Workgroup: Internet Research Task Force (IRTF)
Internet-Draft: draft-irtf-cfrg-hpke-08
Published: 15 February 2021
Intended Status: Informational
Expires: 19 August 2021
A   R.L. Barnes   K. Bhargavan   B. Lipp   C.A. Wood
uCisco   Inria   Inria   Cloudflare
t
h
o
r
s
:

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 19 August 2021.

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 8.1.

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, receiver, 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 an 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 NonceOverflowError 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 NonceOverflowError 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 8.3 for a comment on the choice of a KDF for the remainder of HPKE, and Section 8.5 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 8.1 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 9 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 7.1.5 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 8.1. 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 8.1 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 8.1. 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 8.4 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 8.4 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 NonceOverflowError
  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 8.7 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 10.1.

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.

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. Future KEMs

Section 8.2 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, 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

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 values for psk, psk_id, and info 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 and equals 10, and size_input_label is the size of the label used as parameter to LabeledExtract() or LabeledExpand().

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. Security Considerations

8.1. 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 8.6.2, 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.

8.1.1. 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].

8.1.2. 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 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.

8.1.3. 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.

8.2. 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 8.1. 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.

8.2.1. Encap/Decap Interface

As mentioned in Section 8, [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].

8.2.2. 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 8.1, a definition of Insider-Auth security for KEMs used within HPKE is not useful.

8.3. 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.

8.4. 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).

8.5. 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 have different prefix-free encodings for the second parameter. This is achieved by the different prefix-free label parameters in the calls to LabeledExtract() and LabeledExpand(). This serves to separate the input domains of all Extract() and Expand() invocations. It also justifies modeling them as independent functions even if instantiated by the same KDF.

Future KEM instantiations MUST ensure that all internal invocations of Extract() and Expand() 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 an equal or similar prefixing scheme with an identifier different from "HPKE-v1". 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, even when possibly derived from the same Diffie-Hellman or KEM shared secret as in another scheme.

8.6. 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.

8.6.1. 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.

8.6.2. 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.

8.7. 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 8.1.1.

8.8. 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.

9. 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.

10. 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]

10.1. 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

10.2. 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

10.3. 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

11. 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, Steven Valdez, Riad Wahby, and other contributors in the CFRG for helpful feedback that greatly improved this document.

12. References

12.1. 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>.

12.2. 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, <http://www.ietf.org/internet-drafts/draft-ietf-mls-protocol-11.txt>.

[I-D.ietf-tls-esni] Rescorla, E., Oku, K., Sullivan, N., and C. Wood, "TLS Encrypted Client Hello", Work in Progress, Internet-Draft, draft-ietf-tls-esni-09, 16 December 2020, <http://www.ietf.org/internet-drafts/draft-ietf-tls-esni-09.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/779d0285fe0e9407abb549ba0104e831d9677164/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

These test vectors are also available in JSON format at [TestVectors]. Note that the plaintext is the same for each test

vector. Only the nonce and AAD values differ. In these vectors, GenerateKeyPair() is implemented as DeriveKeyPair(random(Nsk)).

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: 4f6465206f6e2061204772656369616e2055726e
ikmE: 1dacee520c81ade608f4fa3e5ccae0ecedcc7880e3fc6f3e5afd2e4af8396571
pkEm: 890e346283bf75af9d786a526c4a191b84d0110c794b6aa7e9a0b6205fe2c10c
skEm: ee9fcf08d07241b13b93f2cf6dbdd56f94e940d788c3e4c860f757a08974a883
ikmR: 0a3367dadc97e200074936b5adedcd5680f30672d1ec7158fdfcb795040ec909
pkRm: 8bd766c487fa9266ce3ac898827439aea2fa9c0099ab62da954b06f979f2141b
skRm: c867f27c253f720c7074f9b4a495f2c690060629e249f86991bb55edf804f7bd
enc: 890e346283bf75af9d786a526c4a191b84d0110c794b6aa7e9a0b6205fe2c10c
shared_secret:
85a44c9238b103cdaa67ec6ffde55d8f2e75e49aefcf1ade3c65900bddd503f2
key_schedule_context: 00725611c9d98c07c03f60095cd32d400d8347d45ed67097bb
ad50fc56da742d07cb6cffde367bb0565ba28bb02c90744a20f5ef37f30523526106f637
abb05449
secret: aa2c8768a36ce56c54a50a4ef93bdf42c225fa5cdf68a1f65c76b30358cdc478
key: 96d0b503c045e18f6e9f62a52d7f59d2
base_nonce: aa39425b7270fcaf1c7b69ec
exporter_secret:
304296751e7583846d4ec1d49f78b511dee838a32e18dd1bfa44a30a1c1012e0
```

A.1.1.1. Encryptions

```
sequence number: 0
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d30
nonce: aa39425b7270fcaf1c7b69ec
ciphertext: 1d2ae93bff2fc322a909669c94372cdd2ac0da261face2a706e417a95227
2f6e5eaa20d0cd15fc28ee52026c4d

sequence number: 1
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d31
nonce: aa39425b7270fcaf1c7b69ed
ciphertext: db308d0077b75c29fd4ebbf3e3ee57312af210d2d2a795e882e8da0e5ae5
a0775684fc8530aa0c31aea69755b7

sequence number: 2
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d32
nonce: aa39425b7270fcaf1c7b69ee
ciphertext: ae1262b27b76a174a67143392dd384535bb8cd3d3a16ff971baeb81b2784
7238458e257c024f4fe52e1c2d2512

sequence number: 4
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d34
nonce: aa39425b7270fcaf1c7b69e8
ciphertext: 90abc5e812ab0a5952f2222c12753821ab91e5dbabbf041e7fd21fdb1304
5648e90ddd152a183dc2881bd67528

sequence number: 255
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d323535
nonce: aa39425b7270fcaf1c7b6913
ciphertext: a356913480b5d3017d05deda7dae5a399ad14e54dc44a2452c9d909e48b1
383a55fd9ba7a22ceaeb6c27e32540

sequence number: 256
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d323536
nonce: aa39425b7270fcaf1c7b68ec
ciphertext: 22cfb308437ae3abcac038c030b60f825a35d85a3b668253c43811973ace
5c60dee014d97bd13c67e8c4eaaf36
```

A.1.1.2. Exported Values

```
exporter_context:
L: 32
exported_value:
d703147bd36b0218fed1af62840ef3a15869d1c64bc68b4df87371ffc9f9ad95

exporter_context: 00
L: 32
exported_value:
8bfcbf37919c5ee14028640b7eace4e6de00fc39acf073e74cbd9712c9da7beb

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
a71f58a7f54e8ef1ed2a6f70f7a0f158246d4c569750420d545f05822d10fa07
```

A.1.2. PSK Setup Information

```
mode: 1
kem_id: 32
kdf_id: 1
aead_id: 1
info: 4f6465206f6e20612047726563696616e2055726e
ikmE: 7f5b697a87fb11d733e31265410c80b72525637e1b872f9ede00d507c0c534d7
pkEm: 6c869089a41d49afebbef4a046671062cb95f334d333b2796f78b6c56306bf53
skEm: a1fb4d2bda0df27dd5cf33fd6d67d4b2fcf7b2d3ef89ba95ded5bc513cb529c3
ikmR: 89a6ce4b7b8eb12eebff3864615a2539e3e64fa2d4ab525c18c7bc451a2b5e40
pkRm: 2b15f3560e8545473330de96ab3f0df764571141a4ae9d02d32f967b38b0c701
skRm: 52a1b190b90aa604eabdb03853dea870a88c2ab78f812f0137af75c11f00451f
psk: 0247fd33b913760fa1fa51e1892d9f307fbe65eb171e8132c2af18555a738b82
psk_id: 456e6e796e20447572696e206172616e204d6f726961
enc: 6c869089a41d49afebbef4a046671062cb95f334d333b2796f78b6c56306bf53
shared_secret:
a0028c3a2e4542ec179cc4f706d49911305e7634b9a952fefb58a8e709d5586d
key_schedule_context: 01e78d5cf6190d275863411ff5edd0dece5d39fa48e04eec1e
d9b71be34729d18ccb6cffde367bb0565ba28bb02c90744a20f5ef37f30523526106f637
abb05449
secret: 42d002638b73763cd5194a327f1f9c0ac6a3561c051db3206b02a37da7359a07
key: c7295a3618b0d5f60513c1e0c3624b60
base_nonce: 75ac8b35f8d5f59924145c97
exporter_secret:
c7184d43f15a77671045a0f2162963f62c47ab3a933c6861e038a275d7138489
```

A.1.2.1. Encryptions

```
sequence number: 0
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d30
nonce: 75ac8b35f8d5f59924145c97
ciphertext: 0024748142b413ee22311a16a7b1bf813cee46b8aad06da9eb1ae14156c3
d31bd84385f939e4f6554be9fb22e5

sequence number: 1
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d31
nonce: 75ac8b35f8d5f59924145c96
ciphertext: c901001814df06c9209bb849511875b2c1a531775304417bfe460932de21
a4cc77d234a5e4d9144cf092eecc50

sequence number: 2
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d32
nonce: 75ac8b35f8d5f59924145c95
ciphertext: 7954e8125a7c44d2ee29682541b13139563b220c33f81bc38d18b06bd1f2
792f087d64c2de1df6a582a4514984

sequence number: 4
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d34
nonce: 75ac8b35f8d5f59924145c93
ciphertext: b6f292027b94a950cb081fb3e6cd0f3f62ff31934b84b138cc0502550324
f1edff3fe7d46891fde2b13e3f487b

sequence number: 255
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d323535
nonce: 75ac8b35f8d5f59924145c68
ciphertext: bb5bed92d706f18ecad79bae284255719ca717824b91060d0841d088ecaf
1c23ba87a80920c2018dd0485748a2

sequence number: 256
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d323536
nonce: 75ac8b35f8d5f59924145d97
ciphertext: da17b7f8ab265f65eba88ed4d8a7c13a7f14cff2fe8703207109db0a0a4e
4f9e1b611b794ca0951f1e551eb1f3
```

A.1.2.2. Exported Values

```
exporter_context:
L: 32
exported_value:
02e3fec06eb0aa470b793e040746e459c07ca1fdb12fec9c15eb25f9fc40d6ee

exporter_context: 00
L: 32
exported_value:
fa704fa53292124bf443004b0c29573618be834d515f433fed66675250379c5d

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
039bced37cd97c3702e685150baa1c62c003ef3cb3e69cb827d410a44eb1be0b
```

A.1.3. Auth Setup Information

```
mode: 2
kem_id: 32
kdf_id: 1
aead_id: 1
info: 4f6465206f6e20612047726563696616e2055726e
ikmE: f7faa81395a22af1071b5273f9f94db0609ab9a7f4eca54834ea408d97294cc2
pkEm: 5b1f385b0e4063b06d9c20ed518a5002e7a46d30c8267e222406fab1a9b40561
skEm: fd0bba2abfa07ad77664ae76107020832064db688bbf56aa30b0eb64ebd91870
ikmR: a10d107753f85bf7128768693e53129fb44a281ea4849a86231632db48dbb28c
pkRm: 9ea1cb679d2b306cfdae2360d8e67cf2fd4c1b594d68894508b7e4edc2e74f5a
skRm: 0dd0a94308e5f9fdf00939ab8e97e5e0cdbda0f91475762b75ffda681990c77b
ikmS: e1257ce84dd98bca7847d75661b658cd2cc6c4efbf9e989a40f390720b57d409
pkSm: 2cdac014f0d5a65614fc6669e8f55f16d8dba6a92b474640640103b5c26c3a2a
skSm: c3ce378b34fb922adfbaaeef93244edf2107d114b3d6b5e8372482f280ab5a37
enc: 5b1f385b0e4063b06d9c20ed518a5002e7a46d30c8267e222406fab1a9b40561
shared_secret:
53e8d00d80ca3c758365537bbf79f97458452e634b43d6e3616f00f2b6aedd17
key_schedule_context: 02725611c9d98c07c03f60095cd32d400d8347d45ed67097bb
ad50fc56da742d07cb6cffde367bb0565ba28bb02c90744a20f5ef37f30523526106f637
abb05449
secret: 7ef00a01dd67a5576581f0002da22db7fe9c48e5ea4e18a153e83c4ace19c475
key: 3e2e73e913faccf13c182d9c3d162c03
base_nonce: f6fe99886f9e8f9e991e3deb
exporter_secret:
f24ea8a3e650c5f0b9e7c9aa79bde838b41acb1e74a32b105a46c0f9168afac5
```

A.1.3.1. Encryptions

```
sequence number: 0
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d30
nonce: f6fe99886f9e8f9e991e3deb
ciphertext: f3884f172c31e40cd4b90ca2280e87c49ae0b6a83ceab51ec79905349361
2adbc03ae0e39e656878c3148199f5

sequence number: 1
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d31
nonce: f6fe99886f9e8f9e991e3dea
ciphertext: 7bf476821bd718c3fc2982ebd86f81b46a82dba5969fa0ae2aca7af37d83
8fb9473b5bccdbe975457f88e41687

sequence number: 2
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d32
nonce: f6fe99886f9e8f9e991e3de9
ciphertext: ce38a7151f122f2b0575e2a8ed72e2dc5653ba7753438b7c8b980dec44e3
3e5f0e5b1b29852cb410723e13e1d2

sequence number: 4
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d34
nonce: f6fe99886f9e8f9e991e3def
ciphertext: f9d6702904383307af8b888e33b1cfea49d31791358df32be3d9b8d2a25f
6d0900f7b84bec17eb2f7915b93742

sequence number: 255
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323535
nonce: f6fe99886f9e8f9e991e3d14
ciphertext: 15634a665c3a82b89b2b487ac00467d78490463a95f653aa87acef355065
fb4590cdacf591c49496cfd5524a86

sequence number: 256
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323536
nonce: f6fe99886f9e8f9e991e3ceb
ciphertext: 87c1df675abb8cddff53be2a8ddb288c2addadc6e7394b37dc54644b9193
1e1bdd025cc4ca3493a88387f9004f
```

A.1.3.2. Exported Values

```
exporter_context:
L: 32
exported_value:
1f0317bb51c2f650bdfd3a1abd221315149522396df345888946d48f7dcc752d

exporter_context: 00
L: 32
exported_value:
594e2a101ccb8e9d7bc09bb5c284cc86156fc4a6484c8341a69e52e7e7e20061

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
f796b4b0581d12ffe09b34867cc38a787bf7c2cd578f72f45c90d82bc538bacf
```

A.1.4. AuthPSK Setup Information

