796e20447572696e206172616e204d6f726961
enc: 04f336578b72ad7932fe867cc4d2d44a718a318037a0ec271163699cee653fa805c
1fec955e562663e0c2061bb96a87d78892bff0cc0bad7906c2d998ebe1a7246
shared_secret:
ac4f260dce4db6bf45435d9c92c0e11cfdd93743bd3075949975974cc2b3d79e
key_schedule_context: 01622b72afcc3795841596c67ea74400ca3b029374d7d5640b
da367c5d67b3fbeb2e986ea1c671b61cf45eec134dac0bae58ec6f63e790b1400b47c330
38b0269c
secret: 858c8087a1c056db5811e85802f375bb0c19b9983204a1575de4803575d23239
key: 6d61cb330b7771168c8619498e753f16198aad9566d1f1c6c70e2bc1a1a8b142
base_nonce: 0de7655fb65e1cd51a38864e
exporter_secret:
754ca00235b245e72d1f722a7718e7145bd113050a2aa3d89586d4cb7514bfdb
```

A.5.2.1. Encryptions

sequence number: 0
pt: 426561757479206973207472757468 2c20747275746820626561757479
aad: 436f756e742d30
nonce: 0de7655fb65e1cd51a38864e
ct: 21433eaff24d7706f3ed5b9b2e709b07230e2b11df1f2b1fe07b3c70d5948a53d6fa
5c8bed194020bd9df0877b

sequence number: 1
pt: 426561757479206973207472757468 2c20747275746820626561757479
aad: 436f756e742d31
nonce: 0de7655fb65e1cd51a38864f
ct: c74a764b4892072ea8c2c56b9bcd46c7f1e9ca8cb0a263f8b40c2ba59ac9c857033f
176019562218769d3e0452

sequence number: 2
pt: 426561757479206973207472757468 2c20747275746820626561757479
aad: 436f756e742d32
nonce: 0de7655fb65e1cd51a38864c
ct: dc8cd68863474d6e9cbb6a659335a86a54e036249d41acf909e738c847ff2bd36fe3
fcacda4ededa7032c0a220

sequence number: 4
pt: 426561757479206973207472757468 2c20747275746820626561757479
aad: 436f756e742d34
nonce: 0de7655fb65e1cd51a38864a
ct: cd54a8576353b1b9df366cb0cc042e46eef6f4cf01e205fe7d47e306b2fdd90f7185
f289a26c613ca094e3be10

sequence number: 255
pt: 426561757479206973207472757468 2c20747275746820626561757479
aad: 436f756e742d323535
nonce: 0de7655fb65e1cd51a3886b1
ct: 6324570c9d542c70c7e70570c1d8f4c52a89484746bf0625441890ededcc80c24ef2
301c38bfd34d689d19f67d

sequence number: 256
pt: 426561757479206973207472757468 2c20747275746820626561757479
aad: 436f756e742d323536
nonce: 0de7655fb65e1cd51a38874e
ct: 1ea6326c8098ed0437a553c466550114fb2ca1412cca7de98709b9ccdf19206e52c3
d39180e2cf62b3e9f4baf4

A.5.2.2. Exported Values

```
exporter_context:
L: 32
exported_value:
530bbc2f68f078dccc89cc371b4f4ade372c9472bafe4601a8432cbb934f528d

exporter_context: 00
L: 32
exported_value:
6e25075ddcc528c90ef9218f800ca3dfe1b8ff4042de5033133adb8bd54c401d

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
6f6fbd0d1c7733f796461b3235a856cc34f676fe61ed509dfc18fa16efe6be78
```

A.5.3. Auth Setup Information

```
mode: 2
kem_id: 16
kdf_id: 1
aead_id: 3
info: 4f6465206f6e2061204772656369616e2055726e
ikmE: 0ecd212019008138a31f9104d5dba76b9f8e34d5b996041fff9e3df221dd0d5d
pkEm: 040d5176aedba55bc41709261e9195c5146bb62d783031280775f32e507d79b5cb
c5748b6be6359760c73cfe10ca19521af704ca6d91ff32fc0739527b9385d415
skEm: 085fd5d5e6ce6497c79df960cac93710006b76217d8bcfafbd2bb2c20ea03c42
ikmR: d32236d8378b9563840653789eb7bc33c3c720e537391727bf1c812d0eac110f
pkRm: 0444f6ee41818d9fe0f8265bffd016b7e2dd3964d610d0f7514244a60dbb7a11ec
e876bb110a97a2ac6a9542d7344bf7d2bd59345e3e75e497f7416cf38d296233
skRm: 3cb2c125b8c5a81d165a333048f5dcae29a2ab2072625adad66dbb0f48689af9
ikmS: 0e6be0851283f9327295fd49858a8c8908ea9783212945eef6c598ee0a3cedbb
pkSm: 04265529a04d4f46ab6fa3af4943774a9f1127821656a75a35fade898a9a1b014f
64d874e88cddb24c1c3d79004d3a587db67670ca357ff4fba7e8b56ec013b98b
skSm: 39b19402e742d48d319d24d68e494daa4492817342e593285944830320912519
enc: 040d5176aedba55bc41709261e9195c5146bb62d783031280775f32e507d79b5cbc
5748b6be6359760c73cfe10ca19521af704ca6d91ff32fc0739527b9385d415
shared_secret:
1a45aa4792f4b166bfee7eeab0096c1a6e497480e2261b2a59aad12f2768d469
key_schedule_context: 02b738cd703db7b4106e93b4621e9a19c89c838e55964240e5
d3f331aaf8b0d58b2e986ea1c671b61cf45eec134dac0bae58ec6f63e790b1400b47c330
38b0269c
secret: 9193210815b87a4c5496c9d73e609a6c92665b5ea0d760866294906d089ebb57
key: cf292f8a4313280a462ce55cde05b5aa5744fe4ca89a5d81b0146a5eaca8092d
base_nonce: 7e45c21e20e869ae00492123
exporter_secret:
dba6e307f71769ba11e2c687cc19592f9d436da0c81e772d7a8a9fd28e54355f
```

A.5.3.1. Encryptions

