Network Working Group
Internet-Draft
Intended status: Informational
Expires: May 7, 2020

R. Barnes
Cisco
K. Bhargavan
Inria

November 04, 2019

Hybrid Public Key Encryption
draft-irtf-cfrg-hpke-02

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.

Table of Contents

1. Introduction

"Hybrid" public-key encryption schemes (HPKE) that combine asymmetric and symmetric algorithms are a substantially more efficient solution than traditional public key encryption techniques such as those based on RSA or ElGamal. Encrypted messages convey a single ciphertext and authentication tag alongside a short public key, which may be further compressed. The key size and computational complexity of elliptic curve cryptographic primitives for authenticated encryption therefore make it compelling for a variety of use cases. This type of public key encryption has many applications in practice, for example:

- PGP [RFC6637]
- Messaging Layer Security [I-D.ietf-mls-protocol]
- Encrypted Server Name Indication [I-D.ietf-tls-esni]
- Protection of 5G subscriber identities [fiveG]

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 All of these existing schemes have problems, e.g., because they rely on outdated primitives, lack proofs of IND-CCA2 security, or fail to provide test vectors.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

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 BCP14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Security Properties

As a hybrid authentication encryption algorithm, we desire security against (adaptive) chosen ciphertext attacks (IND-CCA2 secure). The HPKE variants described in this document achieve this property under the Random Oracle model assuming the gap Computational Diffie Hellman (CDH) problem is hard [S01].

[[TODO - Provide citations to these proofs once they exist]]

4. Notation

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

- Initiator (I): Sender of an encrypted message.
- Responder (R): Receiver of an encrypted message.
- Ephemeral (E): A fresh random value meant for one-time use.
- "(skX, pkX)": A KEM key pair used in role X; "skX" is the private key and "pkX" is the public key
- "pk(skX)": The public key corresponding to private key "skX"
- "len(x)": The length of the octet string "x", expressed as a two-octet unsigned integer in network (big-endian) byte order
- "encode_big_endian(x, n)": An octet string encoding the integer value "x" as an n-byte big-endian value
- "concat(x0, ..., xN)": Concatenation of octet strings. "concat(0x01, 0x0203, 0x040506) = 0x010203040506"
- "zero(n)": An all-zero octet string of length "n". "zero(4) = 0x00000000"

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

5. Cryptographic Dependencies

HPKE variants rely on the following primitives:

- A Key Encapsulation Mechanism (KEM):
  - GenerateKeyPair(): Generate a key pair (sk, pk)
  - Marshal(pk): Produce a fixed-length octet string encoding the public key "pk"
  - Unmarshal(enc): Parse a fixed-length octet string to recover a public key
  - Encap(pk): Generate an ephemeral, fixed-length symmetric key and a fixed-length encapsulation of that key that can be decapsulated by the holder of the private key corresponding to pk
  - Decap(enc, sk): Use the private key "sk" to recover the ephemeral symmetric key from its encapsulated representation "enc"
  - AuthEncap(pkR, skI) (optional): Same as Encap(), but the outputs encode an assurance that the ephemeral shared key is known only to the holder of the private key "skI"
  - AuthDecap(skR, pkI) (optional): Same as Decap(), but the holder of the private key "skR" is assured that the ephemeral shared key is known only to the holder of the private key corresponding to "pkI"
  - Nenc: The length in octets of an encapsulated key from this KEM
  - Npk: The length in octets of a public key for this KEM
- A Key Derivation Function:
  - Hash(m): Compute the cryptographic hash of input message "m"
  - Extract(salt, IKM): Extract a pseudorandom key of fixed length from input keying material "IKM" and an optional octet 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 Hash and Extract functions

- An AEAD encryption algorithm [RFC5116]:

  * Seal(key, nonce, aad, pt): Encrypt and authenticate plaintext "pt" with associated data "aad" using secret key "key" and nonce "nonce", yielding ciphertext and tag "ct"

  * Open(key, nonce, aad, ct): Decrypt ciphertext "ct" using associated data "aad" with secret key "key" and nonce "nonce", returning plaintext message "pt" or the error value "OpenError"

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

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

A set of algorithm identifiers for concrete instantiations of these primitives is provided in Section 8. Algorithm identifier values are two octets long.

5.1. DH-Based KEM

Suppose we are given a Diffie-Hellman group that provides the following operations:

- GenerateKeyPair(): Generate an ephemeral key pair "(sk, pk)" for the DH group in use

- DH(sk, pk): Perform a non-interactive DH exchange using the private key sk and public key pk to produce a fixed-length shared secret

- Marshal(pk): Produce a fixed-length octet string encoding the public key "pk"

- Unmarshal(enc): Parse a fixed-length octet string to recover a public key

Then we can construct a KEM (which we'll call "DHKEM") in the following way:

```
def Encap(pkR):
  skE, pkE = GenerateKeyPair()
  zz = DH(skE, pkR)
  enc = Marshal(pkE)
  return zz, enc

def Decap(enc, skR):
  pkE = Unmarshal(enc)
  return DH(skR, pkE)

def AuthEncap(pkR, skI):
  skE, pkE = GenerateKeyPair()
  zz = concat(DH(skE, pkR), DH(skI, pkR))
  enc = Marshal(pkE)
  return zz, enc

def AuthDecap(enc, skR, pkI):
  pkE = Unmarshal(enc)
  return concat(DH(skR, pkE), DH(skR, pkI))
```

The GenerateKeyPair, Marshal, and Unmarshal functions are the same as for the underlying DH group. The Marshal functions for the curves referenced in {#ciphersuites} are as follows:

- P-256: The X-coordinate of the point, encoded as a 32-octet big-endian integer

- P-521: The X-coordinate of the point, encoded as a 66-octet big-endian integer

- Curve25519: The standard 32-octet representation of the public key

- Curve448: The standard 56-octet representation of the public key

6. 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 the private key corresponding to "pkR" can decapsulate the key from "enc" and decrypt the ciphertexts. All of 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 two authenticated variants, one of which authenticates

possession of a pre-shared key, and one of which authenticates possession of a KEM private key. The following one-octet values will be used to distinguish between modes:

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

All of 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.

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

6.1. 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:

- "pkR" - The receiver's public key

- "zz" - A shared secret generated via the KEM for this transaction

- "enc" - An encapsulated key produced by the KEM for the receiver

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

- "psk" - A pre-shared secret held by both the initiator and the receiver (optional; default value "zero(Nh)").

- "pskID" - An identifier for the PSK (optional; default value """ = zero(0)"

- "pkI" - The initiator's public key (optional; default value "zero(Npk)")

The "psk" and "pskID" 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.