```
mode: 3
kem_id: 32
kdf_id: 1
aead_id: 1
info: 4f6465206f6e20612047726563696616e2055726e
ikmE: e98713f7e24e7025fb7abddfff317d865ad5f4c3b6be22cda48dbcdde6151a68
pkEm: 9dbb32f56ac1c7e70a13e63ccf63239797cfa80ee86d6eab0a70fbb1b2023f0f
skEm: ef9c696d945edba5e29478de9712423a5dc27d6b7a0b23ddb589e20ffcabc5fc
ikmR: b281654cdb2cd6a9670c3035eaa702736a951585a938381198f7d29085ca7f3a
pkRm: f89ba4eeb8c0e7efa3606872d863a53aaf38fe9122e00b956e9cdd973d8ce46b
skRm: ab88c57171ce7497c26ef70aafbcd902497fe0caf595182d7d3c8770d3642a2f
ikmS: ea95c4d3c5e7868af29fc0cb0841485dead8d54b57ee1b8ed31a153295454835
pkSm: 7bbd011d8cd7724d81b09a65ef49f1faa33890e79086d877fbd0c03e4ff60826
skSm: 3d8ab8e757693d972c1e205c2af3cc01b03e59d9b17fde438c8611a874b0be35
psk: 0247fd33b913760fa1fa51e1892d9f307fbe65eb171e8132c2af18555a738b82
psk_id: 456e6e796e20447572696e206172616e204d6f726961
enc: 9dbb32f56ac1c7e70a13e63ccf63239797cfa80ee86d6eab0a70fbb1b2023f0f
shared_secret:
7d56765a93434310b9571be3bea9919213601f2b7398ee50ff8ba0de79f85986
key_schedule_context: 03e78d5cf6190d275863411ff5edd0dece5d39fa48e04eec1e
d9b71be34729d18ccb6cffde367bb0565ba28bb02c90744a20f5ef37f30523526106f637
abb05449
secret: 5507a37af3ab150446e76f10b10b77082576aca1866c616c328d6d76c3f9fb3e
key: 99cc5cc5d06d85b67432c4fbb5ccb257
base_nonce: 6eac26e93ca9a6772bc5990d
exporter_secret:
7e210913716d706c05d1a5e35f2af8483d5f719d4af92ee768acd943851e02bf
```

A.1.4.1. Encryptions

```
sequence number: 0
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d30
nonce: 6eac26e93ca9a6772bc5990d
ciphertext: 1db161fce3ddf79245da0a725439cf8793de594fbcc8c4c6c1d140ab4cff
3b3881754da1481ca8e152860271c7

sequence number: 1
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d31
nonce: 6eac26e93ca9a6772bc5990c
ciphertext: 9412a31be293a855abdf95ec1bc609dbe594033f7d101ecf2fbb96c825e6
99405ac0b2543dcc371e4f51f88645

sequence number: 2
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d32
nonce: 6eac26e93ca9a6772bc5990f
ciphertext: bbb2f82d70abe28dd624c12ddc637f6f10b86bbaa65c685678e49b5dfc5e
db1ac2eead5e7a616f37e95a675a03

sequence number: 4
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d34
nonce: 6eac26e93ca9a6772bc59909
ciphertext: 050bc824cbc3ba509b3f9fd50aa631523339d2298aacf59c2702f7aff2dc
a794de6ce53d6acd0fb48f2659c451

sequence number: 255
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d323535
nonce: 6eac26e93ca9a6772bc599f2
ciphertext: 352276dc7ad4fcae86aed2c55a543519701d85c11c49f053f35b68b84c9e
37534a030740b976ed621861f0dd2d

sequence number: 256
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d323536
nonce: 6eac26e93ca9a6772bc5980d
ciphertext: 2914fcb5cb68397ed18cd139a2f65eb3c4695760e22a54b8fee2e7cdf625
05f19171c00e48ed8e1b46a05d6b61
```

A.1.4.2. Exported Values

```
exporter_context:
L: 32
exported_value:
e14cb7319666c34d6bacd5b7f2f7d45b877f77d27f74279f728b7442fe939257

exporter_context: 00
L: 32
exported_value:
3f1f049e80d67fa30235490ea2f1f384992a3eeccbbda9393290805f3a791b98

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
197afaa956bb7d00cb2a7c5177ddcd0a8f61ab0bf772f459c7338eba49774bed
```

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: 4f6465206f6e20612047726563369616e2055726e
ikmE: 8284a224fe2689b97c5fb598889d5af5268ef22efc656ac8359a1c6007910f30
pkEm: d862803cbbc56f94f7ffd62bdb7f96954e4fa1e2b2c3a8e4251858ab57b79e0b
skEm: 4f15c72b234d390ced29f2d6b07c2930254bb2101425a1cbe709e23375c2ff05
ikmR: be6e6cf70ed8d40b199fccc9d824ba84a02f0dccc409de3643ffd68962a92ef3
pkRm: e77bad5e13ac74dc341385a0454a0ff48cbc1faed1f56656b6f5ea9ad7d1220e
skRm: 47be98cd1ac849d09e95fb64dbebb4861457a864f98becbe5c399d636025a7d7
enc: d862803cbbc56f94f7ffd62bdb7f96954e4fa1e2b2c3a8e4251858ab57b79e0b
shared_secret:
554871607763734d0809e4e8776af6086bee6efcbe98d054e5a1d8f86edeab82
key_schedule_context: 00431df6cd95e11ff49d7013563baf7f11588c75a6611ee2a4
404a49306ae4cfc5b69c5718a60cc5876c358d3f7fc31ddb598503f67be58ea1e798c0bb
19eb9796
secret: 6797a9ad52fa35dfb5bfa3c597dcaf2bfd395fc6bdd34dc5c4620c6ce6d960f1
key: 84473361e8d74ac69b220fc02f66f4c5d54c4d32ebf0f5b73dda23a7fead9930
base_nonce: f7fb1c2c9a13c5fd8e5c7ab6
exporter_secret:
7f843b1431520f8c2a5e329e75496e96be470b86d25e85ffa20113765f68166e
```

A.2.1.1. Encryptions

```
sequence number: 0
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d30
nonce: f7fb1c2c9a13c5fd8e5c7ab6
ciphertext: 11be91e6a7f80d2f341e3baa06470aa60401dc953d6933234c81bafffc44
76cecff9b427359e00084c5e02d13a

sequence number: 1
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d31
nonce: f7fb1c2c9a13c5fd8e5c7ab7
ciphertext: 752f4c9d907602a3fefcf9e0b8defad1c87082762b533c83da780b3c78ba
b6ede23fff3de660a2e5c4e4ed406e

sequence number: 2
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d32
nonce: f7fb1c2c9a13c5fd8e5c7ab4
ciphertext: 025350759f0ce9e3dbc8358473163894a5da3d785eec934e6801a4ff072d
99024fc81f60b6d6a839a0f2c8fab4

sequence number: 4
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d34
nonce: f7fb1c2c9a13c5fd8e5c7ab2
ciphertext: 28ce79827dbaf7c07b8270170e6851c1e9bb3a98d8d0538ff551c25b4d1d
1e176982d4377cc021510e609ab66b

sequence number: 255
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323535
nonce: f7fb1c2c9a13c5fd8e5c7a49
ciphertext: 0c3d124ab6785da819846e32fc27b7554d1fea1d26ce2dc26e1a693dd7e0
16d51495d08817797258d3cba5e9cd

sequence number: 256
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323536
nonce: f7fb1c2c9a13c5fd8e5c7bb6
ciphertext: 4333ea695ea28b188f44060bbe8358a0c00b953a16911cf430843511eeea
fbcac7537a6405be9b8b40e76fba65
```

A.2.1.2. Exported Values

```
exporter_context:
L: 32
exported_value:
f22324dcf072e206156be29e76820fefac943468e1e1a511d99b967cf1994b01

exporter_context: 00
L: 32
exported_value:
785f3dd5cb3a99bbc76502a0f36856e1f5e88afe22853154333f6bf28672f9b8

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
30d788b5e977debf0b8b51c34bd9514116ac7e1b494f6efd44080c02add02809
```

A.2.2. PSK Setup Information

```
mode: 1
kem_id: 32
kdf_id: 1
aead_id: 3
info: 4f6465206f6e20612047726563696616e2055726e
ikmE: a898eec077b574f86a1bb52899ac760912b39e552076d41067e5f30650da0cbf
pkEm: d9ded3e1e50f70c474b4ebb64b4b9c3d711b5000918c88a1b01ea0bfc611ef25
skEm: 58378b622e94053c3c0e3f4b416365ccb7bfe06b144b599ce23386d13bad3168
ikmR: 8eef790fad795e48041b5abc031e785418cc736a0f2ffe49744a1257e3ee3c11
pkRm: 74a556a4fda89ff0db891cb66775d6c9d9b4e3e23bd9714db2124c5d23f0b155
skRm: 25fa61380093b84d96b13d6e2d6b5d0dd9d182bc0b54c8770581287014370052
psk: 0247fd33b913760fa1fa51e1892d9f307fbe65eb171e8132c2af18555a738b82
psk_id: 456e6e796e20447572696e206172616e204d6f726961
enc: d9ded3e1e50f70c474b4ebb64b4b9c3d711b5000918c88a1b01ea0bfc611ef25
shared_secret:
907709a95470c19f1338ad5f6483e3bc5d6f77a4efb94bf1965e4cb0d26652c8
key_schedule_context: 016870c4c76ca38ae43efbec0f2377d109499d7ce73f4a9e1e
c37f21d3d063b97cb69c5718a60cc5876c358d3f7fc31ddb598503f67be58ea1e798c0bb
19eb9796
secret: fd51da955e44c844126ed9e67aabed4c03530e702c94d9bef0236a0832415326
key: 6db9b0f5e6b99a4f3b2379d0aa26b7557b2d60f6565382222e34fbd807b882a2
base_nonce: 811bc3b560eddafde5ad6ed4
exporter_secret:
afa67deee5056ed230df6dabb629bf5c9535b1bbf4eaff3b9532f00b2c89a15f
```

A.2.2.1. Encryptions

```
sequence number: 0
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d30
nonce: 811bc3b560eddafde5ad6ed4
ciphertext: baeb454095d2218f4cfdc59df314ac4df92edd8e66b16a4d5913bfdf3eea
f305380c0368bf719ef31745f5b84e

sequence number: 1
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d31
nonce: 811bc3b560eddafde5ad6ed5
ciphertext: f798b121acecfe88fd3ed454e70b0a6386ebc735e9cc702d1e9367278278
c259dbaab86ec70083c08806f42cb7

sequence number: 2
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d32
nonce: 811bc3b560eddafde5ad6ed6
ciphertext: 4543b32f9d2ecb0e983551cf3ef53c3f93c779553f1451ef09731edfb466
f7046a189e585177520ca488c86a28

sequence number: 4
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d34
nonce: 811bc3b560eddafde5ad6ed0
ciphertext: 0af1c28bb8d92d7b520db655518c74a3b32cd86aeb484274ddbf82c70134
604a9da5b6e52352a498ec1cae961d

sequence number: 255
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323535
nonce: 811bc3b560eddafde5ad6e2b
ciphertext: 11e42eea033e9c9ea4b94a6b0c4f210bab002b101e4b06a4544477cb8aad
98ac74132d521454f5676456203527

sequence number: 256
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323536
nonce: 811bc3b560eddafde5ad6fd4
ciphertext: 267fce55d3263f581ea42a2cf528b0d67bf6d1dc9220718fb5ed19f1a38e
0c5bdaf6dde2805a915ec039d44006
```

A.2.2.2. Exported Values

```
exporter_context:
L: 32
exported_value:
03983379c266a6b09287be5743290ad19b8773fa87693091cd72a6aa215c2e93

exporter_context: 00
L: 32
exported_value:
5110008bcefc255f1d0feef8fcbcbe0665c42a30355d7bf430fb3ee02a5507b8

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
b6da48879ce5a1e1cbe3338800b061a46d1d87ef526a6fa44a159836f3f148e2
```

A.2.3. Auth Setup Information

```
mode: 2
kem_id: 32
kdf_id: 1
aead_id: 3
info: 4f6465206f6e20612047726563696616e2055726e
ikmE: 1de23785c74ab0765d900a39cfa60ce81d2484b23224255d003ab9009951c8b4
pkEm: f2fe00d5fce2307765db727702cfe2f9278594d1186425afb7868be2d1087639
skEm: 9f9c631189d1f386cbdca0331e5ee38b3d1e6b98842a5fa29de7a17bb5320488
ikmR: f6ed2384937da2327cedac13c34e939606278609269 93a60d031fa6f46fb659a
pkRm: d16548e3896b89d61746ef931e343ee60c90a6f84b02de99e665c2c7f11cf463
skRm: 5066f0736ff5f36f6901c2a142f0ee8a49a3bc73170078cd2797cda565fa3e32
ikmS: 02ac3d858a4b089c86f5b6fe5513e7e49c8e0603b008ae4a4019e308bb0ac484
pkSm: 5c7091fa039a955d152e60a3d8dccd8e4c0d8aff2082163c05a0e00590c12d79
skSm: 57078781148b731cf3eff2f7c4a15aedeee5d5f743d5c2a9af67f8bab52aeb4e
enc: f2fe00d5fce2307765db727702cfe2f9278594d1186425afb7868be2d1087639
shared_secret:
9590bceff3d360d7ca200d0daddb81817889079dd110af273838a0aa226e6354
key_schedule_context: 02431df6cd95e11ff49d7013563baf7f11588c75a6611ee2a4
404a49306ae4cfc5b69c5718a60cc5876c358d3f7fc31ddb598503f67be58ea1e798c0bb
19eb9796
secret: 2496f7812d4f719ed83d17dac1647e5d1dd8e42dd815af8d262389052e4799a0
key: 8c46e9b58caea9becf0a0d3f694d99dcdb58f957d8cb07be09fad142ab2cef0d
base_nonce: 161f276aa6d7bd3029764151
exporter_secret:
5505620ecc591c690b3c397b1f784d6a30048b5324fbe1dd60d79b4508b3944a
```

A.2.3.1. Encryptions

```
sequence number: 0
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d30
nonce: 161f276aa6d7bd3029764151
ciphertext: 7cfb5a4f4cafced25dd42143a5dfb9aa9121ed9695cb70a14aeef805fa70
bcc73724c36dac1377251e8fc1d0c6

sequence number: 1
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d31
nonce: 161f276aa6d7bd3029764150
ciphertext: 7a18b8f1a7b470b518a548e359f307277f8c2560f56b6955a38f3d66f423
4b5c4b719b5c0445ae2407f5bc2e41

sequence number: 2
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d32
nonce: 161f276aa6d7bd3029764153
ciphertext: 149dd1361a871703ed5edc5ea1a74e6973ed4c56cb67982ae6445e3497f2
c6b3f99d3acac56145f4e2e2150bf6

sequence number: 4
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d34
nonce: 161f276aa6d7bd3029764155
ciphertext: dcaa6179a1d1bc2a810be84afdaa4ff73c1b04e3bd1a6d797030853b312c
2355f28b85b5e88b1ae764bde61ec7

sequence number: 255
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323535
nonce: 161f276aa6d7bd30297641ae
ciphertext: 2b4fe7ef3b3b7ba1b89a2792cecb4bcf41e4756e4dbb1f3eae7e17dcf2c5
b3895908f049af597f0132ed68970e

sequence number: 256
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323536
nonce: 161f276aa6d7bd3029764051
ciphertext: 5dbec668404f0ef7940afc990d8d18ecf310ecbd62192589be7c7120a0a5
de49699084834b15b5a2c4d38f6c9a
```

A.2.3.2. Exported Values

```
exporter_context:
L: 32
exported_value:
0f86d39794a3d37b391e6f813da7df9cea25addb572ab981b56141ae726f89fa

exporter_context: 00
L: 32
exported_value:
4313a336bc617f42dffed04c039724594bdc556b4bf71192b88ad91310afc601

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
107bb4fabef1741a4ae6a0beed5930702e37935de814f328c200780662bf3d8d
```