```
sequence number: 0
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d30
nonce: 7e45c21e20e869ae00492123
ct: 25881f219935eec5ba70d7b421f13c35005734f3e4d959680270f55d71e2f5cb3bd2
daced2770bf3d9d4916872

sequence number: 1
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d31
nonce: 7e45c21e20e869ae00492122
ct: 653f0036e52a376f5d2dd85b3204b55455b7835c231255ae098d09ed138719b97185
129786338ab6543f753193

sequence number: 2
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d32
nonce: 7e45c21e20e869ae00492121
ct: 60878706117f22180c788e62df6a595bc41906096a11a9513e84f0141e43239e81a9
8d7a235abc64112fcb8ddd

sequence number: 4
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d34
nonce: 7e45c21e20e869ae00492127
ct: 0f9094dd08240b5fa7a388b824d19d5b4b1e126cebfd67a062c32f9ba9f1f3866cc3
8de7df2702626e2ab65c0f

sequence number: 255
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323535
nonce: 7e45c21e20e869ae004921dc
ct: dd29319e08135c5f8401d6537a364e92172c0e3f095f3fd18923881d11c0a6839345
dd0b54acd0edd8f8344792

sequence number: 256
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323536
nonce: 7e45c21e20e869ae00492023
ct: e2276ec5047bc4b6ed57d6da7da2fb47a77502f0a30f17d040247c73da336d722bc6
c89adf68396a0912c6d152
```

A.5.3.2. Exported Values

```
exporter_context:
L: 32
exported_value:
56c4d6c1d3a46c70fd8f4ecda5d27c70886e348efb51bd5edeaa39ff6ce34389

exporter_context: 00
L: 32
exported_value:
d2d3e48ed76832b6b3f28fa84be5f11f09533c0e3c71825a34fb0f1320891b51

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
eb0d312b6263995b4c7761e64b688c215ffd6043ff3bad2368c862784cbe6eff
```

A.5.4. AuthPSK Setup Information

```
mode: 3
kem_id: 16
kdf_id: 1
aead_id: 3
info: 4f6465206f6e20612047726563696616e2055726e
ikmE: f3a07f194703e321ef1f753a1b9fe27a498dfdfa309151d70bedd896c239c499
pkEm: 043539917ee26f8ae0aa5f784a387981b13de33124a3cde88b94672030183110f3
31400115855808244ff0c5b6ca6104483ac95724481d41bdcd9f15b430ad16f6
skEm: 11b7e4de2d919240616a31ab14944cced79bc2372108bb98f6792e3b645fe546
ikmR: 1240e55a0a03548d7f963ef783b6a7362cb505e6b31dfd04c81d9b294543bfbd
pkRm: 04d383fd920c42d018b9d57fd73a01f1eee480008923f67d35169478e55d2e8817
068daf62a06b10e0aad4a9e429fa7f904481be96b79a9c231a33e956c20b81b6
skRm: c29fc577b7e74d525c0043f1c27540a1248e4f2c8d297298e99010a92e94865c
ikmS: ce2a0387a2eb8870a3a92c34a2975f0f3f271af4384d446c7dc1524a6c6c515a
pkSm: 0492cf8c9b144b742fe5a63d9a181a19d416f3ec8705f24308ad316564823c344e
018bd7c03a33c926bb271b28ef5bf28c0ca00abff249fee5ef7f33315ff34fdb
skSm: 53541bd995f874a67f8bfd8038afa67fd68876801f42ff47d0dc2a4deea067ae
psk: 0247fd33b913760fa1fa51e1892d9f307fbe65eb171e8132c2af18555a738b82
psk_id: 456e6e796e20447572696e206172616e204d6f726961
enc: 043539917ee26f8ae0aa5f784a387981b13de33124a3cde88b94672030183110f33
1400115855808244ff0c5b6ca6104483ac95724481d41bdcd9f15b430ad16f6
shared_secret:
87584311791036a3019bc36803cdd42e9a8931a98b13c88835f2f8a9036a4fd6
key_schedule_context: 03622b72afcc3795841596c67ea74400ca3b029374d7d5640b
da367c5d67b3fbeb2e986ea1c671b61cf45eec134dac0bae58ec6f63e790b1400b47c330
38b0269c
secret: fe52b4412590e825ea2603fa88e145b2ee014b942a774b55fab4f081301f16f4
key: 31e140c8856941315d4067239fdc4ebe077fbf45a6fc78a61e7a6c8b3bacb10a
base_nonce: 75838a8010d2e4760254dd56
exporter_secret:
600895965755db9c5027f25f039a6e3e506c35b3b7084ce33c4a48d59ee1f0e3
```

A.5.4.1. Encryptions

sequence number: 0
pt: 426561757479206973207472757468 2c2074727574682062656175747920
aad: 436f756e742d30
nonce: 75838a8010d2e4760254dd56
ct: 9eadfa0f954835e7e920ffe56dec6b31a046271cf71fdda55db72926e1d8fae94cc6
280fcfabd8db71eaa65c05

sequence number: 1
pt: 426561757479206973207472757468 2c2074727574682062656175747920
aad: 436f756e742d31
nonce: 75838a8010d2e4760254dd57
ct: e357ad10d75240224d4095c9f6150a2ed2179c0f878e4f2db8ca95d365d174d059ff
8c3eb38ea9a65cfc8eaeb8

sequence number: 2
pt: 426561757479206973207472757468 2c2074727574682062656175747920
aad: 436f756e742d32
nonce: 75838a8010d2e4760254dd54
ct: 2fa56d00f8dd479d67a2ec3308325cf3bbccaf102a64ffccdb006bd7dcb932685b9a
7b49cdc094a85fec1da5ef

sequence number: 4
pt: 426561757479206973207472757468 2c2074727574682062656175747920
aad: 436f756e742d34
nonce: 75838a8010d2e4760254dd52
ct: 1fe9d6db14965003ed81a39abf240f9cd7c5a454bca0d69ef9a2de16d537364fbbf1
10b9ef11fa4a7a0172f0ce

sequence number: 255
pt: 426561757479206973207472757468 2c2074727574682062656175747920
aad: 436f756e742d323535
nonce: 75838a8010d2e4760254dda9
ct: eaf4041a5c9122b22d1f8d698eeffe45d64b4ae33d0ddca3a4cdf4a5f595acc95a1a
9334d06cc4d000df6aaad6

sequence number: 256
pt: 426561757479206973207472757468 2c2074727574682062656175747920
aad: 436f756e742d323536
nonce: 75838a8010d2e4760254dc56
ct: fb857f4185ce5286c1a52431867537204963ea66a3eee8d2a74419fd8751faee066d
08277ac7880473aa4143ba

A.5.4.2. Exported Values