The key and nonce computed by this algorithm have the property that they are only known to the holder of the receipient private key, and the party that ran the KEM to generate "zz" and "enc". If the "psk" and "pskID" arguments are provided, then the recipient is assured that the initiator held the PSK. If the "pkIm" argument is provided, then the recipient is assued that the initator held the corresponding private key (assuming that "zz" and "enc" were generated using the AuthEncap / AuthDecap methods; see below).

The HPKE algorithm identifiers, i.e., the KEM "kem_id", KDF "kdf_id", and AEAD "aead_id" 2-octet code points, are assumed implicit from the implementation and not passed as parameters.

```
default_pkIm = zero(Npk)
default_psk = zero(Nh)
default_pskID = zero(0)

def VerifyMode(mode, psk, pskID, pkIm):
  got_psk = (psk != default_psk and pskID != default_pskID)
  no_psk = (psk == default_psk and pskID == default_pskID)
  got_pkIm = (pkIm != default_pkIm)
  no_pkIm = (pkIm == default_pkIm)

  if mode == mode_base and (got_psk or got_pkIm):
    raise Exception("Invalid configuration for mode_base")
  if mode == mode_psk and (no_psk or got_pkIm):
    raise Exception("Invalid configuration for mode_psk")
  if mode == mode_auth and (got_psk or no_pkIm):
    raise Exception("Invalid configuration for mode_auth")
  if mode == mode_psk_auth and (no_psk or no_pkIm):
    raise Exception("Invalid configuration for mode_psk_auth")

def KeySchedule(mode, pkRm, zz, enc, info, psk, pskID, pkIm):
  VerifyMode(mode, psk, pskID, pkI)

  pkRm = Marshal(pkR)
  ciphersuite = concat(kem_id, kdf_id, aead_id)
  pskID_hash = Hash(pskID)
  info_hash = Hash(info)
  context = concat(mode, ciphersuite, enc, pkRm, pkIm, pskID_hash, info_hash)

  secret = Extract(psk, zz)
  key = Expand(secret, concat("hpke key", context), Nk)
  nonce = Expand(secret, concat("hpke nonce", context), Nn)
  return Context(key, nonce)
```

Note that the context construction in the KeySchedule procedure is equivalent to serializing a structure of the following form in the TLS presentation syntax:

```
struct {
    // Mode and algorithms
    uint8 mode;
    uint16 kem_id;
    uint16 kdf_id;
    uint16 aead_id;

    // Public inputs to this key exchange
    opaque enc[Nenc];
    opaque pkR[Npk];
    opaque pkI[Npk];

    // Cryptographic hash of application-supplied pskID
    opaque pskID_hash[Nh];

    // Cryptographic hash of application-supplied info
    opaque info_hash[Nh];
} HPKEContext;
```

6.2. Encryption to a Public Key

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

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

```
def SetupBaseI(pkR, info):
  zz, enc = Encap(pkR)
  return enc, KeySchedule(mode_base, pkR, zz, enc, info,
                          default_psk, default_pskID, default_pkIm)

def SetupBaseR(enc, skR, info):
  zz = Decap(enc, skR)
  return KeySchedule(mode_base, pk(skR), zz, enc, info,
                     default_psk, default_pskID, default_pkIm)
```

6.3. Authentication using a Pre-Shared Key

This variant extends the base mechansism by allowing the recipient to authenticate that the sender possessed a given pre-shared key (PSK). We assume that both parties have been provisioned with both the PSK value "psk" and another octet string "pskID" that is used to identify which PSK should be used.

The primary differences from the base case are:

- The PSK is used as the "salt" input to the KDF (instead of 0)
- The PSK ID is added to the context string used as the "info" input to the KDF

This mechanism is not suitable for use with a low-entropy password as the PSK. A malicious recipient that does not possess the PSK can use decryption of a plaintext as an oracle for performing offline dictionary attacks.

```
def SetupPSKI(pkR, info, psk, pskID):
  zz, enc = Encap(pkR)
  return enc, KeySchedule(mode_psk, pkR, zz, enc, info,
                          psk, pskId, default_pkIm)

def SetupPSKR(enc, skR, info, psk, pskID):
  zz = Decap(enc, skR)
  return KeySchedule(mode_psk, pk(skR), zz, enc, info,
                     psk, pskId, default_pkIm)
```

6.4. Authentication using an Asymmetric Key

This variant extends the base mechansism 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, pkI)" produces the correct shared secret only if the encapsulated value "enc" was produced by "AuthEncap(pkR, skI)", where "skI" is the private key corresponding to "pkI". In other words, only two people could have produced this secret, so if the recipient is one, then the sender must be the other.

The primary differences from the base case are:

- The calls to "Encap" and "Decap" are replaced with calls to "AuthEncap" and "AuthDecap".
- The initiator public key is added to the context string

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

This mechanism authenticates only the key pair of the initiator, not any other identity. If an application wishes to authenticate some other identity for the sender (e.g., an email address or domain name), then this identity should be included in the "info" parameter to avoid unknown key share attacks.

```
def SetupAuthI(pkR, info, skI):
  zz, enc = AuthEncap(pkR, skI)
  pkIm = Marshal(pk(skI))
  return enc, KeySchedule(mode_auth, pkR, zz, enc, info,
                          default_psk, default_pskID, pkIm)

def SetupAuthR(enc, skR, info, pkI):
  zz = AuthDecap(enc, skR, pkI)
  pkIm = Marshal(pkI)
  return KeySchedule(mode_auth, pk(skR), zz, enc, info,
                     default_psk, default_pskID, pkIm)
```

6.5. 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 SetupAuthPSKI(pkR, info, psk, pskID, skI):
  zz, enc = AuthEncap(pkR, skI)
  pkIm = Marshal(pk(skI))
  return enc, KeySchedule(mode_psk_auth, pkR, zz, enc, info,
                          psk, pskID, pkIm)

def SetupAuthPSKR(enc, skR, info, psk, pskID, pkI):
  zz = AuthDecap(enc, skR, pkI)
  pkIm = Marshal(pkI)
  return KeySchedule(mode_psk_auth, pk(skR), zz, enc, info,
                     psk, pskID, pkIm)
```

6.6. 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 decryption must be stateful. Each of the setup procedures above produces a context object that stores the required state:

- The AEAD algorithm in use
- The key to be used with the AEAD algorithm
- A base nonce value

- A sequence number (initially 0)

All of these fields except the sequence number are constant. The sequence number is used to provide nonce uniqueness: The nonce used for each encryption or decryption operation is the result of XORing the base nonce with the current sequence number, encoded as a big-endian integer of the same length as the nonce. Implementations MAY use a sequence number that is shorter than the nonce (padding on the left with zero), but MUST return an error if the sequence number overflows.

Each encryption or decryption operation increments the sequence number for the context in use. A given context SHOULD be used either only for encryption or only for decryption.

It is up to the application to ensure that encryptions and decryptions are done in the proper sequence, so that the nonce values used for encryption and decryption line up. If a Seal or Open operation would cause the "seq" field to wrap, then the implementation MUST return an error.