A.2.4. AuthPSK Setup Information

```
mode: 3
kem_id: 32
kdf_id: 1
aead_id: 3
info: 4f6465206f6e20612047726563696616e2055726e
ikmE: 545fc015fbbe1e6c2aac275e31fc7f22c89d9c9e3c5675c54e1446b49c93c867
pkEm: 15280746df632034bb86dde20324ab57fc772d07da1789673921145c8ea74a78
skEm: ad11a49efa147edc038fddafe5e6df7f8b621ad4bca89730154e644db05eeebe
ikmR: c7e6f05f146f1d4501f6b30932de38c12b4a70918d5108d1d1d557bfd4c31861
pkRm: e5d43ed47df8bad12d58de7b28c9eda2086f6e63afb7af872fb789384844643c
skRm: 7e2f75cd08060313583bfef97ee765b7e1298263b7c5459561ea576c1085b77b
ikmS: 53116291ecfc12dc1628e89828b265507d610190c255c095681e1a14dcbc9685
pkSm: 9478992ba096dfcb7f62540293802901e158db270c06b7c1ff8e51d634ab965e
skSm: 735a72d574724240f4e942e1b0b24cce6f69ed214429e4f9b9afc0a45fdf889f
psk: 0247fd33b913760fa1fa51e1892d9f307fbe65eb171e8132c2af18555a738b82
psk_id: 456e6e796e20447572696e206172616e204d6f726961
enc: 15280746df632034bb86dde20324ab57fc772d07da1789673921145c8ea74a78
shared_secret:
396d5e9b5d295dc55502082de7f6f9fc1391c4675f690f030862b21882bfc59c
key_schedule_context: 036870c4c76ca38ae43efbec0f2377d109499d7ce73f4a9e1e
c37f21d3d063b97cb69c5718a60cc5876c358d3f7fc31ddb598503f67be58ea1e798c0bb
19eb9796
secret: b80690dc0a11402ec637465d8636d75244d11e7d29d3e0d80999a46243d186d1
key: 5f763de36998ad55feb925dbc2509a8d14cee80b59644b85c526f745902f1946
base_nonce: ac3b6f34d0d5b20a32547494
exporter_secret:
096b72330ea640ba0bbd8a3ca4b75cab2a08c86d05f9f8653ca017923e39ecea
```

A.2.4.1. Encryptions

```
sequence number: 0
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d30
nonce: ac3b6f34d0d5b20a32547494
ciphertext: cc9c605ee5b13d090f25b67edf3f7abb48b1dfc6f572f712e48bd67f2a1c
20aeb431d65f02dbf7aaaae94cd268

sequence number: 1
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d31
nonce: ac3b6f34d0d5b20a32547495
ciphertext: 8b65253309a197cccb939ab2647e99e71978371eda75ea2c01b219abebc9
b2b59b57b9d3dd25e9d16aaec00d75

sequence number: 2
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d32
nonce: ac3b6f34d0d5b20a32547496
ciphertext: 8eb4abaa0d8b23fcb8178ee69b76e8d7147d15d34e5c6b9488bd1f9b6e40
ec8124f10879ce6951b37d4468b63a

sequence number: 4
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d34
nonce: ac3b6f34d0d5b20a32547490
ciphertext: 87bf28e98f48b097ddedc8633af82f954f48cdf2cbc91bc6422f7ef5749a
6e9e79d9713d1667b5d926686d3be9

sequence number: 255
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323535
nonce: ac3b6f34d0d5b20a3254746b
ciphertext: 6d22dda2882fa98abe99fca0ce66d6058b37064f2aeb883cdbea539f5a8c
89e26a1fe3db2587dd79b20349896b

sequence number: 256
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323536
nonce: ac3b6f34d0d5b20a32547594
ciphertext: c9ba6a14b5177cd7be0559892259c5f7a7ce5b1084d85cdeb61940300d4a
3d386113130b186d15193e2ee22b03
```

A.2.4.2. Exported Values

```
exporter_context:
L: 32
exported_value:
ecd4e3b963014dc620459d9981da9a9c9dcd445ac941cf37cbd4dbec66d3c4ee

exporter_context: 00
L: 32
exported_value:
b23a4a0119f979bd5011b8207f1d5eae71cb8d35560a3b5ad498c64e81cd3d92

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
97c2faad2e5fba0efcae91e6b7fcb8b59126f0c3ed2fe559cb4537d9bf0f0f4c
```

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: 4f6465206f6e2061204772656369616e2055726e
ikmE: 5377490d651f4cd3e97ddaaeb50f7337230618522c4e54c1d63587adf8c96cc7
pkEm: 046c62e9ee75fe5b73c4aed592220c08b100a8dd0bc8ed09bfe3ccdcc2fcb12c84
fc09748089abca1a2310ceebbbf3cc14e56bd325f74ba2dc8242b789f503f400
skEm: 44d89b96fa66ec92dff344aa2df82e99d438da3c602ee705fffc1c7d5949181b
ikmR: 6b1ec8ebf259e05ca9596fd0ec634035a649d81582b0e3007f8603c6eb3435ad
pkRm: 04d4ead4935ada233d184e73611b575e74983c10b82cb16f1fc12904792673fc31
cc99035421947969b1785cc169b5f18abea0f18413dceb895b47b6f8b457470a
skRm: fcabb035645cbd80f4e63e8a339c34aae82c7c79c51ddbe04890f277a6a8a249
enc: 046c62e9ee75fe5b73c4aed592220c08b100a8dd0bc8ed09bfe3ccdcc2fcb12c84f
c09748089abca1a2310ceebbbf3cc14e56bd325f74ba2dc8242b789f503f400
shared_secret:
4c43dd81351c0d19bf5eff313012c080978aaa3b8d14aff42322f1b832cf2610
key_schedule_context: 00b88d4e6d91759e65e87c470e8b9141113e9ad5f0c8ceefc1
e088c82e6980500798e486f9c9c09c9b5c753ac72d6005de254c607d1b534ed11d493ae1
c1d9ac85
secret: fad3e982298be3f2b529b8ac142213cef6b48ad4c10d860d561cc8f1b6c5dfec
key: 856ed4d1d5ebfdbb25fd2f3d4bca3f72
base_nonce: 7f16c754a173fcd13d14f878
exporter_secret:
56f0f7619fa9a896d18da2f921597cd299c57985ca0c3c1cd473aa1c88d18377
```

A.3.1.1. Encryptions

```
sequence number: 0
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d30
nonce: 7f16c754a173fcd13d14f878
ciphertext: b7481ce0b49e40d4a71a73b60beda9c5ea5656815608b96eb65ad9932511
de4c2354e2444d310db8b9593ffb2c

sequence number: 1
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d31
nonce: 7f16c754a173fcd13d14f879
ciphertext: 28e5588d7dc7e3a1d602ba5629709d0e27bec782439afe4056105f7ecee4
2ac7a61740faef8fefa9e7e6dfb520

sequence number: 2
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d32
nonce: 7f16c754a173fcd13d14f87a
ciphertext: 8db0f0f743112e74777ff9290b9a3a2ebf34872a905c514cfd0b3ffc1495
c6d5a08c6233edbaee198a3321fe0b

sequence number: 4
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d34
nonce: 7f16c754a173fcd13d14f87c
ciphertext: f7c7ebcd92829dd869d4feed2c794f5a35dc23cda6420f4b601d705f7266
9b3d1bf470b2f1410ca0f939ef99ca

sequence number: 255
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323535
nonce: 7f16c754a173fcd13d14f887
ciphertext: 1b720d012a81f2d15443587f2e1c805ef65b60a09dc5776825175aa325e3
6d38c13b6e5c0c9c24cec3845f64a5

sequence number: 256
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323536
nonce: 7f16c754a173fcd13d14f978
ciphertext: b3f206da3ee89d41bcf154666c379267931000f50d172cb5a6b43b0b9021
910669403daf4f63f4d9893a06ad01
```

A.3.1.2. Exported Values

```
exporter_context:
L: 32
exported_value:
40b45cc28dbb2db11cf9e2ebccecc3b1f84cd7cf0818de77fcbe8c930b302411

exporter_context: 00
L: 32
exported_value:
1879b1b4113fdfc0211b1d03d2cd320987e887dc6815436289083e57e63dc098

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
02438d80efe54328b1a2db44026ebc1869bfb190d8164f91d7457623065666cf
```

A.3.2. PSK Setup Information

```
mode: 1
kem_id: 16
kdf_id: 1
aead_id: 1
info: 4f6465206f6e20612047726563696616e2055726e
ikmE: 6470e9c5f96201e747fac5e53e89a3ec71d7dbf6b4ce89ab30cb60c1a2c1376b
pkEm: 044580890bca9e5e8cce209f3c9c1731373012c18c4224b99f21bd844789ef9646
d97a963d9a2f78ce6d7bfa5a8bf01f48f4b4df184f527b072782f46510c5e9e4
skEm: 9dba601a744e55b24275d550fc173f77503332ce61d3c974e2944ccd8d68e445
ikmR: 9f1b6497ae7d50aa847cc71283d6bf3c3a4f28687fe19cab7a83d7e08206b7ab
pkRm: 04a459aac3f085c18c74d7db44da258ca513a51af1ceb738e5ada24461ee0ba146
6a87f2c072239adb2bc0dc1463a31a5755cc761101b9a3e7d93f8cddb6f56e3b
skRm: fb389576827bcc122407e70f30dd2631e7572527dcc908537def4a78205d032f
psk: 0247fd33b913760fa1fa51e1892d9f307fbe65eb171e8132c2af18555a738b82
psk_id: 456e6e796e20447572696e206172616e204d6f726961
enc: 044580890bca9e5e8cce209f3c9c1731373012c18c4224b99f21bd844789ef9646d
97a963d9a2f78ce6d7bfa5a8bf01f48f4b4df184f527b072782f46510c5e9e4
shared_secret:
aca9a7ab75de7d7354d8bcf0028a00f306581a216cb171af7f20295470b656e0
key_schedule_context: 01b873cdf2dff4c1434988053b7a775e980dd2039ea24f950b
26b056ccedcb933198e486f9c9c09c9b5c753ac72d6005de254c607d1b534ed11d493ae1
c1d9ac85
secret: 46d37126521f0de3bf90be29b5e5ed1b0f57a7aaee7a66b7705c82f6c70f2266
key: be7eaf974cdf589e8312169a03155b84
base_nonce: 4250c1f06d8a36fb964e5b86
exporter_secret:
e9c69341bcdf8b4b37756ff53a4fd2f6d5c8b1597fbfa1cf78e9af3dfeecb093
```

A.3.2.1. Encryptions

```
sequence number: 0
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d30
nonce: 4250c1f06d8a36fb964e5b86
ciphertext: b384d901dba02c6052e2ffe957f6332a0e342108e6647d2e7b2bd0b7b69f
49e27c8f0f5ec19ca537550618b872

sequence number: 1
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d31
nonce: 4250c1f06d8a36fb964e5b87
ciphertext: a1c1c47021c2c429cba30e675de4d1181b88c93f02ae4bdb38494c9912a3
177b77e21837e64b760361699c10ed

sequence number: 2
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d32
nonce: 4250c1f06d8a36fb964e5b84
ciphertext: 2da27c295fae23648d8f17508e0774ccb739b012a054a598a813767e74c0
e944edfede921108f60ab1643094e4

sequence number: 4
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d34
nonce: 4250c1f06d8a36fb964e5b82
ciphertext: 613613a4dff2f8e92eee91ec7aa7b56f9956bd2aa0faf2be13502aabb564
69b53cbfb9edb32be4a06c42a7144a

sequence number: 255
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323535
nonce: 4250c1f06d8a36fb964e5b79
ciphertext: fefe87557b20f7b330a8d86bb90535017cccd70c6279e55dd25be01284cc
e75ee52d9af95261c26d864dc24352

sequence number: 256
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323536
nonce: 4250c1f06d8a36fb964e5a86
ciphertext: ded3ac2a2e552a96e61d43c3e91398f3a5339e2f1ae0cb28ac0a7f99a14d
3623f5abba5fcece80bf939ce981ab
```

A.3.2.2. Exported Values

```
exporter_context:
L: 32
exported_value:
10d620103170f0555e7cd04e145cdd1833ee3f6c866ca3b16b92d05d67c72d90

exporter_context: 00
L: 32
exported_value:
ffa41fe8b285ed16d7fb0f5a4867d33cec18030dce739601710063c0bb30a9b9

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
4f878c039d18dfa92bbdaaeb1cae76968e2cfc5bba68fa56ad579123824a6221
```

A.3.3. Auth Setup Information

```
mode: 2
kem_id: 16
kdf_id: 1
aead_id: 1
info: 4f6465206f6e20612047726563696616e2055726e
ikmE: 8062d4cde05a279a8ef4cdc8f7ca43f1662bc9ad2fc44a5c246c8f520be9a028
pkEm: 04bc1605647feb6b48252c835c7b47899e513d0ba82b9246c947b3981af6400817
8522a1965c48e699e1bf5ce22b0939edbcdf1c7bb103a25a6f9fdb9b8cd7fdb3
skEm: 5e3bf594e739cfa790dc21dee74094f83a221dbed9e10652757f4a0cf5976ed8
ikmR: f89f532928e3f23be9d787efc3368fb825d6b2b5a83c72c412037546ab23dbe6
pkRm: 0402d8fa38e12de9421a9056309ea454bbe31c2ce717c55c7d35533ec9133f0c2f
b946972b5cee88f6f6318aee678a2a7adc6ad9db7c757e9c7c87c6005a58010c
skRm: ed65e49ac5a613f66e0f28ff2ed5edb553d21a0b822e31f39246ea954ec38e7e
ikmS: 33886922e11b46947ff1732d2b04aab4b0cc1f2df749a4260f2c5e263484770e
pkSm: 04377e03caea6fbf6a5be744b6294a4644312864066202957a5bdf63abbcecb52b
c3d7232138473af939622f3ecf9ed5b72c915785f55c7118c51fc7bfabf9fd8c
skSm: 38c4e7316db234ab4b43401d86dfa1294fa065e7c857a73f0e2db991b181cb2b
enc: 04bc1605647feb6b48252c835c7b47899e513d0ba82b9246c947b3981af64008178
522a1965c48e699e1bf5ce22b0939edbcdf1c7bb103a25a6f9fdb9b8cd7fdb3
shared_secret:
ea6b93f5344da09ae1f64c4cb29cc69123e552e44ab1fb205bcb5e89eabd55e2
key_schedule_context: 02b88d4e6d91759e65e87c470e8b9141113e9ad5f0c8ceefc1
e088c82e6980500798e486f9c9c09c9b5c753ac72d6005de254c607d1b534ed11d493ae1
c1d9ac85
secret: e6d88abf2af8cb7310d0bcb5f6f4da722ea7d22335eb690f6fb688e76057d23d
key: f77f4763aa41ffe999cfba0d2a8bd769
base_nonce: 124c755a5a9d9445e679181a
exporter_secret:
32449118a68c0db5f63c4fe9db37c2214029bd4db0b137fba08b907ac05ae075
```

A.3.3.1. Encryptions