```
exporter_context:
L: 32
exported_value:
c52b4592cd33dd38b2a3613108ddda28dcf7f03d30f2a09703f758bfa8029c9a

exporter_context: 00
L: 32
exported_value:
2f03bebc577e5729e148554991787222b5c2a02b77e9b1ac380541f710e5a318

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
e01dd49e8bfc3d9216abc1be832f0418adf8b47a7b5a330a7436c31e33d765d7
```

A.6. DHKEM(P-521, HKDF-SHA512), HKDF-SHA512, AES-256-GCM

A.6.1. Base Setup Information

```
mode: 0
kem_id: 18
kdf_id: 3
aead_id: 2
info: 4f6465206f6e20612047726563369616e2055726e
ikmE: 7f06ab8215105fc46aceeb2e3dc5028b44364f960426eb0d8e4026c2f8b5d7e7a9
86688f1591abf5ab753c357a5d6f0440414b4ed4ede71317772ac98d9239f70904
pkEm: 040138b385ca16bb0d5fa0c0665fbbd7e69e3ee29f63991d3e9b5fa740aab8900a
aeed46ed73a49055758425a0ce36507c54b29cc5b85a5cee6bae0cf1c21f2731ece2013d
c3fb7c8d21654bb161b463962ca19e8c654ff24c94dd2898de12051f1ed0692237fb02b2
f8d1dc1c73e9b366b529eb436e98a996ee522aef863dd5739d2f29b0
skEm: 014784c692da35df6ecde98ee43ac425dbdd0969c0c72b42f2e708ab9d535415a8
569bdacfcc0a114c85b8e3f26acf4d68115f8c91a66178cdbd03b7bcc5291e374b
ikmR: 2ad954bbe39b7122529f7dde780bff626cd97f850d0784a432784e69d86eccaade
43b6c10a8ffdb94bf943c6da479db137914ec835a7e715e36e45e29b587bab3bf1
pkRm: 0401b45498c1714e2dce167d3caf162e45e0642afc7ed435df7902ccae0e84ba0f
7d373f646b7738bbbdca11ed91bdeae3cdcba3301f2457be452f271fa6837580e661012a
f49583a62e48d44bed350c7118c0d8dc861c238c72a2bda17f64704f464b57338e7f40b6
0959480c0e58e6559b190d81663ed816e523b6b6a418f66d2451ec64
skRm: 01462680369ae375e4b3791070a7458ed527842f6a98a79ff5e0d4cbde83c27196
a3916956655523a6a2556a7af62c5cadabe2ef9da3760bb21e005202f7b2462847
enc: 040138b385ca16bb0d5fa0c0665fbbd7e69e3ee29f63991d3e9b5fa740aab8900aa
eed46ed73a49055758425a0ce36507c54b29cc5b85a5cee6bae0cf1c21f2731ece2013dc
3fb7c8d21654bb161b463962ca19e8c654ff24c94dd2898de12051f1ed0692237fb02b2f
8d1dc1c73e9b366b529eb436e98a996ee522aef863dd5739d2f29b0
shared_secret: 776ab421302f6eff7d7cb5cb1adaea0cd50872c71c2d63c30c4f1d5e4
3653336fef33b103c67e7a98add2d3b66e2fda95b5b2a667aa9dac7e59cc1d46d30e818
key_schedule_context: 0083a27c5b2358ab4dae1b2f5d8f57f10ccccc822a473326f5
43f239a70aee46347324e84e02d7651a10d08fb3dda739d22d50c53fbfa8122baacd0f9a
e5913072ef45baa1f3a4b169e141feb957e48d03f28c837d8904c3d6775308c3d3faa75d
d64adfa44e1a1141edf9349959b8f8e5291cbdc56f62b0ed6527d692e85b09a4
secret: 49fd9f53b0f93732555b2054edfdc0e3101000d75df714b98ce5aa295a37f1b1
8dfa86a1c37286d805d3ea09a20b72f93c21e83955a1f01eb7c5eead563d21e7
key: 751e346ce8f0ddb2305c8a2a85c70d5cf559c53093656be636b9406d4d7d1b70
base_nonce: 55ff7a7d739c69f44b25447b
exporter_secret: e4ff9dfbc732a2b9c75823763c5ccc954a2c0648fc6de80a5858125
2d0ee3215388a4455e69086b50b87eb28c169a52f42e71de4ca61c920e7bd24c95cc3f99
2
```

A.6.1.1. Encryptions

sequence number: 0
pt: 426561757479206973207472757468 2c20747275746820626561757479
aad: 436f756e742d30
nonce: 55ff7a7d739c69f44b25447b
ct: 170f8beddfe949b75ef9c387e201baf4132fa7374593dfafa90768788b7b2b200aaf
cc6d80ea4c795a7c5b841a

sequence number: 1
pt: 426561757479206973207472757468 2c20747275746820626561757479
aad: 436f756e742d31
nonce: 55ff7a7d739c69f44b25447a
ct: d9ee248e220ca24ac00bbbe7e221a832e4f7fa64c4fbab3945b6f3af0c5ecd5e1681
5b328be4954a05fd352256

sequence number: 2
pt: 426561757479206973207472757468 2c20747275746820626561757479
aad: 436f756e742d32
nonce: 55ff7a7d739c69f44b254479
ct: 142cf1e02d1f58d9285f2af7dcfa44f7c3f2d15c73d460c48c6e0e506a3144bae352
84e7e221105b61d24e1c7a

sequence number: 4
pt: 426561757479206973207472757468 2c20747275746820626561757479
aad: 436f756e742d34
nonce: 55ff7a7d739c69f44b25447f
ct: 3bb3a5a07100e5a12805327bf3b152df728b1c1be75a9fd2cb2bf5eac0cca1fb80ad
db37eb2a32938c7268e3e5

sequence number: 255
pt: 426561757479206973207472757468 2c20747275746820626561757479
aad: 436f756e742d323535
nonce: 55ff7a7d739c69f44b254484
ct: 4f268d0930f8d50b8fd9d0f26657ba25b5cb08b308c92e33382f369c768b558e113a
c95a4c70dd60909ad1adc7

sequence number: 256
pt: 426561757479206973207472757468 2c20747275746820626561757479
aad: 436f756e742d323536
nonce: 55ff7a7d739c69f44b25457b
ct: dbbfc44ae037864e75f136e8b4b4123351d480e6619ae0e0ae437f036f2f8f1ef677
686323977a1ccbb4b4f16a

A.6.1.2. Exported Values