```
def Context.Nonce(seq):
  encSeq = encode_big_endian(seq, len(self.nonce))
  return xor(self.nonce, encSeq)

def Context.IncrementSeq():
  if self.seq >= (1 << Nn) - 1:
    return NonceOverflowError
  self.seq += 1

def Context.Seal(aad, pt):
  ct = Seal(self.key, self.Nonce(self.seq), aad, pt)
  self.IncrementSeq()
  return ct

def Context.Open(aad, ct):
  pt = Open(self.key, self.Nonce(self.seq), aad, ct)
  if pt == OpenError:
    return OpenError
  self.IncrementSeq()
  return pt
```

7. Single-Shot APIs

In many cases, applications encrypt only a single message to a recipient's public key. This section provides templates for HPKE APIs that implement "single-shot" encryption and decryption using APIs specified in Section 6.2 and Section 6.6:

```
def Seal<MODE>(pkR, info, aad, pt, ...):
  enc, ctx = Setup<MODE>I(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. Thus, SealAuthPSK and OpenAuthPSK would be implemented as follows:

```
def SealAuthPSK(pkR, info, aad, pt, psk, pskID, skI):
  enc, ctx = SetupAuthPSKI(pkR, info, psk, pskID, skI)
  ct = ctx.Seal(aad, pt)
  return enc, ct

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

8. Algorithm Identifiers

8.1. Key Encapsulation Mechanisms (KEMs)

Value	KEM	Nenc	Npk	Reference
0x0000	(reserved)	N/A	N/A	N/A
0x0001	DHKEM(P-256)	32	32	[NISTCurves]
0x0002	DHKEM(Curve25519)	32	32	[RFC7748]
0x0003	DHKEM(P-521)	65	65	[NISTCurves]
0x0004	DHKEM(Curve448)	56	56	[RFC7748]

For the NIST curves P-256 and P-521, the Marshal function of the DH scheme produces the normal (non-compressed) representation of the public key, according to [SECG]. When these curves are used, the recipient of an HPKE ciphertext MUST validate that the ephemeral public key "pkE" is on the curve. The relevant validation procedures are defined in [keyagreement]

For the CFRG curves Curve25519 and Curve448, the Marshal function is the identity function, since these curves already use fixed-length octet strings for public keys.

8.2. Key Derivation Functions (KDFs)

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

8.3. Authenticated Encryption with Associated Data (AEAD) Functions

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

9. Security Considerations

The general security properties of HPKE are described in Section 3. In this section, we consider a security issue that may arise in practice and an advanced use case.

9.1. Metadata Protection

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

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 "(pskID, pkI, enc, ciphertext)" before, it would now send (enc2, ciphertext2, enc, ciphertext), where "(enc2, ciphertext2)" represent the encryption of the "pskID" and "pkI" values.

The cost of this approach is an additional KEM operation each for the sender and the receiver. 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.

9.2. Designated-Verifier Signature

The Auth and AuthPSK modes HPKE can be used to construct a lightweight "designated-verifier signature" scheme [JKR96], in the sense that the message is authenticated as coming from the initiator, but the only party who can verify the authentication is the receiver (the holder of "skR").

To create such a signature, the initator simply performs a normal HPKE setup in the proper mode, and calls the Seal method on the resulting context with an empty plaintext value and the content to be signed as AAD. This produces an encoded key "enc" and a ciphertext value that contains only the AAD tag.

For example, using DHKEM-X25519 and AES-128-GCM, this would produce a 48-byte signature comprising a 32-byte ephemeral X25519 key and a 16-byte GCM tag.

To verify such a signature, the receiver performs the corresponding HPKE setup and calls Open with the provided ciphertext. If the AEAD authentication passes, then the signature is valid.

This scheme re-uses the authentication scheme underlying the AEAD algorithm in use, while using the KEM to establish a one-time authentication key from a pair of KEM public keys.

10. Message Encoding

This document does not specify a wire format encoding for HPKE messages. Applications that adopt HPKE must therefore specify an unambiguous encoding mechanism which includes, minimally: the encapsulated value "enc", ciphertext value(s) (and order if there are multiple), and any info values that are not implicit.

11. IANA Considerations

This document requests the creation of three new IANA registries:

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

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

11.1. KEM Identifiers

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

Template:

- Value: The two-byte identifier for the algorithm
- KEM: The name of the algorithm
- Nenc: The length in bytes of an encapsulated key produced by the algorithm
- Npk: The length in bytes of a public key for the algorithm
- Reference: Where this algorithm is defined

Initial contents: Provided in Section 8.1

11.2. KDF Identifiers

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

Template:

- Value: The two-byte identifier for the algorithm
- KDF: The name of the algorithm
- Nh: The length in bytes of the output of the KDF

- Reference: Where this algorithm is defined

Initial contents: Provided in Section 8.2

11.3. AEAD Identifiers

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

Template:

- Value: The two-byte identifier for the algorithm
- AEAD: The name of the algorithm
- Nk: The length in bytes of a key for this algorithm
- Nn: The length in bytes of a nonce for this algorithm
- Reference: Where this algorithm is defined

Initial contents: Provided in Section 8.3

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>.

[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

[ANSI] "Public Key Cryptography for the Financial Services Industry -- Key Agreement and Key Transport Using Elliptic Curve Cryptography", n.d..

[BNT19] "Nonces Are Noticed: AEAD Revisited", n.d., <http://dx.doi.org/10.1007/978-3-030-26948-7_9>.

[fiveG] "Security architecture and procedures for 5G System", n.d., <https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3169>.

[GCM] Dworkin, M., "Recommendation for block cipher modes of operation :", National Institute of Standards and Technology report, DOI 10.6028/nist.sp.800-38d, 2007.

[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", draft-ietf-mls-protocol-07 (work in progress), July 2019.

[I-D.ietf-tls-esni] Rescorla, E., Oku, K., Sullivan, N., and C. Wood, "Encrypted Server Name Indication for TLS 1.3", draft-ietf-tls-esni-04 (work in progress), July 2019.

[IEEE] "IEEE 1363a, Standard Specifications for Public Key Cryptography - Amendment 1 -- Additional Techniques", n.d..

[ISO] "ISO/IEC 18033-2, Information Technology - Security Techniques - Encryption Algorithms - Part 2 -- Asymmetric Ciphers", n.d..

[JKR96] "Designated Verifier Proofs and Their Applications", n.d., <https://doi.org/10.1007%2F3-540-49677-7_30>.

[keyagreement] Barker, E., Chen, L., Roginsky, A., and M. Smid, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography", National Institute of Standards and Technology report, DOI 10.6028/nist.sp.800-56ar2, May 2013.

[MAEA10] "A Comparison of the Standardized Versions of ECIES", n.d., <http://sceweb.sce.uhcl.edu/yang/teaching/csci5234WebSecurityFall2011/Chaum-blind-signatures.PDF>.

[NISTCurves] "Digital Signature Standard (DSS)", National Institute of Standards and Technology report, DOI 10.6028/nist.fips.186-4, July 2013.

[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>.

[RFC6637] Jivsov, A., "Elliptic Curve Cryptography (ECC) in OpenPGP", RFC 6637, DOI 10.17487/RFC6637, June 2012, <https://www.rfc-editor.org/info/rfc6637>.

[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>.

[S01] "A Proposal for an ISO Standard for Public Key Encryption (verison 2.1)", n.d., <http://www.shoup.net/papers/iso-2_1.pdf>.

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

[TestVectors] "HPKE Test Vectors", n.d., <https://github.com/cfrg/draft-irtf-cfrg-hpke/blob/1e98830311b27f9af00787c16e2c5ac43abeadfb/test-vectors.json>.

Appendix A. Test Vectors

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

A.1. DHKEM(Curve25519), HKDF-SHA256, AES-GCM-128

A.1.1. AuthPSK Setup Information