```
sequence number: 0
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d30
nonce: 124c755a5a9d9445e679181a
ciphertext: 8448ac7c43bcaae17f8e40055aa51a73731797703a4e08ccdbe6c7eb7445
278609240d269a8bf553e897045ba9

sequence number: 1
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d31
nonce: 124c755a5a9d9445e679181b
ciphertext: f387e3ff926ce169cdaffc3ee8d6e0921624516af48a34be1a0157ebbeda
9050e9a97e6cb88eda687421ce1b09

sequence number: 2
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d32
nonce: 124c755a5a9d9445e6791818
ciphertext: a03db189863105001d5bd768d621b0197bbb44fdd83cb3e4937bde201201
8bc96f421b442a5bd733e4c777dcd1

sequence number: 4
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d34
nonce: 124c755a5a9d9445e679181e
ciphertext: 86e905b62676387f4dddf2c8b033c4d9e8cf389599992523da6a2ed7e836
d22ad56741f10a6004ed6b7a9dbc66

sequence number: 255
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323535
nonce: 124c755a5a9d9445e67918e5
ciphertext: 7eeaebd496d0da1869c41bf247e36d393f7bf922e91002331fcca495fd54
1418150c179424c3abe8c181dbcf0a

sequence number: 256
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323536
nonce: 124c755a5a9d9445e679191a
ciphertext: ba50ed518ac3e57303d081f9d4462cb29dc9bb0e0815c9f51dcd175a9c12
ec21dae32cee0e1b99a28b920cf69d
```

A.3.3.2. Exported Values

```
exporter_context:
L: 32
exported_value:
9af5e1a89be04b569cba0cc2c62e9770c95f127597b408ca2cf5f08944fbd2f4

exporter_context: 00
L: 32
exported_value:
b0dfce72c3ddfc6671b7efe9d52d7c332b3690dd48ec4ae3550127124c313e8d

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
6be74b9358c5b6169d2c752a03930e72aa2d8b74cc44e36ab096fe2be601a60f
```

A.3.4. AuthPSK Setup Information

```
mode: 3
kem_id: 16
kdf_id: 1
aead_id: 1
info: 4f6465206f6e20612047726563696616e2055726e
ikmE: 741aa0c6fc5ec64372a32cc48662b01c0b75db1fe2d7336e7723610ba5ba2725
pkEm: 04cbf5a3a7d785bf8ca6a1de5896009fbf87dfc60704fc0177068239e2200d74a9
4adb75017898ae49d8884913fc9b4b7719a47832c0861800ddb7b9a596617800
skEm: 30b464ecd2424f30a519d5212b573d6a38088312c575312438b64fd874b9b5d5
ikmR: 86cf426cc60770fe75f02decb23dd126b351475d5f27771751868ce92cf235be
pkRm: 04aa6664cc7044475c09983c507b6e4972bb6869383a66ab83d7264f977872c8b5
29b68397d28fc3a6d259fd70f7cbefabfa96b1d3d661a72e983bb8e0dc5ec4ed
skRm: 5dce991791aebaad1819d144b3293b75ced87577ab1a1c1e5e262d04c62fb6c9
ikmS: 69f8176ac02e8f73193284d46d39824531d57f0e67bf2ae3feac0bb128366f87
pkSm: 049240f92bc7d5faad04d39bc3f1236250ab59918efc3400aa4a463341954a4e1a
a95f0daf52e01c45ca77d73917d5650c255921124470a6a3605f29232f8bd473
skSm: d9ee3f9e2e73c5b681d143c9b8d1865bcfafc96678099803069b91fd0ab55671
psk: 0247fd33b913760fa1fa51e1892d9f307fbe65eb171e8132c2af18555a738b82
psk_id: 456e6e796e20447572696e206172616e204d6f726961
enc: 04cbf5a3a7d785bf8ca6a1de5896009fbf87dfc60704fc0177068239e2200d74a94
adb75017898ae49d8884913fc9b4b7719a47832c0861800ddb7b9a596617800
shared_secret:
9c0b65cba2f417717e3213daa085cbb57c5350adf4b27f91b9e2d3f66418a2c2
key_schedule_context: 03b873cdf2dff4c1434988053b7a775e980dd2039ea24f950b
26b056ccedcb933198e486f9c9c09c9b5c753ac72d6005de254c607d1b534ed11d493ae1
c1d9ac85
secret: fbfb23542c2769d2c0714261d184b04cb5a63a21b39c656886d317c355c5e790
key: 06100c5921975cc1aab7bc7528427e48
base_nonce: 6839d9a86d5fd8bcefa5a1ad
exporter_secret:
dadd70bee6374dabbdb80a0dbb8c1d5904b5cf1d04a8a84e7368828786af7cfc
```

A.3.4.1. Encryptions

```
sequence number: 0
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d30
nonce: 6839d9a86d5fd8bcefa5a1ad
ciphertext: 6ab348e5a953e00c6de6ec2a801b5dd0fe5f8d22ab7bf53bc7d967b03341
ab5a964c54f60fb8bbc2c94a0221c3

sequence number: 1
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d31
nonce: 6839d9a86d5fd8bcefa5a1ac
ciphertext: a1e28e5a3ba3c9e3e3bd824811f7ea2ff8ce7521d317e5da810cc93b0f05
a0483c3d35aca0ef6f1299fbe9bf52

sequence number: 2
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d32
nonce: 6839d9a86d5fd8bcefa5a1af
ciphertext: 07eded4415974d873aeddaa770552adfca18447dd7f08032f53e4fd5d0a6
2d803299921f85e1701c4edfa7cb47

sequence number: 4
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d34
nonce: 6839d9a86d5fd8bcefa5a1a9
ciphertext: a7e7d15c44829efab0c783eb814076aeef9ce7ae8ad789599658387b5bf6
3d2e097821d42999b03c1f2007ead1

sequence number: 255
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323535
nonce: 6839d9a86d5fd8bcefa5a152
ciphertext: 6bca7d01856e0dfe9894fe628b9207b9d7e9a9e01e53eb5b6ab8d8bbd8c2
35226c7c796a121917b90763ab6319

sequence number: 256
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323536
nonce: 6839d9a86d5fd8bcefa5a0ad
ciphertext: cb36ddb362dffa35bba568f33d193f93386a23fbfda0ca84980c10f2bf24
00b7a1bf9533bc32c0539c58342d29
```

A.3.4.2. Exported Values

```
exporter_context:
L: 32
exported_value:
8a4e49b960a7f20110f3a60f6d3797edec527db1e5ab97ef418642e0b1b47e54

exporter_context: 00
L: 32
exported_value:
afdbc7fae5f454ed5c17c53eeecf8cda213d92cdc48dcb197d5853467e325f7d

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
f2cef5c0adfb4398281eacd00943ca7693789d7769a2098c3026e39ff4912d96
```

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: 4f6465206f6e2061204772656369616e2055726e
ikmE: 92a0ed70a2adf2c5ad46064017dda7bddeff4dd7d221766117bac27d01064eb9
pkEm: 04feca0581199c3e379378e66b2723b4f276b520a91edc4719b0445f02b6527dfe
07f4b858c4592468dbfd04bc740476c9bfa2f51a4e1cea48de419182297b038e
skEm: e5077ab32ccc956b2162b713003b26d31eb608142cb549b715cbcdfc6657c3f3
ikmR: f3323a6b1967c7e8e707fe7629b54173e74df5c691a1942aac5fffc9ac64eadb
pkRm: 047ce5de52af235ae850760ec5d0c5a47d7b54dd8813e01b9b71ee2af75c596fe7
e4e4102490b042f01cf342987a89bbcae74d84d37d7d87f8bf00d729c4846eda
skRm: 91daf60bb3ce8a8552fe055bd7d2fe8a2cdbc8efcde552ca568ccf3f501d7674
enc: 04feca0581199c3e379378e66b2723b4f276b520a91edc4719b0445f02b6527dfe0
7f4b858c4592468dbfd04bc740476c9bfa2f51a4e1cea48de419182297b038e
shared_secret:
f7d05af8a43048aab05305e8163c95f9ea5f5632bd91a7cf23cf7539ca168421
key_schedule_context: 005b8a3617af7789ee716e7911c7e77f84cdc4cc46e60fb7e1
9e4059f9aeadc00585e26874d1ddde76e551a7679cd47168c466f6e1f705cc9374c19277
8a34fcd5ca221d77e229a9d11b654de7942d685069c633b2362ce3b3d8ea4891c9a2a87a
4eb7cdb289ba5e2ecbf8cd2c8498bb4a383dc021454d70d46fcbbad1252ef4f9
secret: 2bc978b799460b17b6f9794a0bc651b174aa4923a5f8ec2d149b713ae3834cb7
10590cdff49a0bdbad6ae2d81a26bea15ac6e613c09395f1620ea1a14613ebac
key: c0c0a3473b841724d11e6d50e971e23c
base_nonce: 16de467068dbf5560c0c290b
exporter_secret: ad8bea7e3a2eef94820aef8588c127c48a3b8e4bd48088391ed3bfd
702938e56f387583df84174f86e336ac85de582582189eecfd3cb20fb9c0118e3c98a907
8
```

A.4.1.1. Encryptions

```
sequence number: 0
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d30
nonce: 16de467068dbf5560c0c290b
ciphertext: b6e48055fd0d5fa0a806c3c6b6078f3e12bdd9922e4253114f79eda31619
473b11ee4b6202d4880a3198e7d430

sequence number: 1
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d31
nonce: 16de467068dbf5560c0c290a
ciphertext: f526d2914a2669f80252295e9b5ce0ed6b0bd946b0da5ff1b82c289ef1b9
3d05b66c3459add57952c02fbdfd57

sequence number: 2
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d32
nonce: 16de467068dbf5560c0c2909
ciphertext: dea8d5f2b523d43a6f5d61277b1102205cbbcb0b9fd2469013e66e2fec9a
90585306b967efd2ea1877fc6f6f2b

sequence number: 4
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d34
nonce: 16de467068dbf5560c0c290f
ciphertext: 1a02c1bd2adef2c3bb9b4abd8a389fb9e131cbce793a34bdc1bd75fee125
1808afff68bce3ce3845757e937f1d

sequence number: 255
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d323535
nonce: 16de467068dbf5560c0c29f4
ciphertext: b88db5d66c77a6e8caca5ab43e903d8a73be2b8cec889ddbbca7c605d9b3
3340dae47d0dd1360db8850f598f9f

sequence number: 256
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d323536
nonce: 16de467068dbf5560c0c280b
ciphertext: f54080fc4fd660169febeeb50e5557f412b2ba2d084becac2cdeceecf262
630e3e02e6f06aba9d04216e9d4c3a
```

A.4.1.2. Exported Values

```
exporter_context:
L: 32
exported_value:
13d3ca2708596af36f9d83f08957d53603d7ea8f77c7fb29ac287e5f34900b2e

exporter_context: 00
L: 32
exported_value:
52dc47db68f44adc4c65d16b704a0a3b6294b63186bc3390903ed8e09aae2059

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
f33267be2971d68f05bb299498b3bf8753b4f9f38b4d57808caafc3e32c6ce85
```

A.4.2. PSK Setup Information

```
mode: 1
kem_id: 16
kdf_id: 3
aead_id: 1
info: 4f6465206f6e20612047726563696616e2055726e
ikmE: ed7102804ecbc4b341c38798df8abab74a10315b6e121b0be17b7f1284d10b87
pkEm: 04a1e571da552292859a5fceb9f2389a5a1e4a25acdbd23605532645eba3026a02
5b278254cb0cccf311c1a30a7ddf3fe64a963288326ed5e8fe9979fe81068728
skEm: 443fa787757dcfbe3bab663722bc6e2be5cd807379c8196808bfe55296fd99f5
ikmR: 76005b73e0b0ba475d951f59336172011d958a3dce3281ce664e145637d582cb
pkRm: 04172f1359e661ffc33c610d480d6dc34e9462ca2f5794ed74e58691240fd5fef1
848cb9c649ac17e2a7bc9a1333736a859089162132fefe0463a5f8157774a9ba
skRm: a7f9b4af9b3f0b4b4c2fc9f53bdcaf993774752a47c1399faa930b2a399ec878
psk: 0247fd33b913760fa1fa51e1892d9f307fbe65eb171e8132c2af18555a738b82
psk_id: 456e6e796e20447572696e206172616e204d6f726961
enc: 04a1e571da552292859a5fceb9f2389a5a1e4a25acdbd23605532645eba3026a025
b278254cb0cccf311c1a30a7ddf3fe64a963288326ed5e8fe9979fe81068728
shared_secret:
89ba3cfeab0c167317ddab5d0dfae577d119d292dda323dcc56967f630331408
key_schedule_context: 01713f73042575cebfd132f0cc4338523f8eae95c80a749f7c
f3eb9436ff1c612ca62c37df27ca46d2cc162445a92c5f5fdc57bcde129ca7b1f284b0c1
2297c037ca221d77e229a9d11b654de7942d685069c633b2362ce3b3d8ea4891c9a2a87a
4eb7cdb289ba5e2ecbf8cd2c8498bb4a383dc021454d70d46fcbbad1252ef4f9
secret: 844bccdfc2f658d6ad8d7fde0cbccc80d4717c776ffcf099c0a5c8274edd3e5b
64380ba0cf7cc2d1f069481af490847f2cf16cc5b6b745e1083adf5698dee8b5
key: b5b03f9572b39518ac955c9c447f64ac
base_nonce: 55e771c657aaa0c3d1ca31c9
exporter_secret: 705dff024cc41ca90021af1a086ac050614884fcfb59ecfc71556d6
d84453912d3e4bca6b4b3a6e433db01ea39968303754ae0aac95e96a6d9ceb44d6cf7676
e
```

A.4.2.1. Encryptions

```
sequence number: 0
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d30
nonce: 55e771c657aaa0c3d1ca31c9
ciphertext: 57aee54fcdc5600b2499c65a4e846958b09fd33e7b11a2f6411e5ec76792
37495f66711395bc75d737ccbf138c

sequence number: 1
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d31
nonce: 55e771c657aaa0c3d1ca31c8
ciphertext: 78c44cc4824cc7976911f83ffa2e28464a61f4ee058dce4b131f0b0d5894
4b439c535ab81b3ed7bfd518baddb7

sequence number: 2
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d32
nonce: 55e771c657aaa0c3d1ca31cb
ciphertext: e45f01a429fe17107f4276cfb17375602a8407ddd239a8b33b3a072871b6
d8fa7cb12e1701669df8004e9bcd34

sequence number: 4
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d34
nonce: 55e771c657aaa0c3d1ca31cd
ciphertext: e9559781b27e48665044c6c6967671cf4e48386af6b65c3024997e00d8b1
2233eab7dd4ec0f43673500c849839

sequence number: 255
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323535
nonce: 55e771c657aaa0c3d1ca3136
ciphertext: 3154a0771cac2511f5a3e54a8050215d41400b6de3a3edbab401ad80d07f
894953848d4d19cf155b6bc9a3afcd

sequence number: 256
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323536
nonce: 55e771c657aaa0c3d1ca30c9
ciphertext: 8c507ca5cf5a184b659b5c5581e8bb23fc41b75b89608b64a03a49fba4db
679eab7db07652f2d8453638c1e85f
```

A.4.2.2. Exported Values

```
exporter_context:
L: 32
exported_value:
db6ff89db59370a4570ef50705539529b5bb29328e245db9aa4b7aa36a9a9d59

exporter_context: 00
L: 32
exported_value:
3f0eb5ee334e706e821c6ea8200b488cceda8a250dfae18638240afff35fb944

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
cb26bec375cedcde8210d4937d76136d749c84e892f42ca7bf41a3f6ac980298
```