```
exporter_context:
L: 32
exported_value:
05e2e5bd9f0c30832b80a279ff211cc65eceb0d97001524085d609ead60d0412

exporter_context: 00
L: 32
exported_value:
fca69744bb537f5b7a1596dbf34eaa8d84bf2e3ee7f1a155d41bd3624aa92b63

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
f389beaac6fcf6c0d9376e20f97e364f0609a88f1bc76d7328e9104df8477013
```

A.6.2. PSK Setup Information

```
mode: 1
kem_id: 18
kdf_id: 3
aead_id: 2
info: 4f6465206f6e20612047726563696616e2055726e
ikmE: f3ebfa9a69a924e672114fcd9e06fa9559e937f7eccce4181a2b506df53dbe514b
e12f094bb28e01de19dd345b4f7ede5ad7eaa6b9c3019592ec68eaae9a14732ce0
pkEm: 040085eff0835cc84351f32471d32aa453cdc1f6418eaaecf1c2824210eb1d48d0
768b368110fab21407c324b8bb4bec63f042cfa4d0868d19b760eb4beba1bff793b30036
d2c614d55730bd2a40c718f9466faf4d5f8170d22b6df98dfe0c067d02b349ae4a142e0c
03418f0a1479ff78a3db07ae2c2e89e5840f712c174ba2118e90fdcb
skEm: 012e5cfe0daf5fe2a1cd617f4c4bae7c86f1f527b3207f115e262a98cc65268ec8
8cb8645aec73b7aa0a472d0292502d1078e762646e0c093cf873243d12c39915f6
ikmR: a2a2458705e278e574f835effecd18232f8a4c459e7550a09d44348ae5d3b1ea9d
95c51995e657ad6f7cae659f5e186126a471c017f8f5e41da9eba74d4e0473e179
pkRm: 04006917e049a2be7e1482759fb067ddb94e9c4f7f5976f655088dec45246614ff
924ed3b385fc2986c0ecc39d14f907bf837d7306aada59dd5889086125ecd038ead40060
3394b5d81f89ebfd556a898cc1d6a027e143d199d3db845cb91c5289fb26c5ff80832935
b0e8dd08d37c6185a6f77683347e472d1edb6daa6bd7652fea628fae
skRm: 011bafd9c7a52e3e71afbdab0d2f31b03d998a0dc875dd7555c63560e142bde264
428de03379863b4ec6138f813fa009927dc5d15f62314c56d4e7ff2b485753eb72
psk: 0247fd33b913760fa1fa51e1892d9f307fbe65eb171e8132c2af18555a738b82
psk_id: 456e6e796e20447572696e206172616e204d6f726961
enc: 040085eff0835cc84351f32471d32aa453cdc1f6418eaaecf1c2824210eb1d48d07
68b368110fab21407c324b8bb4bec63f042cfa4d0868d19b760eb4beba1bff793b30036d
2c614d55730bd2a40c718f9466faf4d5f8170d22b6df98dfe0c067d02b349ae4a142e0c0
3418f0a1479ff78a3db07ae2c2e89e5840f712c174ba2118e90fdcb
shared_secret: 0d52de997fdaa4797720e8b1bebd3df3d03c4cf38cc8c1398168d36c3
fc7626428c9c254dd3f9274450909c64a5b3acbe45e2d850a2fd69ac0605fe5c8a057a5
key_schedule_context: 0124497637cf18d6fbcc16e9f652f00244c981726f293bb781
9861e85e50c94f0be30e022ab081e18e6f299fd3d3d976a4bc590f85bc7711bfce32ee1a
7fb1c154ef45baa1f3a4b169e141feb957e48d03f28c837d8904c3d6775308c3d3faa75d
d64adfa44e1a1141edf9349959b8f8e5291cbdc56f62b0ed6527d692e85b09a4
secret: 2cf425e26f65526afc0634a3dba4e28d980c1015130ce07c2ac7530d7a391a75
e5a0db428b09f27ad4d975b4ad1e7f85800e03ffeea35e8cf3fe67b18d4a1345
key: f764a5a4b17e5d1ffba6e699d65560497ebaea6eb0b0d9010a6d979e298a39ff
base_nonce: 479afdf3546ddba3a9841f38
exporter_secret: 5c3d4b65a13570502b93095ef196c42c8211a4a188c4590d3586366
5c705bb140ecba6ce9256be3fad35b4378d41643867454612adfd0542a684b61799bf293
f
```

A.6.2.1. Encryptions

sequence number: 0
pt: 4265617574792069732074727574682c20747275746820626561757479
aad: 436f756e742d30
nonce: 479afdf3546ddba3a9841f38
ct: de69e9d943a5d0b70be3359a19f317bd9aca4a2ebb4332a39bcdfc97d5fe62f3a777
02f4822c3be531aa7843a1

sequence number: 1
pt: 4265617574792069732074727574682c20747275746820626561757479
aad: 436f756e742d31
nonce: 479afdf3546ddba3a9841f39
ct: 77a16162831f90de350fea9152cfc685ecfa10acb4f7994f41aed43fa5431f2382d0
78ec88baec53943984553e

sequence number: 2
pt: 4265617574792069732074727574682c20747275746820626561757479
aad: 436f756e742d32
nonce: 479afdf3546ddba3a9841f3a
ct: f1d48d09f126b9003b4c7d3fe6779c7c92173188a2bb7465ba43d899a6398a333914
d2bb19fd769d53f3ec7336

sequence number: 4
pt: 4265617574792069732074727574682c20747275746820626561757479
aad: 436f756e742d34
nonce: 479afdf3546ddba3a9841f3c
ct: 829b11c082b0178082cd595be6d73742a4721b9ac05f8d2ef8a7704a53022d82bd0d
8571f578c5c13b99eccff8

sequence number: 255
pt: 4265617574792069732074727574682c20747275746820626561757479
aad: 436f756e742d323535
nonce: 479afdf3546ddba3a9841fc7
ct: a3ee291e20f37021e82df14d41f3fbe98b27c43b318a36cacd8471a3b1051ab12ee0
55b62ded95b72a63199a3f

sequence number: 256
pt: 4265617574792069732074727574682c20747275746820626561757479
aad: 436f756e742d323536
nonce: 479afdf3546ddba3a9841e38
ct: eecc2173ce1ac14b27ee67041e90ed50b7809926e55861a579949c07f6d26137bf9c
f0d097f60b5fd2fbf348ec

A.6.2.2. Exported Values