```
mode: 3
kemID: 2
kdfID: 1
aeadID: 1
info: 4f6465206f6e20612047726563696616e2055726e
skR: 2d7c739195ba102216de162f9435991aa3ad42aeefdb7e22391ae34bae7e5a13
skI: 59c77f5734aef369f30d83c7e30c6bf372e120391cdaf13f34c915030284b75d
skE: 6827bbf4f7ebd0182a8ebc0ea364c7ddae1d1c8a4f58a903fa9f9f9d4228b126
psk: 6d656c6c6f6e
pskID: 456e6e796e20447572696e206172616e204d6f726961
pkR: cc980df06e532bdb6b957f9f5a5caf55c55f46822cdfbd97e76f6ad4c62b322b
pkI: db6ee4a53276b7bc90657cdde514f948af83c140540797ec717881490afed921
pkE: bc09d66a6e8a77ce2fe3bf6603f227d5c673f5329a3c9ad031bbdfadbc9b1d28
enc: bc09d66a6e8a77ce2fe3bf6603f227d5c673f5329a3c9ad031bbdfadbc9b1d28
zz: fb907aabc5e9e03f9665c937606c46d8da4932380d297a35e0c6aa3ff641ff349695
5ffd7f908cd9f8a476cd230de614d60fa4bdf599ca238580cccd7a7e7b7f
context: 03000200010001bc09d66a6e8a77ce2fe3bf6603f227d5c673f5329a3c9ad03
1bbdfadbc9b1d28cc980df06e532bdb6b957f9f5a5caf55c55f46822cdfbd97e76f6ad4c
62b322bdb6ee4a53276b7bc90657cdde514f948af83c140540797ec717881490afed921e
ca994d516108a16db86e155390f3c3cec6f0aff60ade1ae9e3189140b0f3dea55c404062
9c64c5efec2f7230407d612d16289d7c5d7afcf9340280abd2de1ab
secret: 5980d041d0343ae0ee09932c03ea7c3e383f30fd55ef4d66c7459e01a78683ea
key: 958a60f49b2b9ee8addb9ed96e4fd4fb
nonce: 52b1a515904660435ef1feac
```

A.1.1.1. Encryptions

```
sequence number: 0
plaintext: 4265617574792069732074727574682c20747275746820626561757479
aad: 436f756e742d30
ciphertext: 08ff327654d2696724eaf57b3899299ee51412ecafb3435f3cd5f31698f5
2003b0487aa0b3182d237973f3344c

sequence number: 1
plaintext: 4265617574792069732074727574682c20747275746820626561757479
aad: 436f756e742d31
ciphertext: afd8ddcb4329cb3e1fc8a46d3900eb34dcf29b6fc9e293d4ca3c59fd6f40
90ced7aef880d54d2a11922dc2134a

sequence number: 2
plaintext: 4265617574792069732074727574682c20747275746820626561757479
aad: 436f756e742d32
ciphertext: 1c9ecb83737a1723d22ddd0bd3827b549128035667c58b035e2026d51d04
0191e6b3c6c91919b80cb879e9177d

sequence number: 4
plaintext: 4265617574792069732074727574682c20747275746820626561757479
aad: 436f756e742d34
ciphertext: f26b7df1e08ba8d5d6702c66a5f6c9801ed051211dc0f1aad9169e17e149
22f7b5319e3830ab9c5bf2e9ed8c88
```

A.2. DHKEM(Curve25519), HKDF-SHA256, AES-GCM-128

A.2.1. Base Setup Information

mode: 0
kemID: 2
kdfID: 1
aeadID: 1
info: 4f6465206f6e20612047726563696616e2055726e
skR: 139c8d38df6d8dfa2c8de98af621667c76c3f63f65c7c3966c4258c316f05033
skI: dd5f9525b2fa94f21b7237ade72006a76f612dabf020a02527fcb75db6bebe6f
skE: 7227b3fee6b8e01b293b826ec8671b69894a1142981bb1513afa42819f2a22ef
psk: 6d656c6c6f6e
pskID: 456e6e796e20447572696e206172616e204d6f726961
pkR: 186c394e175b7b161760b1bd5b822a0804bd066b170c695c0df123176fa7df6f
pkI: 55f618acd854b99bc5167b72c6fbaa70056b6b30d709768658468e830c62dd5d
pkE: ef0bf7ee58713568663204cf720cff64a852c77ace25f478cfe7dc0721508e03
enc: ef0bf7ee58713568663204cf720cff64a852c77ace25f478cfe7dc0721508e03
zz: e57ee01e09664418fcb3dba12ad3d1fae7a28f733e5979d6e9e7a8b11ced503d
context: 00000200010001ef0bf7ee58713568663204cf720cff64a852c77ace25f478c
fe7dc0721508e03186c394e175b7b161760b1bd5b822a0804bd066b170c695c0df123176
fa7df6f0000000000000000000000000000000000000000000000000000000000000000e
3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b85555c404062
9c64c5efec2f7230407d612d16289d7c5d7afcf9340280abd2de1ab
secret: 42d03b2f22bd9da609ec3455497ce2eb71a5d02b0b456e5183674d14d41ca39e
key: 91183d395d6ed690ed165a26ec9d815c
nonce: 79b082c2bba5fc13d6d29fc7

A.2.1.1. Encryptions