A.4.3. Auth Setup Information

```
mode: 2
kem_id: 16
kdf_id: 3
aead_id: 1
info: 4f6465206f6e20612047726563696616e2055726e
ikmE: 1de4172bcfcea0613ed7901dd57e039085f7991f6cf5a4e5ff4f18db6cfd2f67
pkEm: 048eaea7f3b2f97a830a446edc645b45e1bb218ea4a53ad7715fae2a24e213f798
dea05ca2d6bf4951b9263b7bd63bab010922ba7627c35617d75bcc16cbe66912
skEm: db4e25e7e42523d00b3821a7c41af39bd51782944cced4575f4cc1bd7eb64b67
ikmR: 04e157810f9ac6f41cd4f34f1bdfe976aa6d69197399a64865b9a05a2e377dee
pkRm: 046a14182d61fb9f3a507f45cdc4b46c6e2d531ed4f34f56bcc0efc88556a457ad
e99e5c6ed8cecc65c9bcdb0838dce9e300c63e822854d1b719e49546574149f8
skRm: 03cca98e70145f2e719e78f3722dcaf469628eb89437bd91e11926a20ed550e6
ikmS: a101bfa89f76c75127ab4a8e3163272d6b0e8181b2343befcb4d0f956c104500
pkSm: 044bc49f888e8c531dc9f44416d19743c5827687a1223c564f284c646814f58455
67e5a6075b6b90f366a6fe3b171e3a50108fed3804c5fa0f4be90b3383fd37bf
skSm: 1d82c7053ba88b4029dfd57db0eb77ac3cf768644a70bb6e723c93d455530e1b
enc: 048eaea7f3b2f97a830a446edc645b45e1bb218ea4a53ad7715fae2a24e213f798d
ea05ca2d6bf4951b9263b7bd63bab010922ba7627c35617d75bcc16cbe66912
shared_secret:
7311fe0d7cca8e0dae7685133e0802bfdf83c1c68116b7ffb7d7231861c05e3b
key_schedule_context: 025b8a3617af7789ee716e7911c7e77f84cdc4cc46e60fb7e1
9e4059f9aeadc00585e26874d1ddde76e551a7679cd47168c466f6e1f705cc9374c19277
8a34fcd5ca221d77e229a9d11b654de7942d685069c633b2362ce3b3d8ea4891c9a2a87a
4eb7cdb289ba5e2ecbf8cd2c8498bb4a383dc021454d70d46fcbbad1252ef4f9
secret: c09564d17226a0d8d4cfeccf416bbebdf2e7ce87f1dc7f9cf7dda3d8ce6192dd
b6a6179f8d3f0c060c5333ff97ccc22189b6bd7e33bca541e5d298517074d406
key: f71fb8d0c28bd5d76a337f06607fdb92
base_nonce: 7fc03640bde39e9b173e6083
exporter_secret: d54619b13e7307e1e5b338f420b967fab3aeed904aee3a1f68b447f
575a63070d74e77a3494f7bb5d2db20643c7e127e9d79cec990cda80ce7dd714a535ad1f
2
```

A.4.3.1. Encryptions

```
sequence number: 0
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d30
nonce: 7fc03640bde39e9b173e6083
ciphertext: 8f1b32460349d0c595792cbab00320ee75c99a49c4593955a11714f1bbed
a64a2a98a8d931b9715eae278d3041

sequence number: 1
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d31
nonce: 7fc03640bde39e9b173e6082
ciphertext: 31450e2b97229cbdb0cd1aab1b06a1d705e78bd00a33bc55219570006c17
2cb26cc197fef8183b88dd6064c0f0

sequence number: 2
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d32
nonce: 7fc03640bde39e9b173e6081
ciphertext: 6e19ed46a69d0d246d28fe69d1f5f16a7c31dced1afd0a19b91b7fda8222
ae72083e9ef8054653fd0663501bd2

sequence number: 4
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d34
nonce: 7fc03640bde39e9b173e6087
ciphertext: b9b56e290ae56752a4f32a05e59b94c85dae58d164a5f3736f542361f303
d44f83e8b4f68b4bb0edb14dd011ff

sequence number: 255
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d323535
nonce: 7fc03640bde39e9b173e607c
ciphertext: 2353d612842f82fd7868b491d5f83f0a536559840baa38d5c64cc5c6e94d
93e079c16e724389d494520536f4f3

sequence number: 256
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d323536
nonce: 7fc03640bde39e9b173e6183
ciphertext: c5ba37eec59e5da80949d76af3df1b1aac8afebeb92a7fb4099cfee7018f
7a5dc6543cbb4ad85c95121989e27e
```

A.4.3.2. Exported Values

```
exporter_context:
L: 32
exported_value:
15b8ca5bf226bf9924d4dd3c770d9e9f534e4069655e29d477f69a6fd983a830

exporter_context: 00
L: 32
exported_value:
fbb906f58e73f64d6d664417a33be2555ce2019f96905b9e7a3a35e4bbd2488b

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
433012037ddd16977cce565e09a6d144d159903452c02b0910a6ec37fb59f400
```

A.4.4. AuthPSK Setup Information

```
mode: 3
kem_id: 16
kdf_id: 3
aead_id: 1
info: 4f6465206f6e2061204772656369616e2055726e
ikmE: ee2ec13874fece5f5d8ce261dee3990ffe0bca2c64ac43ea16a2a1f9aba67326
pkEm: 04652f4ea1af053a798309dafbc6914c626fb4bd74b58e59626202616846ca5a14
ad775625d7f367fbc604848abf6fdab5ebcc69661222f636e2a193c0495ec972
skEm: 1f0d98c09a4d87bfaa26b4b033ec35fabea3f8c7ac451a12e5a31d48dfb55a54
ikmR: 464ebc07ef77a6bb4199d29000c06daa56f3bb3edaec71eb196e4e6d3c217a4e
pkRm: 0489f50dae6951783eba36c3a547461c30008ff6622d9d127d110425784a8f8256
138760e3912933aaedc5a150b989bc865c4e00ca4f65d5392362cf66678f0fad
skRm: f41b1f9aab1c994fb377ad3aab673e3d73fa8e96e5c813e66b67297c712329f9
ikmS: c8ea69798ad512161ed789fa8b6a84923796c6e6ec808001f50e21967e4bfd6a
pkSm: 0458b5ca140547123bdc9064cea10fa15e15f4f64c9e9304813ecd3b801fc7d647
52844b2823c48590313e1a486621f7ab141f80c5b2c25eaaa4dac79bb6b67e75
skSm: e2f65d9300a0d46ce4577652fd18c70c6b8c4a76c1dd4442830d4c4e747467cf
psk: 0247fd33b913760fa1fa51e1892d9f307fbe65eb171e8132c2af18555a738b82
psk_id: 456e6e796e20447572696e206172616e204d6f726961
enc: 04652f4ea1af053a798309dafbc6914c626fb4bd74b58e59626202616846ca5a14a
d775625d7f367fbc604848abf6fdab5ebcc69661222f636e2a193c0495ec972
shared_secret:
a9a1bdf17b345553038713662d613c4547cfb3b3ce1ba98571da4d3d1e40da1f
key_schedule_context: 03713f73042575cebfd132f0cc4338523f8eae95c80a749f7c
f3eb9436ff1c612ca62c37df27ca46d2cc162445a92c5f5fdc57bcde129ca7b1f284b0c1
2297c037ca221d77e229a9d11b654de7942d685069c633b2362ce3b3d8ea4891c9a2a87a
4eb7cdb289ba5e2ecbf8cd2c8498bb4a383dc021454d70d46fcbbad1252ef4f9
secret: 1e499a79ab8d4028bca025bcdd95264a39cadc37123af22ae889e29e82c7f28c
c3138c8330df8d94a3729b341c727b313cb447c70a02112e95d84d7adbdf3076
key: eec30ef9db6d0644f0554f5f82be7496
base_nonce: 4efce095cca5c3ec361b4279
exporter_secret: b23dfc175ac2762266a3b83cec4f404189009076d0aa2db5041cdbf
fb6b0bbb5248523afab3c8183d7f32aa65bf3dea35329ed3c36c12bf30381626ba446bf3
e
```

A.4.4.1. Encryptions

```
sequence number: 0
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d30
nonce: 4efce095cca5c3ec361b4279
ciphertext: c8c87b83059cd9ffd7ef48c3994383059c9adfadf9c3daf71c57e5f875a6
d911725d601708939ffa639bef253b

sequence number: 1
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d31
nonce: 4efce095cca5c3ec361b4278
ciphertext: 4c12ddee0d58e2a12daf92a3914e3a7640435a3e79ab153698639aa7315c
b4ac7c69d30dc7b4ba55392ae13106

sequence number: 2
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d32
nonce: 4efce095cca5c3ec361b427b
ciphertext: 66d6b14249225950ce08f63aebe37ff3fa928c4c2957c02ad557d7d44124
5f1c95ec2d0c16caf79661d3cdf32d

sequence number: 4
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d34
nonce: 4efce095cca5c3ec361b427d
ciphertext: 0dbfccf43ee2c2ee8dfcebd212a5bfeba171236931228b10f5c0aff7e673
94fab97a98ef94022dfdb4b42ba951

sequence number: 255
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323535
nonce: 4efce095cca5c3ec361b4286
ciphertext: 526a2dfdd4fe51764b1e2383970d82576cb2d7911bfcd20e7e05e143aec4
181d837c9bcfde8072cfd8089f20a7

sequence number: 256
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323536
nonce: 4efce095cca5c3ec361b4379
ciphertext: 255f576f0be7bbe08acd09d228e3783b16b4388a0e3cfd450cd6f68cc6f0
b144829430977a877104121b53b177
```

A.4.4.2. Exported Values

```
exporter_context:
L: 32
exported_value:
6ad40ac5e73b9136646d91edabd3d13bc0f10f550276cf1e52ab013ab10ae48e

exporter_context: 00
L: 32
exported_value:
ba058dff7765a1734ea0bda12fd59c50ac2a28d727448881fe19305904fe1af1

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
c4663c3044c52cbbfeb67391423ae29c17caf1bcbfbbb6e0f53826f4b29d8cf2
```

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: 4f6465206f6e2061204772656369616e2055726e
ikmE: b962b3a828601bef545646ff117408f06b23417df1352c7728663c2834503d12
pkEm: 049b53cb1aa6004646da50eb6c96bbe3543c5054f1758086d8a5268316dbc423cf
e6fe4b0ce9902eaa14ac8ed48449a0597cad62345b2d73bc8b1fbcaa108782ab
skEm: 4bd3b845e64e673a36c52ea2e819b8291f2d381b15d7649c41870c14c3bdcf8a
ikmR: 565f23469bf1d7dec4429ec38443a45dddf287754324ceb72c664fab7fe343cf
pkRm: 04c67b9451ed0c36a980b3a20876b9da797e814234177d1c3365c786de5fdabf2c
4b19a0f7bcd5567132d59dc08eae061e87c29536e245f95cbb83ab1b2f74164c
skRm: 0edded4d3adc8f3bf601579d6e803e9a4bc7eef8beb3fd33e1196ae75413a6f1
enc: 049b53cb1aa6004646da50eb6c96bbe3543c5054f1758086d8a5268316dbc423cfe
6fe4b0ce9902eaa14ac8ed48449a0597cad62345b2d73bc8b1fbcaa108782ab
shared_secret:
8493bd3b5eb99e1f93ef3822860e97eaaebc0d02f6a5d3e74b79d29dda9af7f1
key_schedule_context: 00b738cd703db7b4106e93b4621e9a19c89c838e55964240e5
d3f331aaf8b0d58b2e986ea1c671b61cf45eec134dac0bae58ec6f63e790b1400b47c330
38b0269c
secret: 9a267f765989b3a21dd18b1f1e99acec3b5ddff46c6d1680e6a7807d4b9adc46
key: 70baf5e6f34736265b72b423500bc71b3828e5f78404b355850034f2c1fa921c
base_nonce: 991700c41169507e6ee5600d
exporter_secret:
ca013271612b219874d7d76dedc14296e3bb65885a1016e471c052027d5c76a8
```

A.5.1.1. Encryptions

```
sequence number: 0
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d30
nonce: 991700c41169507e6ee5600d
ciphertext: 74c9155d64e6ce4f667f8267a326e4014d9b5c0074937b02e1802f75f29e
b56471d02c923ac125d9cee9c1a5a4

sequence number: 1
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d31
nonce: 991700c41169507e6ee5600c
ciphertext: eb002fe214bf4643fb4e70bc8662ef4753b75e2c47b989524aa0eab9e4de
a869be1fa28120980addfe75b43f14

sequence number: 2
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d32
nonce: 991700c41169507e6ee5600f
ciphertext: 81ecafe01f618f4acc1a680763a2ec28dbbef8509b1f43691a60ff9aab7e
801ac72c6c8c99784c7c4b16287ea7

sequence number: 4
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d34
nonce: 991700c41169507e6ee56009
ciphertext: 25d24e72f6dff4e4a6f283a68d990e303d04d44fbddeaab7078aa44e2edf
f82be34105b94a154fbde7494515af

sequence number: 255
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d323535
nonce: 991700c41169507e6ee560f2
ciphertext: 706097d0fdb2485330b4f05e13b775bd7dca86c7023f2e4212e059b63c47
d4a3e433d8e48ce4490048d61cdd9f

sequence number: 256
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d323536
nonce: 991700c41169507e6ee5610d
ciphertext: 11e683139b5061f3017e4ec0415c0d31ff94ed45a96e5a871a8a6e4b1aa6
d667ed762ab393206270a29970d274
```

A.5.1.2. Exported Values

```
exporter_context:
L: 32
exported_value:
6d00e1d04e62a7581cc6ef222a85c505bc55fa366188d3d1f0dff0c7b5529130

exporter_context: 00
L: 32
exported_value:
f85d328a79b510416f9811d29f2a73d64b52dfbd892e0f2c8b2040cd9a0c7eb1

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
52686bf245d640bc3e35a8c8230aa4e6765eedab049edb438f66e3bc64442708
```

A.5.2. PSK Setup Information

```
mode: 1
kem_id: 16
kdf_id: 1
aead_id: 3
info: 4f6465206f6e20612047726563696616e2055726e
ikmE: fe33f50c45015f93e3e37dd270c3c5b039d4aa01de88227afffeeb8c3c6166ce
pkEm: 041c2ff4daf657e6b98689468fc7e9d59f8099fdbdb472eae4f877fbee6a78aed6
52fcadb6858fc54b1026ff26c1cde00fa8b0266ed47bf127a3978371e94be85e
skEm: 8299a99a0d429530be5436c4730a016a99b96024a68d239a1dcbdfc5a4ebff77
ikmR: e6ea53d940abe2f08af1da06622e309396e4d7d218c41d00b9eb82e1c3637531
pkRm: 04679e98f97235de2d538fcaa40dc1d918cf66e49843ee03858561087849592393
588a041a9d3a213f349bfbaeb615e3e7a1c43bbd5685b826ac70b0d020efe947
skRm: 6e930d13d90ac31b5388e6409463b5731208882d738f67067060eda4e484655e
psk: 0247fd33b913760fa1fa51e1892d9f307fbe65eb171e8132c2af18555a738b82
psk_id: 456e6e796e20447572696e206172616e204d6f726961
enc: 041c2ff4daf657e6b98689468fc7e9d59f8099fdbdb472eae4f877fbee6a78aed65
2fcadb6858fc54b1026ff26c1cde00fa8b0266ed47bf127a3978371e94be85e
shared_secret:
44eb1433b722fbe97df1df2dc40dd54b9057774cbf6f2bc2f14b889e6a808acf
key_schedule_context: 01622b72afcc3795841596c67ea74400ca3b029374d7d5640b
da367c5d67b3fbeb2e986ea1c671b61cf45eec134dac0bae58ec6f63e790b1400b47c330
38b0269c
secret: 28b21c7a59645e4cdfcc4c0e28b4d8777fd0ae25b8c60da9f25202df83e7754a
key: 21e540160f6bb520d0fd423e4f65e23aa09b859a0cc533df445452bc01a31e2a
base_nonce: 8d5058f6faee855e19fb6e40
exporter_secret:
465070d3afe9e5e6c3dc92b2ced7d2575f3ff6b8025c340795ea1ef24c182149
```

A.5.2.1. Encryptions