```
exporter_context:
L: 32
exported_value:
62691f0f971e34de38370bff24deb5a7d40ab628093d304be60946afcdb3a936

exporter_context: 00
L: 32
exported_value:
76083c6d1b6809da088584674327b39488eaf665f0731151128452e04ce81bff

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
0c7cfc0976e25ae7680cf909ae2de1859cd9b679610a14bec40d69b91785b2f6
```

A.6.3. Auth Setup Information

```
mode: 2
kem_id: 18
kdf_id: 3
aead_id: 2
info: 4f6465206f6e20612047726563696616e2055726e
ikmE: fe1c589c2a05893895a537f38c7cb4300b5a7e8fef3d6ccb8f07a498029c61e902
62e009dc254c7f6235f9c6b2fd6aeff0a714db131b09258c16e217b7bd2aa619b0
pkEm: 04017de12ede7f72cb101dab36a111265c97b3654816dcd6183f809d4b3d111fe7
59497f8aefdc5dbb40d3e6d21db15bdc60f15f2a420761bcaeef73b891c2b117e9cf01e2
9320b799bbc86afdc5ea97d941ea1c5bd5ebeeac7a784b3bab524746f3e640ec26ee1bd9
1255f9330d974f845084637ee0e6fe9f505c5b87c86a4e1a6c3096dd
skEm: 0185f03560de87bb2c543ef03607f3c33ac09980000de25eabe3b224312946330d
2e65d192d3b4aa46ca92fc5ca50736b624402d95f6a80dc04d1f10ae9517137261
ikmR: 8feea0438481fc0ecd470d6adfcda334a759c6b8650452c5a5dd9b2dd2cc9be33d
2bb7ee64605fc07ab4664a58bb9a8de80defe510b6c97d2daf85b92cd4bb0a66bf
pkRm: 04007d419b8834e7513d0e7cc66424a136ec5e11395ab353da324e3586673ee73d
53ab34f30a0b42a92d054d0db321b80f6217e655e304f72793767c4231785c4a4a6e008f
31b93b7a4f2b8cd12e5fe5a0523dc71353c66cbdad51c86b9e0bdfcd9a45698f2dab1809
ab1b0f88f54227232c858accc44d9a8d41775ac026341564a2d749f4
skRm: 013ef326940998544a899e15e1726548ff43bbdb23a8587aa3bef9d1b857338d87
287df5667037b519d6a14661e9503cfc95a154d93566d8c84e95ce93ad05293a0b
ikmS: 2f66a68b85ef04822b054ef521838c00c64f8b6226935593b69e13a1a2461a4f1a
74c10c836e87eed150c0db85d4e4f506cbb746149befac6f5c07dc48a615ef92db
pkSm: 04015cc3636632ea9a3879e43240beae5d15a44fba819282fac26a19c989fafdd0
f330b8521dff7dc393101b018c1e65b07be9f5fc9a28a1f450d6a541ee0d76221133001e
8f0f6a05ab79f9b9bb9ccce142a453d59c5abebb5674839d935a3ca1a3fbc328539a60b3
bc3c05fed22838584a726b9c176796cad0169ba4093332cbd2dc3a9f
skSm: 001018584599625ff9953b9305849850d5e34bd789d4b81101139662fbea8b6508
ddb9d019b0d692e737f66beae3f1f783e744202aaf6fea01506c27287e359fe776
enc: 04017de12ede7f72cb101dab36a111265c97b3654816dcd6183f809d4b3d111fe75
9497f8aefdc5dbb40d3e6d21db15bdc60f15f2a420761bcaeef73b891c2b117e9cf01e29
320b799bbc86afdc5ea97d941ea1c5bd5ebeeac7a784b3bab524746f3e640ec26ee1bd91
255f9330d974f845084637ee0e6fe9f505c5b87c86a4e1a6c3096dd
shared_secret: 26648fa2a2deb0bfc56349a590fd4cb7108a51797b634694fc02061e8
d91b3576ac736a68bf848fe2a58dfb1956d266e68209a4d631e513badf8f4dcfc00f30a
key_schedule_context: 0283a27c5b2358ab4dae1b2f5d8f57f10ccccc822a473326f5
43f239a70aee46347324e84e02d7651a10d08fb3dda739d22d50c53fbfa8122baacd0f9a
e5913072ef45baa1f3a4b169e141feb957e48d03f28c837d8904c3d6775308c3d3faa75d
d64adfa44e1a1141edf9349959b8f8e5291cbdc56f62b0ed6527d692e85b09a4
secret: 56b7acb7355d080922d2ddc227829c2276a0b456087654b3ac4b53828bd34af8
cf54626f85af858a15a86eba73011665cc922bc59fd07d2975f356d2674db554
key: 01fced239845e53f0ec616e71777883a1f9fcab22a50f701bdeee17ad040e44d
base_nonce: 9752b85fe8c73eda183f9e80
exporter_secret: 80466a9d9cc5112ddad297e817e038801e15fa18152bc4dc010a35d
7f534089c87c98b4bacd7bbc6276c4002a74085adcd9019fca6139826b5292569cfb7fe4
7
```

A.6.3.1. Encryptions