```
sequence number: 0
plaintext: 4265617574792069732074727574682c2074727574820626561757479
aad: 436f756e742d30
ciphertext: 25002bf75aaf65a173b0fa98e1c73e5ee1b88f1c88e9870c9867e46b2904
3f73e15fe8c035715f0b10635e376d

sequence number: 1
plaintext: 4265617574792069732074727574682c2074727574820626561757479
aad: 436f756e742d31
ciphertext: bbcb5528d0e696fd8ed4f76f326c81063aa1eca92968a8445dbf28190c18
c970a6a797e529541130299334fd37

sequence number: 2
plaintext: 4265617574792069732074727574682c2074727574820626561757479
aad: 436f756e742d32
ciphertext: fcb24627efbd6c3b08b2d16567ef0883faebe867fc0eb20b7070717279a9
febe4c240251bed8cb20d9f10603e7

sequence number: 4
plaintext: 4265617574792069732074727574682c2074727574820626561757479
aad: 436f756e742d34
ciphertext: 93b1b27a5727eb806f9c2ea7f99db94682e620caf64b2cea54fee79c123c
b31f11d1c04be867599b2592a5447d
```

A.3. DHKEM(Curve25519), HKDF-SHA256, AES-GCM-128

A.3.1. PSK Setup Information

```
mode: 1
kemID: 2
kdfID: 1
aeadID: 1
info: 4f6465206f6e20612047726563696616e2055726e
skR: fdf4507c4507ececd2b2232a761ce2cdb50d993b6216e6cb9538a1fc53baed02
skI: 01f60c92806f04327cee19d1822f78a8e88fc0e6d1cada7cd94f902511bb2457
skE: 0dc7bf88c1849fd7800f8bd52eed75f26160a3cb9d634190c48ced1daa4c341a
psk: 6d656c6c6f6e
pskID: 456e6e796e20447572696e206172616e204d6f726961
pkR: 696977f50e929c99287852ded3fc9b2709f8cd51c0d270c45c338dece959f437
pkI: 5bbcd7531308ffc2653b89ad79d6c9df64bd0c981536c08e887f147b8f92d16a
pkE: ff8ad2005cae6857bf12f1a3e34b1e43dcd6b34c4685f2473d09d6595c11634f
enc: ff8ad2005cae6857bf12f1a3e34b1e43dcd6b34c4685f2473d09d6595c11634f
zz: 9839c5431904c86ff5b6f24ed3f78c10fe89ddaa4f8a45d87f80cba69c3e6c6f
context: 01000200010001ff8ad2005cae6857bf12f1a3e34b1e43dcd6b34c4685f2473
d09d6595c11634f696977f50e929c99287852ded3fc9b2709f8cd51c0d270c45c338dece
959f437000000000000000000000000000000000000000000000000000000000000000e
ca994d516108a16db86e155390f3c3cec6f0aff60ade1ae9e3189140b0f3dea55c404062
9c64c5efec2f7230407d612d16289d7c5d7afcf9340280abd2de1ab
secret: 08cfc4eadd1b6652df9cc2e5d022f5ff095c869d9ac9b66abefe2bf42cefa9e4
key: 3beaafd348e6124cad09a017506a1d27
nonce: e366f7d523287dcee75ee15b
```

A.3.1.1. Encryptions

```
sequence number: 0
plaintext: 4265617574792069732074727574682c2074727574820626561757479
aad: 436f756e742d30
ciphertext: c8652f1d6efd1331352b5c5dc2bacf69ae8a591cb90cd48a7c9aac75fc0c
99dafee6dca1839f2fd0a512ba3a1d

sequence number: 1
plaintext: 4265617574792069732074727574682c2074727574820626561757479
aad: 436f756e742d31
ciphertext: 0e67a63aa0cd0791ee3fc05a948c6dc1341a0699afcd074e0c68e1a64f6d
d5c938492583e8be4b9930c0f9da5c

sequence number: 2
plaintext: 4265617574792069732074727574682c2074727574820626561757479
aad: 436f756e742d32
ciphertext: df7a3586fbf5d438320c4ad17263431624a784ebcabdc534aaf9085d068d
6bed7e33f5978eec27d0d93147ec41

sequence number: 4
plaintext: 4265617574792069732074727574682c2074727574820626561757479
aad: 436f756e742d34
ciphertext: 2653ca0b51c544df3ef474c6ec057d4659547869caccb26ada2c7deb0324
3a04e7e647c301c7ac6eba1d45eb1b
```

A.4. DHKEM(Curve25519), HKDF-SHA256, AES-GCM-128

A.4.1. Auth Setup Information

```
mode: 2
kemID: 2
kdfID: 1
aeadID: 1
info: 4f6465206f6e20612047726563696616e2055726e
skR: a3f26ded71d69e6b7c4924e5756388efce4c8857b89cd00e492cd5a5778c40d7
skI: 42962c361fe2e92343113407b95abe4eebef311e315726c579fc6fb7f22e16c5
skE: 372c79a960e1c33c6b6f69c7f4c63567f743a018d5d04fc4a243e0af2aeb3708
psk: 6d656c6c6f6e
pskID: 456e6e796e20447572696e206172616e204d6f726961
pkR: 39c07ff4a258dfe61a1e9e23ef1d9d39efce89c5326da949a9dfa3b11b9f2f2e
pkI: 9888840b794623840275e594ec76a526e690b9111d21cdec3fe2faf18227f676
pkE: 617c577bb03e3b6423efae1e4da7e9b82f690ef4d312af6c68a6b78a0977a550
enc: 617c577bb03e3b6423efae1e4da7e9b82f690ef4d312af6c68a6b78a0977a550
zz: 47bdf821a19dfff3f1c868fcaccef934541076ef3cb90a215f4aa9c55512ff738d50
64e12a9f0f016254be7cef5d7cecf2fc481b338794dc10379421229d815b
context: 02000200010001617c577bb03e3b6423efae1e4da7e9b82f690ef4d312af6c6
8a6b78a0977a55039c07ff4a258dfe61a1e9e23ef1d9d39efce89c5326da949a9dfa3b11
b9f2f2e9888840b794623840275e594ec76a526e690b9111d21cdec3fe2faf18227f676e
3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b85555c404062
9c64c5efec2f7230407d612d16289d7c5d7afcf9340280abd2de1ab
secret: 32b18835e15daaa76a1ff0b8a04000fa1e1c735a033322c12b4ab1446e0d9a40
key: 71b005dc14a7446c512af54fc5fc5e44
nonce: 75f22f1765b8349fafa53bdc
```

A.4.1.1. Encryptions

```
sequence number: 0
plaintext: 4265617574792069732074727574682c2074727574820626561757479
aad: 436f756e742d30
ciphertext: 5bcb5fb904f4af215155fe742ec23028a07a1b87fd61017a6bcb66959892
6bafdb88d577d95f682c24f6148e5f

sequence number: 1
plaintext: 4265617574792069732074727574682c2074727574820626561757479
aad: 436f756e742d31
ciphertext: b0d8feae5b28a09e6cdec173e6f0a18184cb2b9cc56614300a684622ef3e
baf058df215141839f7bd7d6a639d9

sequence number: 2
plaintext: 4265617574792069732074727574682c2074727574820626561757479
aad: 436f756e742d32
ciphertext: d62e75b44c04c37b25cd026935920dc842196500a34922ff7db97a9576a2
3a415f5edaf753402080ed77ff0d6c

sequence number: 4
plaintext: 4265617574792069732074727574682c2074727574820626561757479
aad: 436f756e742d34
ciphertext: 54b8d9c1e2ebf0a78c8f64d11e19e95d5a5ee993da38c144f9aca3f39e49
96c23cef432d6b5a902f1de404e5e1
```

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

A.5.1. Auth Setup Information