```
sequence number: 0
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d30
nonce: 8d5058f6faee855e19fb6e40
ciphertext: fc2ec19752defa8eb833ffd9fa2528257387af4022685bd4236acbbfae98
02333036d599cda9f9d2448475046d

sequence number: 1
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d31
nonce: 8d5058f6faee855e19fb6e41
ciphertext: 98fbea1c17fc7c3568b4c720b5fc83ffdc388e000179e76d4d9d48113e4f
800b30d05ee098596b94e9d807e6f6

sequence number: 2
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d32
nonce: 8d5058f6faee855e19fb6e42
ciphertext: b5b6851d940fcd02a6fbbe319e7f1ddae826ef8536a13c8cead1cd005682
4e0a474a51e4ec4120ae45bb7e830b

sequence number: 4
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d34
nonce: 8d5058f6faee855e19fb6e44
ciphertext: 9fdb6c9581abb2973b3674ce7881292ce01dcc014680cfc6740c2ca831c1
2b83c09647b55f7134b74809c1422d

sequence number: 255
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d323535
nonce: 8d5058f6faee855e19fb6ebf
ciphertext: 415cb7cfcfe2cc289a0e9b1495a6f3ba823db29d296b56bdc444cf9ae9c4
194f27e2e7f6465dc71e85ccb0c9ed

sequence number: 256
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d323536
nonce: 8d5058f6faee855e19fb6f40
ciphertext: 100632412552c3d8a61fb0a3143b6e17b3a3a97655cd02977ef4aa6782ca
674b78c22b9e8bc6d59051c350e3cf
```

A.5.2.2. Exported Values

```
exporter_context:
L: 32
exported_value:
fe1c70a1345927810fe4829074cce85f40a3d3ba730603d6c5c89b5a0819aace

exporter_context: 00
L: 32
exported_value:
aaee3d3794b39bee27fdd7346b1ed0a9b4aa14902ba19f55a5a25d691837ddfd

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
276806c2d6bc889869baa6a5daa6c37061cef9ce18afd1da2db49e69527d1758
```

A.5.3. Auth Setup Information

```
mode: 2
kem_id: 16
kdf_id: 1
aead_id: 3
info: 4f6465206f6e20612047726563696616e2055726e
ikmE: b9ebf57515a283c9f07b008d595e4235e24d9f51b4ca122324d16d2d21105ca2
pkEm: 042a6f0d82e2a5b26221ab792e7a20cb5d4a00aaefb4b567b681eda62061f3b1f4
0cc030bf435c96ad84d2779f296ef34083f4a3c24e889b495d4ed39448dc00fe
skEm: 6735a602ebb7cd784a7c346e021f4dd751c0d75b21a22301eb2203e9767c13b3
ikmR: d739aa3b63b5a4578d02f73e8c816e3d47a76e13c67f11424a43f16df804ba6f
pkRm: 04969eb48e5d51742b1b7b894a9663bf4d611a3b1e279b3eba770862a39d10beed
a63ec435e24d2995d55a0c01ac721cbf7641b73d145fc71669b2223e16d65f2a
skRm: 3b6617f0adf7712ad9e703cda709dcf95f18c87ae7fcceed6e48a89bcfaddd7e
ikmS: bdae20856b3f57de5b6164899a9ee58d6a783768a15d73b600355e11010a41c8
pkSm: 0475640437118869621c90d5a28d64d4ac54f60329f2df8ce24e6913de257e6194
8bfff5915339c616dc76096563250fb9b43096427dba1c59dc6c770a07bc660e
skSm: 2f6d391c7f58bfd0c6710f8f09e4dd212121ed2971092eb94a28431a371f60ab
enc: 042a6f0d82e2a5b26221ab792e7a20cb5d4a00aaefb4b567b681eda62061f3b1f40
cc030bf435c96ad84d2779f296ef34083f4a3c24e889b495d4ed39448dc00fe
shared_secret:
d90c1f6591f0f280c081466103f743bb1a35edb167be9ae314f0c06349e75f04
key_schedule_context: 02b738cd703db7b4106e93b4621e9a19c89c838e55964240e5
d3f331aaf8b0d58b2e986ea1c671b61cf45eec134dac0bae58ec6f63e790b1400b47c330
38b0269c
secret: db0ac5f766d6a76c205d6c03ab42c5720c57b1688d1d3f3eb54d4ce6318902d8
key: 2e2094c37bffcc58a4bfac89323aa4ddbd2625a146775ff02d6995fade6b5c1a
base_nonce: 8b12afe06aba01dbf3fb4a25
exporter_secret:
1f695d99b621169c6d1d05ceb65c1d071d85962cfff863081d630d2998bf72ed
```

A.5.3.1. Encryptions

```
sequence number: 0
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d30
nonce: 8b12afe06aba01dbf3fb4a25
ciphertext: 7d6ea96e48fb2415e2bff2923b93a68e0d27144b4af33c2628aba4844515
282ed1159b258d086cf2b4ed9127b3

sequence number: 1
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d31
nonce: 8b12afe06aba01dbf3fb4a24
ciphertext: 224c5046b185cff41467a4cf7658e415ef17743b9a7ad2048bbcf05df79b
7df1f8a35cc4eb18e1f4b8da5b0c9b

sequence number: 2
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d32
nonce: 8b12afe06aba01dbf3fb4a27
ciphertext: 57dae566c98c844526283fe1a264a3b0a0ac14bcb1e90ca7008d8aeee806
c409a5ad2df3be59d6885d3c3ae800

sequence number: 4
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d34
nonce: 8b12afe06aba01dbf3fb4a21
ciphertext: 4931c9dbb1a59c76e07d3b4b2bb1d8df2c9700866253c90a938d4fe757e6
d9c55fec82be11c78b6201d93d36c6

sequence number: 255
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323535
nonce: 8b12afe06aba01dbf3fb4ada
ciphertext: 9c85f47168932b7aeea2021d4ac4498a11f4c8f044abe138993d5e75de83
91bf16755d3023bc62ac62303b1a84

sequence number: 256
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323536
nonce: 8b12afe06aba01dbf3fb4b25
ciphertext: 1c53da0969b652a0e68487cd8a5c37924fb5bec3297cb1b76895e8aa0e15
c8a54557b224e00d60a70b71ba9bed
```

A.5.3.2. Exported Values

```
exporter_context:
L: 32
exported_value:
eba9e7b2561729004a41a63a0d0a8c0c32df6ac648bfaf2572921eb06bcdeab8

exporter_context: 00
L: 32
exported_value:
bd197199251b848b7be9541642ea42c9923cc877617e21086abbb09a5744fce4

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
65c80f44834b083b6c03167ea18ea488e31d889fed20081c9bb621a086c8894d
```

A.5.4. AuthPSK Setup Information

```
mode: 3
kem_id: 16
kdf_id: 1
aead_id: 3
info: 4f6465206f6e20612047726563696616e2055726e
ikmE: 0dab8c455c70e2b3e8aa005079c549808aa962acb883b9785be1a9dd8df74516
pkEm: 041f60c33669011334c7e381f62c5721880703c0f3c3e4f48fbbdcbf634f9c07c6
501b15ae5f7769f5d322a7ef50568ae68042d7a1a446fcd5335f4b398a81bf46
skEm: 3c6be324b2ce4714afdfe35dc2a89080eea77fda921501319edeb60ba2fcc3e9
ikmR: 0c8f261cd3b28808a654626ad4c5a317b60b4214e2413060da887888b8e92044
pkRm: 0418c1b436ff369770e1cb77f7714153a5e5813be25ae73a4087c5195388686e26
d6c163796894481d049156477aed1f00f8bc1fc9d752695fbe04487c4cd6ef60
skRm: aa852b071565f4377c61dfe1284b018ff95d9ba5f57fe00e062657f118bcce5d
ikmS: 3d54b26abc25308fa2afd191c70e061af37aba14680095e3c5f7d479edcc618e
pkSm: 04d03feb9de4104308b301f1dbbe2fac0d9ca17a43ce6f77608202e3315c62969a
40c5f5e1c8090233a1c875b929985388d75a86c49a9306392d1cd7bbf74f22c3
skSm: 5a139206c68fb86dd348e809a4016171dc8cb1014e1930d805afd4ba65dda45e
psk: 0247fd33b913760fa1fa51e1892d9f307fbe65eb171e8132c2af18555a738b82
psk_id: 456e6e796e20447572696e206172616e204d6f726961
enc: 041f60c33669011334c7e381f62c5721880703c0f3c3e4f48fbbdcbf634f9c07c65
01b15ae5f7769f5d322a7ef50568ae68042d7a1a446fcd5335f4b398a81bf46
shared_secret:
99b4c46712a0b8743e8e0d64e565654a3cb00a010ca4b2075d6ccd8f2b975db7
key_schedule_context: 03622b72afcc3795841596c67ea74400ca3b029374d7d5640b
da367c5d67b3fbeb2e986ea1c671b61cf45eec134dac0bae58ec6f63e790b1400b47c330
38b0269c
secret: 67ede433394c24d54c79d344501b554f08b203757a5526be4e80ecedd7fed7a0
key: adc8af7c821a748315c94801694f2b2dcd1ec2df8e7dafdb8a54b708eb8caa0d
base_nonce: 198f11ca53b11dfd8472dd1a
exporter_secret:
c5b58e033e236a420ec1ff056c8018dc77ad8eca6ddeab0e48f081d29fa49afd
```

A.5.4.1. Encryptions

```
sequence number: 0
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d30
nonce: 198f11ca53b11dfd8472dd1a
ciphertext: 1036b955fa72326b07755abeaf034fb4e78513599b25928a84b61c593c37
30e756458d37119b8f576976113e19

sequence number: 1
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d31
nonce: 198f11ca53b11dfd8472dd1b
ciphertext: 87c25e56a88e802958fa2fc41f012971eedb6717f61fbe5aa1c513c1f23b
ac00357e9b19a6c3ec17f20b3ae7e2

sequence number: 2
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d32
nonce: 198f11ca53b11dfd8472dd18
ciphertext: 96d27aa40b0d71d76125e1d2d90e0602009638e4df7cba8faed64a632b4b
1c08d5c6e1d542c76f667b82a5edf8

sequence number: 4
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d34
nonce: 198f11ca53b11dfd8472dd1e
ciphertext: ed623241d3265fdbc0bc8b2efcf1f758729819f1dc687e0a29b9eda1485b
d410657511f0d5dd797d25c63d6049

sequence number: 255
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323535
nonce: 198f11ca53b11dfd8472dde5
ciphertext: 9d75b0f1b7e69cb8a0e2dce87705070ecb5fda34a45224a0e5841206a48a
cff4d529856aa14c92fce1484b6d42

sequence number: 256
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323536
nonce: 198f11ca53b11dfd8472dc1a
ciphertext: 59cace8e1405fc3e21c23482711108842e010a29b539542b10449f015080
0937d7a29a31d69c52920048df246e
```

A.5.4.2. Exported Values

```
exporter_context:
L: 32
exported_value:
c8e0071643d846c1c8fc96c1dbde3f8b360ed4c7b751fcfc12ec8a608c8b1a68

exporter_context: 00
L: 32
exported_value:
ffdf33b89269c9bbfe72e5884a14979ab9b75d12a9ab4d76eb8f0e37c2da86bb

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
4ffffc850b4fdf16a5055b82bdfc15862f2859d46bb1e1ce7efb00dc8463e58a
```

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: 4f6465206f6e20612047726563696616e2055726e
ikmE: bb8f0c6c0d81a0a54d6084e3d7d8001875ce2b4831400075246ddeabc3f62d065d
91fa8f2ca0cfee3a94f1fa6f43f53a2e39241ccd5ad9f0d889aba103e78286ca17
pkEm: 0401f1f6d9583bcdaf771212a996542e370a73e9e36a207eed1168f4142a139832
f9ef2f8fc87664f3d9074dedca1b01c2d46b5095e3f0d561ca9d364dff041ca6f3b400f7
82800038ea8e52a752f5ffaad25df0e971c53803199c7f0ec96283d874d9184c6d6f74d5
bc9533b91d6b7183a073ffb8d0d1d2006deba12e10529fc64ee025bf
skEm: 01a33bedb85d1993389a3e524477411c8c088572e2b3e160eea59b1a074626262d
0409e48ce9fc7ed4af8e298669e20646895a6460666b23867135c549ac8ef45c96
ikmR: 28b1403adcf3d2bba7d7c733df4ef8ab9f4c2cf92c840beb7de3fd139611a95be8
50589a49825450c033b2e1f6cb66127c83827d5464fca1e427c2bb31bf18e857e2
pkRm: 0401cf0f6149f3c205096fb29d415a38a3a10c5e882822b582220ae74230f78d18
3b92824fb2b1d9b005b8af49c43fe2341f210d5262da9b97bb3ae750292656f63d39001b
4b39296e906399e82a2b4413a3b2ff2b2657a166c1b85926d33190ad79f7bcc8b0a80092
e93c6998088c87bb5bb372e596b902fbf100f2fb0a24f6392b4a444c
skRm: 001926452a313ec1ffb91ec82081e0c2c900594cfdfd0f83391d1574e887fa9e9e
49fb1f3cd9dae4c230c932d043d4def44bc7a68780a7c94ffe643e692245fda188
enc: 0401f1f6d9583bcdaf771212a996542e370a73e9e36a207eed1168f4142a139832f
9ef2f8fc87664f3d9074dedca1b01c2d46b5095e3f0d561ca9d364dff041ca6f3b400f78
2800038ea8e52a752f5ffaad25df0e971c53803199c7f0ec96283d874d9184c6d6f74d5b
c9533b91d6b7183a073ffb8d0d1d2006deba12e10529fc64ee025bf
shared_secret: 3283c13d50bea6519df780c0724eaea0d6cdd3e4d28f0bdd6e9be3311
71f2db39a9b02167f0dba3fc3e7df4ccd9dfd02fc8ac67d535ee08ff5a287884198a960
key_schedule_context: 0083a27c5b2358ab4dae1b2f5d8f57f10ccccc822a473326f5
43f239a70aee46347324e84e02d7651a10d08fb3dda739d22d50c53fbfa8122baacd0f9a
e5913072ef45baa1f3a4b169e141feb957e48d03f28c837d8904c3d6775308c3d3faa75d
d64adfa44e1a1141edf9349959b8f8e5291cbdc56f62b0ed6527d692e85b09a4
secret: 56cf3fbd135b2c3608795e6ca3fe7b70dcf787cf7923cc573e66883d33f0217d
e29149cd536b615666ab46384e8dd1edc43c9c0f28ae84fd31bd9a913114fa3f
key: 862fe13e4860e0673e31c9f3e19a7190faa3a170b3aca6042e1879687890db16
base_nonce: a6e3f71000879a06930a7e10
exporter_secret: fc45ac88c7b8e3d498a5e4f704b864de5832c15f80d0c5fea9e4bb5
2de4cff821b2f25b8d52e93852524be6a751de6144d659758f85aba6639babacd325bb8d
6
```