```
sequence number: 0
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d30
nonce: 9752b85fe8c73eda183f9e80
ct: 0116aeb3a1c405c61b1ce47600b7ecd11d89b9c08c408b7e2d1e00a4d64696d12e68
81dc61688209a8207427f9

sequence number: 1
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d31
nonce: 9752b85fe8c73eda183f9e81
ct: 37ece0cf6741f443e9d73b9966dc0b228499bb21fbf313948327231e70a18380e080
529c0267f399ba7c539cc6

sequence number: 2
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d32
nonce: 9752b85fe8c73eda183f9e82
ct: d17b045cac963e45d55fd3692ec17f100df66ac06d91f3b6af8efa7ed3c8895550eb
753bc801fe4bd27005b4bd

sequence number: 4
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d34
nonce: 9752b85fe8c73eda183f9e84
ct: 50c523ae7c64cada96abea16ddf67a73d2914ec86a4cedb31a7e6257f7553ed24462
6ef79a57198192b2323384

sequence number: 255
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323535
nonce: 9752b85fe8c73eda183f9e7f
ct: 53d422295a6ce8fcc51e6f69e252e7195e64abf49252f347d8c25534f1865a6a17d9
49c65ce618ddc7d816111f

sequence number: 256
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323536
nonce: 9752b85fe8c73eda183f9f80
ct: 0dfcfc22ea768880b4160fec27ab10c75fb27766c6bb97aed373a9b6eae35d31afb0
8257401075cbb602ac5abb
```

A.6.3.2. Exported Values

```
exporter_context:
L: 32
exported_value:
8d78748d632f95b8ce0c67d70f4ad1757e61e872b5941e146986804b3990154b

exporter_context: 00
L: 32
exported_value:
80a4753230900ea785b6c80775092801fe91183746479f9b04c305e1db9d1f4d

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
620b176d737cf366bcc20d96adb54ec156978220879b67923689e6dca36210ed
```

A.6.4. AuthPSK Setup Information

mode: 3
kem_id: 18
kdf_id: 3
aead_id: 2
info: 4f6465206f6e20612047726563696616e2055726e
ikmE: 54272797b1fbc128a6967ff1fd606e0c67868f7762ce1421439cbc9e90ce1b28d5
66e6c2acbce712e48eebf236696eb680849d6873e9959395b2931975d61d38bd6c
pkEm: 04000a5096a6e6e002c83517b494bfc2e36bfb8632fae8068362852b70d0ff71e5
60b15aff96741ecffb63d8ac3090c3769679009ac59a99a1feb4713c5f090fc0dbed01ad
73c45d29d369e36744e9ed37d12f80700c16d816485655169a5dd66e4ddf27f2acffe0f5
6f7f77ea2b473b4bf0518b975d9527009a3d14e5a4957e3e8a9074f8
skEm: 003430af19716084efeced1241bb1a5625b6c826f11ef31649095eb27952619e36
f62a79ea28001ac452fb20ddfbb66e62c6c0b1be03c0d28c97794a1fb638207a83
ikmR: 3db434a8bc25b27eb0c590dc64997ab1378a99f52b2cb5a5a5b2fa540888f6c0f0
9794c654f4468524e040e6b4eca2c9dcf229f908b9d318f960cc9e9baa92c5eee6
pkRm: 0401655b5d3b7cfafaba30851d25edc44c6dd17d99410efbed8591303b4dbeea8c
b1045d5255f9a60384c3bbd4a3386ae6e6fab341dc1f8db0eed5f0ab1aaac6d7838e00da
df8a1c2c64b48f89c633721e88369e54104b31368f26e35d04a442b0b428510fb23caada
686add16492f333b0f7ba74c391d779b788df2c38d7a7f4778009d91
skRm: 0053c0bc8c1db4e9e5c3e3158bfdd7fc716aef12db13c8515adf821dd692ba3ca5
3041029128ee19c8556e345c4bcb840bb7fd789f97fe10f17f0e2c6c2528072843
ikmS: 65d523d9b37e1273eb25ad0527d3a7bd33f67208dd1666d9904c6bc04969ae5831
a8b849e7ff642581f2c3e56be84609600d3c6bbdaded3f6989c37d2892b1e978d5
pkSm: 040013761e97007293d57de70962876b4926f69a52680b4714bee1d4236aa96c19
b840c57e80b14e91258f0a350e3f7ba59f3f091633aede4c7ec4fa8918323aa45d590107
6dec8eeb22899fda9ab9e1960003ff0535f53c02c40f2ae4cdc6070a3870b85b4bdd0bb7
7f1f889e7ee51f465a308f08c666ad3407f75dc046b2ff5a24dbe2ed
skSm: 003f64675fc8914ec9e2b3ecf13585b26dbaf3d5d805042ba487a5070b8c5ac1d3
9b17e2161771cc1b4d0a3ba6e866f4ea4808684b56af2a49b5e5111146d45d9326
psk: 0247fd33b913760fa1fa51e1892d9f307fbe65eb171e8132c2af18555a738b82
psk_id: 456e6e796e20447572696e206172616e204d6f726961
enc: 04000a5096a6e6e002c83517b494bfc2e36bfb8632fae8068362852b70d0ff71e56
0b15aff96741ecffb63d8ac3090c3769679009ac59a99a1feb4713c5f090fc0dbed01ad7
3c45d29d369e36744e9ed37d12f80700c16d816485655169a5dd66e4ddf27f2acffe0f56
f7f77ea2b473b4bf0518b975d9527009a3d14e5a4957e3e8a9074f8
shared_secret: 9e1d5f62cb38229f57f68948a0fbc1264499910cce50ec62cb24188c5
b0a98868f3c1cfa8c5baa97b3f24db3cdd30df6e04eae83dc4347be8a981066c3b5b945
key_schedule_context: 0324497637cf18d6fbcc16e9f652f00244c981726f293bb781
9861e85e50c94f0be30e022ab081e18e6f299fd3d3d976a4bc590f85bc7711bfce32ee1a
7fb1c154ef45baa1f3a4b169e141feb957e48d03f28c837d8904c3d6775308c3d3faa75d
d64adfa44e1a1141edf9349959b8f8e5291cbdc56f62b0ed6527d692e85b09a4
secret: 50a57775958037a04098e0054576cd3bc084d0d08d29548ba4befa5676b91eb4
dcd0752813a052c9a930d0aba6ca10b89dd690b64032dc635dece35d1bf4645c
key: 1316ed34bd52374854ed0e5cb0394ca0a79b2d8ce7f15d5104f21acdfb594286
base_nonce: d9c64ec8deb8a0647fafe8ff
exporter_secret: 6cb00ff99aebb2e4a05042ce0d048326dd2c03acd61a601b1038a65
398406a96ab8b5da3187412b2324089ea16ba4ff7e6f4fe55d281fc8ae5f2049032b69eb
d

A.6.4.1. Encryptions