```
mode: 2
kemID: 1
kdfID: 1
aeadID: 3
info: 4f6465206f6e20612047726563696616e2055726e
skR: 0bfbc9b95d6bb528019f8852c67dd2b1e449796cd5ea9d0a4d875ba105c0998b
skI: 03ca6aa924d700e2ae2e4347e658e05e1543c8cb580683c155ac30bfbe56cb07
skE: 4a0fe976e4ef4f3c835739775549689bac1d9e815a84706bc648c02a8ea475e8
psk: 6d656c6c6f6e
pskID: 456e6e796e20447572696e206172616e204d6f726961
pkR: 04b09c11d56c7ce347cb9d45c8b0d8e7cfeed5792e2dd52430bf79908b98641ec91
b5b4468376c6e797a0b5d9ff440de0b03d4ab03d69e4a5ab46623b92b7a45d8
pkI: 0465df804f772b75d8349296932f59a17215ab58cb5d1fb3f530840bd2a3a946181
2706edcdada222a42288ca90e8b9c04560c5acc3f89120f7291bfe5d010e0c1
pkE: 04b619b60048a5cdd14aa8e5299e2439259f87b92ad890105cd1c5cf191ecb5520b
de19ced20f4e2bced384351ee4128e3e7147775cf44f33d8623f7b40dee58d6
enc: 04b619b60048a5cdd14aa8e5299e2439259f87b92ad890105cd1c5cf191ecb5520b
de19ced20f4e2bced384351ee4128e3e7147775cf44f33d8623f7b40dee58d6
zz: 18bd7013cc5c7accacc968230580615e506f5f4b937f2812141d88537a66fee12c05
58d9fd9fb9c8645d97e7e7f09cff4692a7c2b701730dfb84d3a36923ba48
context: 0200010001000304b619b60048a5cdd14aa8e5299e2439259f87b92ad890105
cd1c5cf191ecb5520bde19ced20f4e2bced384351ee4128e3e7147775cf44f33d8623f7b
40dee58d604b09c11d56c7ce347cb9d45c8b0d8e7cfeed5792e2dd52430bf79908b98641
ec91b5b4468376c6e797a0b5d9ff440de0b03d4ab03d69e4a5ab46623b92b7a45d80465d
f804f772b75d8349296932f59a17215ab58cb5d1fb3f530840bd2a3a9461812706edcdad
a222a42288ca90e8b9c04560c5acc3f89120f7291bfe5d010e0c1e3b0c44298fc1c149af
bf4c8996fb92427ae41e4649b934ca495991b7852b85555c4040629c64c5efec2f723040
7d612d16289d7c5d7afcf9340280abd2de1ab
secret: b4771505c7f85be0c8e1a67793c5181bcf8d78485f03d3df51a632d44a8da4e9
key: edfd08c4da2c9f3db58cac53cae1798ccbef1e640f1c09072390cbe0b53cdd0c
nonce: 28e30b5955de5fc27aa0d54c
```

A.5.1.1. Encryptions

```
sequence number: 0
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d30
ciphertext: f9b218b328c413b180b299a1e6ff0b04976a9f6c46b06f0add237dbe1598
1e1fd9a203d3d9056e5a1fdb25ae46

sequence number: 1
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d31
ciphertext: b088e62de28db00b504c786e73b0ac35a969169dc40f5af6926a06b872d7
7f3739d20b197ca3c9b95978ea49d9

sequence number: 2
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d32
ciphertext: 2f58a5698c8b14661abddc3a654d3549523b35a9e15f3404bd83c0b005bd
7d965d60ab4ef0db9a8b6ba9b4d9db

sequence number: 4
plaintext: 4265617574792069732074727574682c207472757468206265617574 79
aad: 436f756e742d34
ciphertext: 3a3c7983d212943c45bb8d50dc16ce00626656fc395506a871b25d5517cf
1d7797aa7139828e3703530c8309c7
```

A.6. DHKEM(P-256), HKDF-SHA256, ChaCha20Poly1305

A.6.1. AuthPSK Setup Information

```
mode: 3
kemID: 1
kdfID: 1
aeadID: 3
info: 4f6465206f6e20612047726563696616e2055726e
skR: 9f7b84c3e08b21dfb811b1809d53d4154456e6b1dc893bcc8e169ea32134af04
skI: e6ee38aad52ea20c46f987264011ee5f822fde3f0382ca98638c8b08dc4661f1
skE: 498b7e5c0bb0679ec27868accb89135c14f392189de6242d4170fdb640173292
psk: 6d656c6c6f6e
pskID: 456e6e796e20447572696e206172616e204d6f726961
pkR: 04277f8aa3debfbbfb0ee02a6a3e121cd4946ce33d86a67fce3fe5d5cb556ca0626
0f6dee4211aa005cfac2c764e7cb2ee7d92a68d98ced6d92ab4ba9516d04434
pkI: 0468bc33b8751319c315e73a2e3b0929177ffebe3e6725a65b951c21b30fa0ca7fe
2e9da68b2a817c489879872f84aeb116149f10a5c0a24b8db2b7cb9b7e9051a
pkE: 04340d95e86854286d59cc28f2adce3b67888a9c0f770e53a12bd8a2632558a0fed
6b3d41fd6dfa3cbdf813e3663c5f53f63816b2346139be8acc1f38228a2d30f
enc: 04340d95e86854286d59cc28f2adce3b67888a9c0f770e53a12bd8a2632558a0fed
6b3d41fd6dfa3cbdf813e3663c5f53f63816b2346139be8acc1f38228a2d30f
zz: 10bce6407decf68f945fd6393942128fa6f16d8c52e6422e6e83e36e0a3a505bb39f
dd1224be26566ced9d5484f42f1c9c16eb3858c70af34772ee9771781e64
context: 0300010001000304340d95e86854286d59cc28f2adce3b67888a9c0f770e53a
12bd8a2632558a0fed6b3d41fd6dfa3cbdf813e3663c5f53f63816b2346139be8acc1f38
228a2d30f04277f8aa3debfbbfb0ee02a6a3e121cd4946ce33d86a67fce3fe5d5cb556ca
06260f6dee4211aa005cfac2c764e7cb2ee7d92a68d98ced6d92ab4ba9516d044340468b
c33b8751319c315e73a2e3b0929177ffebe3e6725a65b951c21b30fa0ca7fe2e9da68b2a
817c489879872f84aeb116149f10a5c0a24b8db2b7cb9b7e9051aeca994d516108a16db8
6e155390f3c3cec6f0aff60ade1ae9e3189140b0f3dea55c4040629c64c5efec2f723040
7d612d16289d7c5d7afcf9340280abd2de1ab
secret: 0f29c1d186095b37e99f8e6247420c76eaf90da90285dcb86a88867668263aad
key: 1f8336823ab323cbc05303691c0891929f309efbc0c669e6ec6042ef801c576b
nonce: bdae0dfcac3dc33199992830
```

A.6.1.1. Encryptions

sequence number: 0
plaintext: 4265617574792069732074727574682c2074727574820626561757479
aad: 436f756e742d30
ciphertext: 50fcf59dc592c112d501370036005502d89952ae38a7a61efa5a296b4173
d2c32d46a916c3fa86e037cc8e900a

sequence number: 1
plaintext: 4265617574792069732074727574682c2074727574820626561757479
aad: 436f756e742d31
ciphertext: 1ac7522e5aaab444dab431f1c52ea84ea655cac0f60296dda43cbf120c5a
a104a0d9262278511660f9c4ee4c33

sequence number: 2
plaintext: 4265617574792069732074727574682c2074727574820626561757479
aad: 436f756e742d32
ciphertext: 05dc4db59436f4da435407e2ca5ed662926c1dab93cc344610f85227ed9e
607e0a422faa3ac3a1ccef97ad16e3

sequence number: 4
plaintext: 4265617574792069732074727574682c2074727574820626561757479
aad: 436f756e742d34
ciphertext: 524b40454bee3b4edb8be19770122f7b0fc7d13746044572a23087f389ee
11258a9873b0843eeadf0cf9a5d3c1

A.7. DHKEM(P-256), HKDF-SHA256, ChaCha20Poly1305

A.7.1. Base Setup Information