A.6.1.1. Encryptions

```
sequence number: 0
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d30
nonce: a6e3f71000879a06930a7e10
ciphertext: 554474e2bd659202052be4a523a85b323ba8f79368794b03d9a737c9f051
5945fd420bfae77098cb69f1c7c167

sequence number: 1
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d31
nonce: a6e3f71000879a06930a7e11
ciphertext: 168e20a7c425ef17e90dc57c9f2a66273ac5487eef7cfac9e5013791dcc6
c27a163acab4e1fe0adf35792c8327

sequence number: 2
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d32
nonce: a6e3f71000879a06930a7e12
ciphertext: 50afa46d709c5f13a29534b2f4e5401c42780c0ab59c564e37e1bb7a04f7
af4a6ec3da77ff499b459888d39709

sequence number: 4
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d34
nonce: a6e3f71000879a06930a7e14
ciphertext: 94a72b9d570c131173cf4228d53e8ad55422cd55171b6f117d636bd7d069
fda14376262790e0db47e6223d78a9

sequence number: 255
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323535
nonce: a6e3f71000879a06930a7eef
ciphertext: aef5c08d51466cbcd5161ac1f94febccfad6915752f47c525f4a7164102c
5250fb6ecc89892c5463d33daea754

sequence number: 256
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323536
nonce: a6e3f71000879a06930a7f10
ciphertext: 49a82248f2c9c834227a824dd6c8acb030f9f53b5fdbb7710116bf9908cd
ecf6a1924bf3cac67c23b83ddd2345
```

A.6.1.2. Exported Values

```
exporter_context:
L: 32
exported_value:
3b57330cc3d9a6187011d50fcb29587f588b1b906bd0572bd952f2d6370b9a91

exporter_context: 00
L: 32
exported_value:
09aaf1ca160ce21cd25a45c78ec9033ff41da87b0a026f28a853b1dac8e5def8

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
654ee749178b028efe413321ba83453faa4d95b61550000b74f727e94a111e6b
```

A.6.2. PSK Setup Information

```
mode: 1
kem_id: 18
kdf_id: 3
aead_id: 2
info: 4f6465206f6e20612047726563696616e2055726e
ikmE: bd12d21f5a36063ec0242aa26f22144aa7d24c3cc2e6a1805868250d8e9ce6fb2a
04f8852cd6ec3241249fcf262b8f245d179636fcfbf275a75539cfc756a4dd3f53
pkEm: 0400e7b838035aa5a1ed5c405fd0d0c24878b95dd69f21a838c7eb74255eb9564b
ab5ee9d85200e233cc19d6c879318631973160a978c218e5477f1dd4114e2cf932eb0108
0401d3933f56f8167db6ddc922081750fa2d93b5f853f202dd7c9e0dfad662079d464974
e1042623a4aa4925e9f8ec0a1c7d6b66273825929bcf5669c0ea4f68
skEm: 002a7455582d9757faa99131bca1401e30eb17ca38354deff31dc4d23c4adb7f37
2e5e484f40aa048ad763583b4095a14d57a29a0c6491a4c150972efaa8bdf91e74
ikmR: e9a016952866c33da399e64ecd629cf32826285698726f2c22debd4799b8fac112
ac1a546ce39452406396a389fb10f51036c7022dc1cdde177b54d005731fb79ce4
pkRm: 04017c27b164b369382597e1e3f4d5e4f4bee43fb95d39dc09fe8f9b6c2ee342b2
523bbf30238518dbc338cdd140879f12a5b439815e8a8662008f77198e80a33936db0129
01dd36d837faaa995cfa014b0439ca25a0ce1200140ec43b482226beefd1086758940c59
e34994737827df08b34affd812554c26d6449fb4dda9a249d0ee6a7d
skRm: 00435ad3bc01485574a909a79b60cd839ec53ed4c56b7f4677e56253c9c4d0e02a
b27c27e072961ba2d4473d4a73a3dce0b585c2b5d61caf9b5d89bed7b32bd5481f
psk: 0247fd33b913760fa1fa51e1892d9f307fbe65eb171e8132c2af18555a738b82
psk_id: 456e6e796e20447572696e206172616e204d6f726961
enc: 0400e7b838035aa5a1ed5c405fd0d0c24878b95dd69f21a838c7eb74255eb9564ba
b5ee9d85200e233cc19d6c879318631973160a978c218e5477f1dd4114e2cf932eb01080
401d3933f56f8167db6ddc922081750fa2d93b5f853f202dd7c9e0dfad662079d464974e
1042623a4aa4925e9f8ec0a1c7d6b66273825929bcf5669c0ea4f68
shared_secret: 8c32977bc310e95106ba15ef64c7dfe17fd3ed3879f6430cb2d1fcf11
bae1cd8107dfa03dc8c54f20206c07ca03da3c30705a660f82d080e62a6a69a15460931
key_schedule_context: 0124497637cf18d6fbcc16e9f652f00244c981726f293bb781
9861e85e50c94f0be30e022ab081e18e6f299fd3d3d976a4bc590f85bc7711bfce32ee1a
7fb1c154ef45baa1f3a4b169e141feb957e48d03f28c837d8904c3d6775308c3d3faa75d
d64adfa44e1a1141edf9349959b8f8e5291cbdc56f62b0ed6527d692e85b09a4
secret: db776cab2c066a8147d8778a456b748d8b4881cb22cebb674474c8e9bb17c123
66a1bfecd62e297f3f715ddd3e4b31772284f02c8a943d7da9fdec8c1ab6f694
key: 0cc4c3fe74377465899ed383739be3ca3a9927a2bc6cc81909adc3ce7ca7ac6d
base_nonce: 07eaf15d8bfb9104fc2b851d
exporter_secret: 4e8f1813831858a3ea10f4c088bbccb8f01a44667d97071ad04e391
b74068e045dd3cb2fa224dd4e64d76176a1541e7108db8a986a345947fff855976080f65
f
```

A.6.2.1. Encryptions

```
sequence number: 0
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d30
nonce: 07eaf15d8bfb9104fc2b851d
ciphertext: 0ead36e9a07a606bc0cd3b12baf8b6dcf279df41d68ecac35a05301c8cfb
f60e62693f92356d9bcdfbf7a19920

sequence number: 1
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d31
nonce: 07eaf15d8bfb9104fc2b851c
ciphertext: 891d0ce19337e650fc7e3442e084cfe068c2371f0052b8203b0c9baa6df2
aae1c06f697e74df612da850cef6c1

sequence number: 2
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d32
nonce: 07eaf15d8bfb9104fc2b851f
ciphertext: 3e5b960fddd28c38c906c512d11359288c1c764e88a857977991827b85ad
41d570c92661ea4b231d1e95b7f346

sequence number: 4
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d34
nonce: 07eaf15d8bfb9104fc2b8519
ciphertext: 0ccbec29295faaa05020a95204c161970d4f2c27a411d04cb8fac373e7e8
0c400cd1c6cb34cd15332085583b7b

sequence number: 255
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323535
nonce: 07eaf15d8bfb9104fc2b85e2
ciphertext: 13822dc8aabe3b76c61c771b154c39bd75af9cabb402b7232495a10c2694
742f0de6e47258ea94201f256c592d

sequence number: 256
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323536
nonce: 07eaf15d8bfb9104fc2b841d
ciphertext: 07a99443b34e8e182c881412783081513507f1b762520304b1d1599ae377
b92893522640c25444ada2245c7f7c
```

A.6.2.2. Exported Values

```
exporter_context:
L: 32
exported_value:
d9d957949e909a759151226f3fd56b83bc325383f21b37b87d42e81fa02b7887

exporter_context: 00
L: 32
exported_value:
8081a897cdfc12ee2a6b77665644348fdd52024ab6df23ba3e332b1dd410c69c

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
0316e7c7228d4bdaf0ed5eeb62e85b46b24d784711a175c377186d1807a9899f
```

A.6.3. Auth Setup Information

```
mode: 2
kem_id: 18
kdf_id: 3
aead_id: 2
info: 4f6465206f6e20612047726563696616e2055726e
ikmE: 0e5ec115c172744e793ef333abfd5d9838fd31eef595f0b77a2f5623404aa42f0e
77198c2ac78defeae75cc79cf5dc5c7587b6eda733e48f3b17d5e3496482f19cdb
pkEm: 0400a9607fbf96874b3f004e1655c497b3ccefc6c0881cbf19a58f801a572904a0
f1895c55f4874293e730909c615c7d772f0f264d3ade79628768639e09bbcf48842501c3
5aa8b265c848705c0f44325e884596266746c75a5b577013e061f9c2c894ed9b311711c7
b80bf11dc552cb8d6b1101e8fb7999e56abf3626cc07c1bb5c1fb562
skEm: 01821ed9d92632a1d419d88bdc2183659ad469bdab6b5a02ad4b0fadc8fb7f7d40
39c035a5d0dd10e74f1147bc881f38fc9b128afe24c1dd126d28dce89aca0c6bf8
ikmR: 3f3c12492eee6f1f8029a89002b6539ea9754b48412c14d2fe94368cf1fa473f01
71cdb1117aaa447ad6914bfb794ead14a9cd22cd83caa22bd905bd7d614d401ddc
pkRm: 0401045406cb181897f1dc5d5df704abfc98f1362a5d10767ab662305c994e57c6
a7778b640873feb12d1324936bc8fa9f1ef611fafb67e959fe9d150d6171d9839c9c0166
86a75d943b99294cba531457a21db66d32a857f6be1465d25ef91a3dcab2c19addca3a0b
7c0c55b2b85ab52d9e0bc7f652bb8a0035bf52070ba6eba1b1587935
skRm: 003057de4eea0c3d69d7adb681725840a6b12993d01fea25f77fec0152e9cac6e2
773d50be68d32ff8ee91e199e108dfd9d1a339b13a12cb391fbbf6002e246ea727
ikmS: 02d1581b120d0415bda1fb2cbe6a3c81edf28209f0fc1460745ad9018747c31e6d
42a36ed1734214312311db80bb3f88c2da6ea7624cf2915d32f2993ebeb92c2934
pkSm: 040130fe47be7e02492934f677133cb674d2937de86d1760312cde593813602038
83ced55deb54001b8f3ba04b208aac6debdde298df8564d2a643503474a7359126c70107
d12509f35982aadca5dfd1ba3901a9e7c5165fe6f9fdd84df6530be060d1e69456eea6b9
dee8ec4ae6e548f1f8f6f3f6a37ff30baff0566e955f6b124d402a1a
skSm: 003cca20fff091ee720d27962c16df28c00b1eedbc9ffa075cd803633480bb67b2
7eb7f51d9327582979ce2564cb30d554d3f38e20c1c2b4800654e335a16a097b65
enc: 0400a9607fbf96874b3f004e1655c497b3ccefc6c0881cbf19a58f801a572904a0f
1895c55f4874293e730909c615c7d772f0f264d3ade79628768639e09bbcf48842501c35
aa8b265c848705c0f44325e884596266746c75a5b577013e061f9c2c894ed9b311711c7b
80bf11dc552cb8d6b1101e8fb7999e56abf3626cc07c1bb5c1fb562
shared_secret: c5fb6471c9b8d5f03579664cf04ab831c1ec97265e7c3ecd9569fbdd8
86f689586668d40271e10e1ab867335be404a8d46feb76c0638b1e7338d07ab32b98a2f
key_schedule_context: 0283a27c5b2358ab4dae1b2f5d8f57f10ccccc822a473326f5
43f239a70aee46347324e84e02d7651a10d08fb3dda739d22d50c53fbfa8122baacd0f9a
e5913072ef45baa1f3a4b169e141feb957e48d03f28c837d8904c3d6775308c3d3faa75d
d64adfa44e1a1141edf9349959b8f8e5291cbdc56f62b0ed6527d692e85b09a4
secret: 4a59a771c67da727b1cd17178bcaecc3b85b8fb90298bcb7b2f55dd89143efe3
3939731b1b6817fa8332d2a1bce7f9ac3ddb4a6d320228c11498e9c8d1496def
key: bb4784ce90885ac815d16e0fe5a383f884832709081454bfaa995a2f88381cd7
base_nonce: b11485b497a8deec191e7b53
exporter_secret: ecc1e48df1e954fa0895fe596cb2aa590c7561148107d00dfbef02d
41b2a0b8b6daa7fb3d4a5697e30a736a14f35e675d090726d0105fc99c2d5214fc082e0d
a
```

A.6.3.1. Encryptions

```
sequence number: 0
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d30
nonce: b11485b497a8deec191e7b53
ciphertext: 4ee6c1e2b7fa65f5a6da8d6711d1fb88f85f633459041361cefed80d9461
052e7ed50ac68dc006b0b72e8fc9d0

sequence number: 1
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d31
nonce: b11485b497a8deec191e7b52
ciphertext: e3776a04ad5a72583da06970c6e492b16f2b6352f3f61c7b442feb75c291
ed659073595e9f5810e1250e489c4f

sequence number: 2
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d32
nonce: b11485b497a8deec191e7b51
ciphertext: 38f82aaaf1435e7a1dc716a594f359983734a150467858cb8db1745c0bba
1b87eb646dbf69f852f121b8a030fb

sequence number: 4
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d34
nonce: b11485b497a8deec191e7b57
ciphertext: 7b218dd6a38559a7b1970038018e5a680896e44f3f84c3e3def263e48a04
c70b05d6abf4210562e0c3b75927e3

sequence number: 255
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323535
nonce: b11485b497a8deec191e7bac
ciphertext: fdabf6a84e5b9f66b12ff30dff7a0b3649445467bed6fd9e67c26b7991e0
b5732421298315d67a9d52dfc7bb41

sequence number: 256
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323536
nonce: b11485b497a8deec191e7a53
ciphertext: 34226ccd6c612382fb96c5c11832073dafcfe25ce49ba8834738582b64c3
e05416470d94d0fded20a993db944d
```

A.6.3.2. Exported Values

```
exporter_context:
L: 32
exported_value:
d5541537e029f9044889b522bf0f85dfd8c31431cc75322ed2964c433b935a6c

exporter_context: 00
L: 32
exported_value:
3e12d526ad95b7648594d093f838209a6780f71d775b12a5fad16f4546332d43

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
646c979d94c8d2814d3f9f5f05e2bfc2e3045bd74ab6ca6377b377ce539c1e0a
```