```
sequence number: 0
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d30
nonce: d9c64ec8deb8a0647fafe8ff
ct: 942a2a92e0817cf032ce61abccf4f3a7c5d21b794ed943227e07b7df2d6dd92c9b8a
9371949e65cca262448ab7

sequence number: 1
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d31
nonce: d9c64ec8deb8a0647fafe8fe
ct: c0a83b5ec3d7933a090f681717290337b4fede5bfaa0a40ec29f93acad742888a151
3c649104c391c78d1d7f29

sequence number: 2
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d32
nonce: d9c64ec8deb8a0647fafe8fd
ct: 2847b2e0ce0b9da8fca7b0e81ff389d1682ee1b388ed09579b145058b5af6a93a85d
d50d9f417dc88f2c785312

sequence number: 4
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d34
nonce: d9c64ec8deb8a0647fafe8fb
ct: fbd9948ab9ac4a9cb9e295c07273600e6a111a3a89241d3e2178f39d532a2ec5c15b
9b0c6937ac84c88e0ca76f

sequence number: 255
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323535
nonce: d9c64ec8deb8a0647fafe800
ct: 63113a870131b567db8f39a11b4541eafbd2d3cf3a9bf9e5c1cfcb41e52f9027310b
82a48682159591316 94d15

sequence number: 256
pt: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323536
nonce: d9c64ec8deb8a0647fafe9ff
ct: 24f9d8dadd2107376ccd143f70f9bafcd2b21d8117d45ff327e9a78f603a32606e42
a6a8bdb57a852591d20907
```

A.6.4.2. Exported Values

exporter_context:
L: 32
exported_value:
a39502ef5ca116aa1317bd9583dd52f15b0502b71d900fc8a622d19623d0cb5d

exporter_context: 00
L: 32
exported_value:
749eda112c4cfdd6671d84595f12cd13198fc3ef93ed72369178f344fe6e09c3

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
f8b4e72cefbff4ca6c4eabb8c0383287082cfcbb953d900aed4959afd0017095

A.7. DHKEM(X25519, HKDF-SHA256), HKDF-SHA256, Export-Only AEAD

A.7.1. Base Setup Information

mode: 0
kem_id: 32
kdf_id: 1
aead_id: 65535
info: 4f6465206f6e20612047726563696616e2055726e
ikmE: 55bc245ee4efda25d38f2d54d5bb6665291b99f8108a8c4b686c2b14893ea5d9
pkEm: e5e8f9bfff6c2f29791fc351d2c25ce1299aa5eaca78a757c0b4fb4bcd830918
skEm: 095182b502f1f91f63ba584c7c3ec473d617b8b4c2cec3fad5af7fa6748165ed
ikmR: 683ae0da1d22181e74ed2e503ebf82840deb1d5e872cade20f4b458d99783e31
pkRm: 194141ca6c3c3beb4792cd97ba0ea1faff09d98435012345766ee33aae2d7664
skRm: 33d196c830a12f9ac65d6e565a590d80f04ee9b19c83c87f2c170d972a812848
enc: e5e8f9bfff6c2f29791fc351d2c25ce1299aa5eaca78a757c0b4fb4bcd830918
shared_secret:
e81716ce8f73141d4f25ee9098efc968c91e5b8ce52ffff59d64039e82918b66
key_schedule_context: 009bd09219212a8cf27c6bb5d54998c5240793a70ca0a89223
4bd5e082bc619b6a3f4c22aa6d9a0424c2b4292fdf43b8257df93c2f6adbf6ddc9c64fee
26bdd292
secret: 04d64e0620aa047e9ab833b0ebcd4ff026cefbe44338fd7d1a93548102ee01af
key:
base_nonce:
exporter_secret:
79dc8e0509cf4a3364ca027e5a0138235281611ca910e435e8ed58167c72f79b

A.7.1.1. Exported Values

```
exporter_context:
L: 32
exported_value:
7a36221bd56d50fb51ee65edfd98d06a23c4dc87085aa5866cb7087244bd2a36

exporter_context: 00
L: 32
exported_value:
d5535b87099c6c3ce80dc112a2671c6ec8e811a2f284f948cec6dd1708ee33f0

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
ffaabc85a776136ca0c378e5d084c9140ab552b78f039d2e8775f26efff4c70e
```

A.7.2. PSK Setup Information