```
mode: 0
kemID: 1
kdfID: 1
aeadID: 3
info: 4f6465206f6e20612047726563696616e2055726e
skR: f0bcfbd0971219e2babcf8bf46f9b45afb170fdfe93a9218fb5eeb18592e6f7a
skI: 8d8dbf4b6b8d118b1ffe5708b754b4e1d148daa1a2e16647ef7c0facfb4888ad
skE: 68d9ec029f23625ef2d5ac9cc5f22f8325369ace5eef51f191baecba755bfbaf
psk: 6d656c6c6f6e
pskID: 456e6e796e20447572696e206172616e204d6f726961
pkR: 045e8a58b5816749e863f570ab5a321fb6b6564d8aae2d9d6677131cabc534cc35c
2d8f0747fa83ca5928b49dff4a211c16766ee38333d032dbdb4993cf175fc1b
pkI: 04fc797f3f266ca4f552b38adfbf40acee82bdf1b26dc3309cbedee696d07c78da2
b611090af50273826c67502e6d502052e772bef4d62c60978e2b4cedb21ecf2
pkE: 04ce353132868f8e171c4094c962916be7e15abe88d508c53a771ee274d3d275b39
43682116d07fe93b1566f9fe862dfd7c3d18e5237924cf7884789400d308056
enc: 04ce353132868f8e171c4094c962916be7e15abe88d508c53a771ee274d3d275b39
43682116d07fe93b1566f9fe862dfd7c3d18e5237924cf7884789400d308056
zz: eb7399ba2fb4a6a06e0b414e865dbf3d09031a4df4430ca73a4af6408f10a999
context: 0000010001000304ce353132868f8e171c4094c962916be7e15abe88d508c53
a771ee274d3d275b3943682116d07fe93b1566f9fe862dfd7c3d18e5237924cf78847894
00d308056045e8a58b5816749e863f570ab5a321fb6b6564d8aae2d9d6677131cabc534c
c35c2d8f0747fa83ca5928b49dff4a211c16766ee38333d032dbdb4993cf175fc1b00000
0000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000e3b0c44298fc1c149af
bf4c8996fb92427ae41e4649b934ca495991b7852b85555c4040629c64c5efec2f723040
7d612d16289d7c5d7afcf9340280abd2de1ab
secret: 7fd7a388a9f38c65daeee6ec32cb28c3675a77f606578ecef69c155ebe7cebd3
key: 6eb3b1f3db37ca004725290462a31ffd132e597f17f26ff2dd654d0ca89286bb
nonce: 07354970f99e0f602c4c4b53
```

A.7.1.1. Encryptions

```
sequence number: 0
plaintext: 4265617574792069732074727574682c2074727574820626561757479
aad: 436f756e742d30
ciphertext: 627c7e1c3c820c19aac9db27580df3ea16c9f6f298393ffa23e33b464c0d
f123ce2083c8c1f8b904d11ddb7afe

sequence number: 1
plaintext: 4265617574792069732074727574682c2074727574820626561757479
aad: 436f756e742d31
ciphertext: f9725c7f6760dc056dacb72d6b9e530c221df7c9ce8e466584f13f9a6934
59da09b459055e8536b17b38e74a5c

sequence number: 2
plaintext: 4265617574792069732074727574682c2074727574820626561757479
aad: 436f756e742d32
ciphertext: 6447b64e328d19a742e957b0a5a23aae04d7ba2dc54235008a06ec32c1fe
79b33de60ad462829c70d99fe197b3