A.6.4. AuthPSK Setup Information

```
mode: 3
kem_id: 18
kdf_id: 3
aead_id: 2
info: 4f6465206f6e20612047726563696616e2055726e
ikmE: c393a0a0cab3313f0526d72b8bef55b26a156493263fa1a653069ed5f2f3637f37
1b15704778c89a6e01c3fd958dfee5f85613e2d4b2eb17a152810690d8252df44b
pkEm: 04010a0d92e86189e02810958da08f1c5265cdc000ae80d96b3eb8582649a5b640
a5d1642f7d6be0d89251cdfc3c65a6eec16b2144867410440ff9f700c240872657870098
14e7d54549ea1b680c891cbf87e70b4d5b22f3dadbe092f085e09f27fd034342e3bfd98f
33094b7feb00e9e237376594a410e84ce0350d70f38217b0d466beb0
skEm: 00054bc38331e6fdef87a0569e20382d65575293b5854d705cf4af30fd6a8cf33c
39201d95a5a88f8a0d7aeb12d58323d2d700fef00aacea80d4fe6e3195b7b96782
ikmR: e7fa07c9938a3d9d883d256abd7fb5777ed7c4195a8d244c0e682fd1a08fdcce58
ab1aa7ba31b1fa120f03d9df438509a1e31992506dbc196ac4489079b3b64bea9b
pkRm: 0400cddaa5efdded3ad07444b153f0d81dc7e00adfb8e9db9433f81878b5fc1bce
46d73f209ccd9bbeb8e7666ef6e0a2579bafd19c2d2ee6a007fd2fa4fdb26e72b401012d
d6b9e771740c8356e3362ad42c35d16cbd3a9c928775d912a7cec7b0ed21cfd9e7cfe898
2790b5ecec0865db3c0d4f7fb8da6358308d6d3081b5e0a7c0ed433c
skRm: 0021b5ef4db61b4b91d3c6d88141b605c258397366222b350294463bdebdbe0036
852486810a2fd9453aec333907de2c7581d946003337c7ba584ec23cfe832bb61f
ikmS: 403e8d5ff0155c71232a887f542df221c385956eaca3acceffdc4681b99c7e9613
f90ba03b0c03c0c57a90b2dd31effc6278ab6dc059f3e5389dff6495bda12b4da4
pkSm: 04000a76a0865d1cdd11370b523d2a021b11b60f2896735b7257ebcd9810341775
ab86df3a4fbbe35788116331b88112515cfe02c551a74e2b1172f1fb275e18602db2013f
d046bbc404c4f742b663a6766b624e7b1b554bb083925c5501ebd460f0c5a2f22013c11d
2a5d8c8b4c00f676673f72a643a898b0dc063e68d04520fe4f60f202
skSm: 0071246eb324a0d343614abc4f6053ca7bed89d65b37183a14b49023826d28bcd9
2365ff317012ea0b4ce2a02af6dc4e2903aa12f7140deba1d8260a412afc12ff57
psk: 0247fd33b913760fa1fa51e1892d9f307fbe65eb171e8132c2af18555a738b82
psk_id: 456e6e796e20447572696e206172616e204d6f726961
enc: 04010a0d92e86189e02810958da08f1c5265cdc000ae80d96b3eb8582649a5b640a
5d1642f7d6be0d89251cdfc3c65a6eec16b2144867410440ff9f700c2408726578700981
4e7d54549ea1b680c891cbf87e70b4d5b22f3dadbe092f085e09f27fd034342e3bfd98f3
3094b7feb00e9e237376594a410e84ce0350d70f38217b0d466beb0
shared_secret: b4159baa2f2b745a8fb7660693c8e01bb248fce102a1b171a475c2e38
ecc4c3dbec5381817cfc0d0ec19dc007a6ea7933c00a8f9ab35adfffa7c220966c975fb
key_schedule_context: 0324497637cf18d6fbcc16e9f652f00244c981726f293bb781
9861e85e50c94f0be30e022ab081e18e6f299fd3d3d976a4bc590f85bc7711bfce32ee1a
7fb1c154ef45baa1f3a4b169e141feb957e48d03f28c837d8904c3d6775308c3d3faa75d
d64adfa44e1a1141edf9349959b8f8e5291cbdc56f62b0ed6527d692e85b09a4
secret: c53588ac939347f934b9d1aa14d20f7adbe770006b687da6a1336a8d403ea856
cedbb4553649c3adc3e1a7816ca9cb5bdff6d9ea7aaf346c8f5f1a76568ce4b5
key: b4f8d530bd73a4fd4cad439d000ab2597fcec2566b7724a9192d9570457481de
base_nonce: 15bbbca1da3014f719666d1a
exporter_secret: 0590c71d26e4d72f70abce80f1d4bc74b6a296b92c7d766671ad515
5e9265b7adb45f818177bb36b97601b30f70eca1587b690c710da90dc1b5b33436891eb9
7
```

A.6.4.1. Encryptions

```
sequence number: 0
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d30
nonce: 15bbbca1da3014f719666d1a
ciphertext: 3ec4207c64fdc69bae26e1fe16d5b6d7718ba85464aade6570d5dfe4711a
cf91b639d03515304308c6c61e9a83

sequence number: 1
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d31
nonce: 15bbbca1da3014f719666d1b
ciphertext: 2cd09c9cf81da78928ec9e2311aea6a2ac7e22354d858be8a1895c53ec36
e1fcca75eaa140d3696aac46faed74

sequence number: 2
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d32
nonce: 15bbbca1da3014f719666d18
ciphertext: 86956df21daf7224c8386316d03a8943daf5e9aa9b068a6fbcc7c767bb95
ff28c4ff68283c33fc80af6296ce9c

sequence number: 4
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d34
nonce: 15bbbca1da3014f719666d1e
ciphertext: 4553f172fcb1df84624a79a73de39bd8b5dc6922b3798c1dc2750c3b9ca0
da59fc7cd8a4223b9fb85ac9eb9fa4

sequence number: 255
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323535
nonce: 15bbbca1da3014f719666de5
ciphertext: 26f491ac3c21e6b6da7087e8f3bee2cb93f05b289b31cdba96314f502fd7
99df3dfcf8c3f1544ac3fb12c27d8a

sequence number: 256
plaintext: 4265617574792069732074727574682c2074727574682062656175747 9
aad: 436f756e742d323536
nonce: 15bbbca1da3014f719666c1a
ciphertext: 1d1b175ee55a95ec796808afaf7530eccade222e7ebffeb6aac823a01730
fce7f75e7df526880e17b6086d5413
```

A.6.4.2. Exported Values

```
exporter_context:
L: 32
exported_value:
cc48006e0b6a981ef700ba3fa7c5c1d439a9513026de4de15f09f2720bb588b3

exporter_context: 00
L: 32
exported_value:
8956126b1c0c52a6117359b42fada73e66492080adcf8bfc1bf6c07e0883d0de

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
3e336d66841cd35f9543c58e6965db4cb3caf877842d8e6b8231e15cbdb2237f
```

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: 4f6465206f6e2061204772656369616e2055726e
ikmE: 4badd64fa6444ca54f5e4fdd0228b1e79eb6c51272db080c79f7befdc4d101b6
pkEm: 6606dea00a41c3e1568e13de1144941c3054040b18afd2ba843ea80d702d9b1a
skEm: 395deaaa3990242a5451cf3e2e5e3102c4fa7608e5d5d9df4e84a1320438af2e
ikmR: 2224a63fdbf32767205865d3450ce4ecea88a761bc53acb8c1e6510f4d37a205
pkRm: c0cd6b271a1c7db71ba8e90005708c053177983bd998eccbe6eed9d8cad81d32
skRm: 212524ce7d428c8acbf38b7104cd0831a057374591a1117b61585106edb32420
enc: 6606dea00a41c3e1568e13de1144941c3054040b18afd2ba843ea80d702d9b1a
shared_secret:
7875513d8a11a18d5355ff559dc8d89a14476ed35b8e667f3d4ce2da0ec30066
key_schedule_context: 009bd09219212a8cf27c6bb5d54998c5240793a70ca0a89223
4bd5e082bc619b6a3f4c22aa6d9a0424c2b4292fdf43b8257df93c2f6adbf6ddc9c64fee
26bdd292
secret: 98219284a6ba8877d9094ddf8c7dad9780fab937433cee3d6fff84834219d354
key:
base_nonce:
exporter_secret:
3a194afc96f98ad98a3ac18f1f542d74fb7ddd102f744cdf77597df164ebac39
```

A.7.1.1. Exported Values

exporter_context:
L: 32
exported_value:
05a913ac0d4719899ecf620233911ac6be165b79629cb88612fa16d66fdffd50

exporter_context: 00
L: 32
exported_value:
982cb3624cb067a3a475e75d959c0f60f060ca15a3a6325a5e3f9b687e50e1da

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
4eaca1950d8229abe7f767b337794876419ab50b78d8f16620f87c00df77db25

A.7.2. PSK Setup Information

mode: 1
kem_id: 32
kdf_id: 1
aead_id: 65535
info: 4f6465206f6e20612047726563696616e2055726e
ikmE: 6693402e7157cceeee0885dc88d8ec08392bea50c465daceb236b13119644ad1
pkEm: 07be5897241debd4785fc3dd99846181160786900fa5d358c4ea2f9cf9b58f67
skEm: cb55b62fb17e2be38338ca52a3c69a6b03365379f2d9ec1ea2f2ef02c2cc92ed
ikmR: d3d715efdf0a22f84803cf245f4313d856b460fb595ce531622a361b1c591536
pkRm: 09522a141e2fdb82d2edef1f3b75e0246dc7c6c54e85bda45ca5916561e91861
skRm: b5c60a69ac2e0914c83f35fa01ef6a4e067aca7b7e33d31f03002a1a9da8b000
psk: 0247fd33b913760fa1fa51e1892d9f307fbe65eb171e8132c2af18555a738b82
psk_id: 456e6e796e20447572696e206172616e204d6f726961
enc: 07be5897241debd4785fc3dd99846181160786900fa5d358c4ea2f9cf9b58f67
shared_secret:
df2701f03faaab5bdeae2997171cd638507f40a90266c007392eb012a796d0f8
key_schedule_context: 01446fb1fe2632a0a338f0a85ed1f3a0ac475bdea2cd72f8c7
13b3a46ee737379a3f4c22aa6d9a0424c2b4292fdf43b8257df93c2f6adbf6ddc9c64fee
26bdd292
secret: 31baab353fea75e4118f1e97b07820f8885c180295a4a45df858bc971141f2f3
key:
base_nonce:
exporter_secret:
c484b62c5e206626dfcb3e807f1dbeee624aaac63c835019795a6654786e499b

A.7.2.1. Exported Values

exporter_context:
L: 32
exported_value:
8ca2bed1f829faaea4c3bd8b1046445e5fa16c0b061079892eefddb78842fdb0

exporter_context: 00
L: 32
exported_value:
ee258342b703e4c17ded97898f81adca8b650a2dfba42730949652edbe5a5000

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
5e76237d3039230d00986209d3c897f00083fc569a03bb850464b432d1cc9f65

A.7.3. Auth Setup Information

mode: 2
kem_id: 32
kdf_id: 1
aead_id: 65535
info: 4f6465206f6e20612047726563696616e2055726e
ikmE: 68d3d6f62651149fdd8074e05d9d3cd213ba31783924746f1b222cfdd2c8ac8f
pkEm: 0a6fd8963b85dc0313470376b2cf13cb435a4b64a16089c22dc881f631d87338
skEm: 3dde154c0925ea12eb0c7ca926dcecb1eccedc36333596d3129ec3ec26686132
ikmR: 0be0028ac1bf53a747414edbba45869f029f2745afa95226f9a7e90e4b0faa26
pkRm: dd247b4daf1e884bdf7968c75c47382b415d12a4087d46f8c98b22db36abd316
skRm: c0c40f716ee2def28e804a1d530597cb165051ffcc875c6d87d22d6aaa96c7b8
ikmS: 6c3407c148fbbdfe178227b12b19f89fd367736b018032f9b18874a9bf33fe85
pkSm: c992207c059e4926c94ead3c0626bd207d7ea33f6dc8faa764656b679b3b7b1f
skSm: 85b89c57d5bcd435cdd12fac2f62ecb1b1f0eed9ca970edc4e53839eb22fb0eb
enc: 0a6fd8963b85dc0313470376b2cf13cb435a4b64a16089c22dc881f631d87338
shared_secret:
6cd3370e74a573117c2600b29626715a5699c8f63058eb28d24ffe47f27d7d6f
key_schedule_context: 029bd09219212a8cf27c6bb5d54998c5240793a70ca0a89223
4bd5e082bc619b6a3f4c22aa6d9a0424c2b4292fdf43b8257df93c2f6adbf6ddc9c64fee
26bdd292
secret: bd85c287f80123425fc8027f005add28a02ad261c08e2767e3469ddc13b0dfb2
key:
base_nonce:
exporter_secret:
d4d6380c40c217ba10040f66f14f6008cf3f542e087a1ac8fac500e0becd4ebc

A.7.3.1. Exported Values

```
exporter_context:
L: 32
exported_value:
6f6d2d736c909732f8bd83cd7de30ee52d1882650861e3747aee2ea8a4da18e5

exporter_context: 00
L: 32
exported_value:
571c3957f45f75a76b78c8a21132a138b7e6c1f73d86e88247cfd981fe3da981

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
d9de87a7c1641b8da0f2c06a05ed651c797b5c2c0c43c81a16318782a5e0bff9
```

A.7.4. AuthPSK Setup Information

```
mode: 3
kem_id: 32
kdf_id: 1
aead_id: 65535
info: 4f6465206f6e20612047726563696616e2055726e
ikmE: cca2b6e24aef16cc966b370d43816dbcbd658a41f57b37e004bb06b67eb7b09d
pkEm: 006972416dd84e5166e93417680406e0fa51d58aa83809192c5da3ced5e7f63a
skEm: a1e558078c8cad670f26804c1b8bfcae4fc53d09782f214d9bef29115664c54e
ikmR: d14b6e1073edce2b3691a24c3bea74c79d91aec20677e1bf1f6d08cf312e11d8
pkRm: f67127ee61c836e3e69ad3a36a07d950df7c0aa0bf0094856a4223126535aa64
skRm: f70439b9a37877e235ae9401a3e5e29df66f6d0bef0695335fa833e018ca46d0
ikmS: 0af7837a79149ad70dee6cd959e8d0410cdee80898eee54863ce349db6c3885c
pkSm: fec16220ff024951bed80fd0ad775b64e8238d548abfd8ec4e00d4841f8f2d1b
skSm: a5cf8d3b9920c1110f2ed7f40f1d33e25943c341db44b242db55f5a21639f565
psk: 0247fd33b913760fa1fa51e1892d9f307fbe65eb171e8132c2af18555a738b82
psk_id: 456e6e796e20447572696e206172616e204d6f726961
enc: 006972416dd84e5166e93417680406e0fa51d58aa83809192c5da3ced5e7f63a
shared_secret:
2a07de6b76201e3944a053529660699b1f1e14fdd0e2184e4bd6d0357e41e083
key_schedule_context: 03446fb1fe2632a0a338f0a85ed1f3a0ac475bdea2cd72f8c7
13b3a46ee737379a3f4c22aa6d9a0424c2b4292fdf43b8257df93c2f6adbf6ddc9c64fee
26bdd292
secret: 116e9c5d9d1be4e9003007cb1cca3b9d54a4f757de88148a1de2fb027575401d
key:
base_nonce:
exporter_secret:
9153dafda84187ead5d0c8d05a3aea9e7aa53e548424604fcbdb69fd70561b13
```

A.7.4.1. Exported Values

```
exporter_context:
L: 32
exported_value:
4b644b9638557fa11a0a7219048407f4aa76f875518e0d275ccb75e099906dbc

exporter_context: 00
L: 32
exported_value:
2ac0e6dc2b74c4f964ec8760fbab357a85cda30d859878721cb7f7ba0398c40d

exporter_context: 54657374436f6e74657874
L: 32
exported_value:
4de53bb53c95c4f1c7e60aaf42d9154e2d9e8cf33f494993838950a47329babd
```

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