```
mode: 1
kem_id: 32
kdf_id: 1
aead_id: 65535
info: 4f6465206f6e20612047726563696616e2055726e
ikmE: c51211a8799f6b8a0021fcba673d9c4067a98ebc6794232e5b06cb9febcbbdf5
pkEm: d3805a97cbcd5f08babd21221d3e6b362a700572d14f9bbeb94ec078d051ae3d
skEm: 1d72396121a6a826549776ef1a9d2f3a2907fc6a38902fa4e401afdb0392e627
ikmR: 5e0516b1b29c0e13386529da16525210c796f7d647c37eac118023a6aa9eb89a
pkRm: d53af36ea5f58f8868bb4a1333ed4cc47e7a63b0040eb54c77b9c8ec456da824
skRm: 98f304d4ecb312689690b113973c61ffe0aa7c13f2fbe365e48f3ed09e5a6a0c
psk: 0247fd33b913760fa1fa51e1892d9f307fbe65eb171e8132c2af18555a738b82
psk_id: 456e6e796e20447572696e206172616e204d6f726961
enc: d3805a97cbcd5f08babd21221d3e6b362a700572d14f9bbeb94ec078d051ae3d
shared_secret:
024573db58c887decb4c57b6ed39f2c9a09c85600a8a0ecb11cac24c6aaec195
key_schedule_context: 01446fb1fe2632a0a338f0a85ed1f3a0ac475bdea2cd72f8c7
13b3a46ee737379a3f4c22aa6d9a0424c2b4292fdf43b8257df93c2f6adbf6ddc9c64fee
26bdd292
secret: 638b94532e0d0bf812cf294f36b97a5bdcb0299df36e22b7bb6858e3c113080b
key:
base_nonce:
exporter_secret:
04261818aeae99d6aba5101bd35ddf3271d909a756adcef0d41389d9ed9ab153
```

A.7.2.1. Exported Values

exporter_context:
L: 32
exported_value:
be6c76955334376aa23e936be013ba8bbae90ae74ed995c1c6157e6f08dd5316

exporter_context: 00
L: 32
exported_value:
1721ed2aa852f84d44ad020c2e2be4e2e6375098bf48775a533505fd56a3f416

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
7c9d79876a288507b81a5a52365a7d39cc0fa3f07e34172984f96fec07c44cba

A.7.3. Auth Setup Information

mode: 2
kem_id: 32
kdf_id: 1
aead_id: 65535
info: 4f6465206f6e20612047726563696616e2055726e
ikmE: 43b078912a54b591a7b09b16ce89a1955a9dd60b29fb611e044260046e8b061b
pkEm: 5ac1671a55c5c3875a8afe74664aa8bc68830be9ded0c5f633cd96400e8b5c05
skEm: 83d3f217071bbf600ba6f081f6e4005d27b97c8001f55cb5ff6ea3bbea1d9295
ikmR: fc9407ae72ed614901ebf44257fb540f617284b5361cfecd620bafc4aba36f73
pkRm: ffd7ac24694cb17939d95feb7c4c6539bb31621deb9b96d715a64abdd9d14b10
skRm: ed88cda0e91ca5da64b6ad7fc34a10f096fa92f0b9ceff9d2c55124304ed8b4a
ikmS: 2ff4c37a17b2e54046a076bf5fea9c3d59250d54d0dc8572bc5f7c046307040c
pkSm: 89eb1feae431159a5250c5186f72a15962c8d0debd20a8389d8b6e4996e14306
skSm: c85f136e06d72d28314f0e34b10aadc8d297e9d71d45a5662c2b7c3b9f9f9405
enc: 5ac1671a55c5c3875a8afe74664aa8bc68830be9ded0c5f633cd96400e8b5c05
shared_secret:
e204156fd17fd65b132d53a0558cd67b7c0d7095ee494b00f47d686eb78f8fb3
key_schedule_context: 029bd09219212a8cf27c6bb5d54998c5240793a70ca0a89223
4bd5e082bc619b6a3f4c22aa6d9a0424c2b4292fdf43b8257df93c2f6adbf6ddc9c64fee
26bdd292
secret: 355e7ef17f438db43152b7fb45a0e2f49a8bf8956d5dddfec1758c0f0eb1b5d5
key:
base_nonce:
exporter_secret:
276d87e5cb0655c7d3dad95e76e6fc02746739eb9d968955ccf8a6346c97509e

A.7.3.1. Exported Values

exporter_context:
L: 32
exported_value:
83c1bac00a45ed4cb6bd8a6007d2ce4ec501f55e485c5642bd01bf6b6d7d6f0a

exporter_context: 00
L: 32
exported_value:
08a1d1ad2af3ef5bc40232a64f920650eb9b1034fac3892f729f7949621bf06e

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
ff3b0e37a9954247fea53f251b799e2edd35aac7152c5795751a3da424feca73

A.7.4. AuthPSK Setup Information

mode: 3
kem_id: 32
kdf_id: 1
aead_id: 65535
info: 4f6465206f6e2061204772656369616e2055726e
ikmE: 94efae91e96811a3a49fd1b20eb0344d68ead6ac01922c2360779aa172487f40
pkEm: 81cbf4bd7eee97dd0b600252a1c964ea186846252abb340be47087cc78f3d87c
skEm: a2b43f5c67d0d560ee04de0122c765ea5165e328410844db97f74595761bbb81
ikmR: 4dfde6fadfe5cb50fced4034e84e6d3a104aa4bf2971360032c1c0580e286663
pkRm: f47cd9d6993d2e2234eb122b425accfb486ee80f89607b087094e9f413253c2d
skRm: c4962a7f97d773a47bdf40db4b01dc6a56797c9e0deaab45f4ea3aa9b1d72904
ikmS: 26c12fef8d71d13bbbf08ce8157a283d5e67ecf0f345366b0e90341911110f1b
pkSm: 29a5bf3867a6128bbdf8e070abe7fe70ca5e07b629eba5819af73810ee20112f
skSm: 6175b2830c5743dff5b7568a7e20edb1fe477fb0487ca21d6433365be90234d0
psk: 0247fd33b913760fa1fa51e1892d9f307fbe65eb171e8132c2af18555a738b82
psk_id: 456e6e796e20447572696e206172616e204d6f726961
enc: 81cbf4bd7eee97dd0b600252a1c964ea186846252abb340be47087cc78f3d87c
shared_secret:
d69246bcd767e579b1eec80956d7e7dfbd2902dad920556f0de69bd54054a2d1
key_schedule_context: 03446fb1fe2632a0a338f0a85ed1f3a0ac475bdea2cd72f8c7
13b3a46ee737379a3f4c22aa6d9a0424c2b4292fdf43b8257df93c2f6adbf6ddc9c64fee
26bdd292
secret: c15c5bec374f2087c241d3533c6ec48e1c60a21dd00085619b2ffdd84a7918c3
key:
base_nonce:
exporter_secret:
695b1faa479c0e0518b6414c3b46e8ef5caea04c0a192246843765ae6a8a78e0

A.7.4.1. Exported Values

```
exporter_context:
L: 32
exported_value:
dafd8beb94c5802535c22ff4c1af8946c98df2c417e187c6ccafe45335810b58

exporter_context: 00
L: 32
exported_value:
7346bb0b56caf457bcc1aa63c1b97d9834644bdacac8f72dbbe3463e4e46b0dd

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
84f3466bd5a03bde6444324e63d7560e7ac790da4e5bbab01e7c4d575728c34a
```

Authors' Addresses

Richard L. Barnes
Cisco

Email: rlb@ipv.sx

Karthik Bhargavan
Inria

Email: karthikeyan.bhargavan@inria.fr

Benjamin Lipp
Inria

Email: ietf@benjaminlipp.de

Christopher A. Wood
Cloudflare

Email: caw@heapingbits.net