sequence number: 4
plaintext: 4265617574792069732074727574682c2074727574820626561757479
aad: 436f756e742d34
ciphertext: bbb89746f9c9121f8ad078c27559354a86796bcf057ecca1ae6142d9e339
ff76f8aeb7101215b26200520538df
```

A.8. DHKEM(P-256), HKDF-SHA256, ChaCha20Poly1305

A.8.1. PSK Setup Information

mode: 1
kemID: 1
kdfID: 1
aeadID: 3
info: 4f6465206f6e20612047726563696616e2055726e
skR: e121dd61c164efe26be222b4c0409ae56099d65690ea98036f3bd6591036e5d4
skI: eafe161c698c549fc4f1ac49aa43e9a4683dda0ecdf4f10480a43e0d4643377f
skE: fed0929f409b0c4e94215ce0a0709d78b16bd3437fd094b663e9593e27867de6
psk: 6d656c6c6f6e
pskID: 456e6e796e20447572696e206172616e204d6f726961
pkR: 043b950975425f0eb96360e99704e558a3fccbe983930a262dcf0d84bcedb4a70a6
073ef769fdc9e9a2a13e39d25bc0d5152813c002e93114c4f0d0a31cedf50e0
pkI: 047542daee27355cf4e2667f7df7a95e6e759a044a214d9a1c79909f4b67614b920
3386ca9b03c64ec19a5ff5db5d42983c12ce690c7151a83513a8518119a5091
pkE: 04f266eac3045b039ac6aaa51b0deae083e5eb8e2d7e6c720d2cb572f1c2496aca9
9d74f7e722fd9d20ed61f44ad6f83d6aceaf51b40b8bd6c04035fd0fe87941b
enc: 04f266eac3045b039ac6aaa51b0deae083e5eb8e2d7e6c720d2cb572f1c2496aca9
9d74f7e722fd9d20ed61f44ad6f83d6aceaf51b40b8bd6c04035fd0fe87941b
zz: bcbf9865e689d4cc851ced68c176405cbec7dc87bf035ee4d510069ccc378611
context: 0100010001000304f266eac3045b039ac6aaa51b0deae083e5eb8e2d7e6c720
d2cb572f1c2496aca99d74f7e722fd9d20ed61f44ad6f83d6aceaf51b40b8bd6c04035fd
0fe87941b043b950975425f0eb96360e99704e558a3fccbe983930a262dcf0d84bcedb4a
70a6073ef769fdc9e9a2a13e39d25bc0d5152813c002e93114c4f0d0a31cedf50e000000
0000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000eca994d516108a16db8
6e155390f3c3cec6f0aff60ade1ae9e3189140b0f3dea55c4040629c64c5efec2f723040
7d612d16289d7c5d7afcf9340280abd2de1ab
secret: 1faa3fca75887e37ea8f1614d13c080cfe8ef587f9ea5c8d4e35ceeb10de13fb
key: 2480ce03ab0e7af028275c37792fbede44a826e3bc52ae7484d7ad88635f924f
nonce: 32f27c096edc4cd8c7dd2305

A.8.1.1. Encryptions

sequence number: 0
plaintext: 4265617574792069732074727574682c2074727574820626561757479
aad: 436f756e742d30
ciphertext: ad365850dc8801505650b6ea634b6dfc419f285cf6db33a94f07fb077ea0
c35ccb2d5d41863dbc948276d88820

sequence number: 1
plaintext: 4265617574792069732074727574682c2074727574820626561757479
aad: 436f756e742d31
ciphertext: f0337efa1fba66dff53816b0904898383ac58913962fbf3753eedd28f3db
e274c533fba24725f2ca6707a87a2f

sequence number: 2
plaintext: 4265617574792069732074727574682c2074727574820626561757479
aad: 436f756e742d32
ciphertext: 0ae22018b989b6d06e7f1e28cf9e79812a3e47e8f1b795e10c11dd324c81
be40ea63107d50136bc98f8e74b59e

sequence number: 4
plaintext: 4265617574792069732074727574682c2074727574820626561757479
aad: 436f756e742d34
ciphertext: 31c4a4c2b83a9db26e825d1e5d0057c99e3385149f474238f24646921aa5
a7e09905878a66af9256f83d2311f6

A.9. DHKEM(P-521), HKDF-SHA512, AES-GCM-256

A.9.1. Base Setup Information

mode: 0
kemID: 3
kdfID: 2
aeadID: 2
info: 4f6465206f6e2061204772656369616e2055726e
skR: 011d0d02be6fe4eb49654d5641f0ebb40b594aca84a7c1e76faa1fa0f5f3582502d
29fd140d07e71f538c75b93c8ca416af95ad79c5e7e2c0889adce490367492aee
skI: 01f6343e4068d73fc49b7d1701612b264b2ce45c035ac197ad11e3e057333c6b752
0a7a76936c69f04315d2a4e9497dceee45ed96e392ebe2f571e2beb8d6b465a8f
skE: 0099f6326effd78a5c54eeba93e911f35bdd418f2e1de3281ba2172aebf251ff736
14598c81686775c20618721c8c84466f2f6ca7b33636a3edbc69697087f8b414a
psk: 6d656c6c6f6e
pskID: 456e6e796e20447572696e206172616e204d6f726961
pkR: 04015c0e75e50b266e5fb8e3a3f317608bbf64843c47cca29faa3b894a4f9b95ba5
33bd40d343e5747f9e85d9ffee5b4ed803a20a23ff0a246ca4d9d59eae9c38291d000c39
6eade14ed2d42fafe7c4be26da6f07af8b6902af5fbb39f8b0b15c7120d7f57f46ed24e7
11dda2469f418377d8cc8de7ec8b0594e155b8d6d6ca699f99a4189
pkI: 0400430b1dbd3c329055a1e76b57abf0e9c0e28f01c6daf8cdddee13c0d0668789e
dcf4bbf7157e75b12b5f91294aaac807060bc8270b5d590fdb5c7e4d6d5b5c8cb5d01c01

e9ea663d2a487ea671ea08939e88a300bb8a14c3e439cfcd58266e0124cf02f86dc32dab
2aefea0f5485fa85540e2577a0b0e4d3936f03a20a6da341c59546c
pkE: 04007dd2d51007fb7c1f690aa26da4afcc005434fe0b0dd88aec70898e5f1443514
1c4d3e33328e87138e0a27b7295cc769add1c97eec91f955a510f52ef689f265d58006d5
08e92a0e2bbf318015ccd52772469af216708c2290386021648f9cc5c804965690852371
bd89375f6ede601219d606d73d35a1f02c049edb424ebf73f909e64
enc: 04007dd2d51007fb7c1f690aa26da4afcc005434fe0b0dd88aec70898e5f1443514
1c4d3e33328e87138e0a27b7295cc769add1c97eec91f955a510f52ef689f265d58006d5
08e92a0e2bbf318015ccd52772469af216708c2290386021648f9cc5c804965690852371
bd89375f6ede601219d606d73d35a1f02c049edb424ebf73f909e64
zz: 010114f78d86b60c3db2c0dfdf24a06b6c61d7bb1d60c9e412c792a4d468fb897e8d
23c93ff9b4e28eac54b6235b79f307f0429e00cc04587ed93d21ee722ce93e54
context: 000003000200020400 7dd2d51007fb7c1f690aa26da4afcc005434fe0b0dd88
aec70898e5f14435141c4d3e33328e87138e0a27b7295cc769add1c97eec91f955a510f5
2ef689f265d58006d508e92a0e2bbf318015ccd52772469af216708c2290386021648f9c
c5c804965690852371bd89375f6ede601219d606d73d35a1f02c049edb424ebf73f909e6
404015c0e75e50b266e5fb8e3a3f317608bbf64843c47cca29faa3b894a4f9b95ba533bd
40d343e5747f9e85d9ffee5b4ed803a20a23ff0a246ca4d9d59eae9c38291d000c396ead
e14ed2d42fafe7c4be26da6f07af8b6902af5fbb39f8b0b15c7120d7f57f46ed24e711dd
a2469f418377d8cc8de7ec8b0594e155b8d6d6ca699f99a41890000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000cf83e1357eefb8bdf1542850d66d8007d620e4050b5
715dc83f4a921d36ce9ce47d0d13c5d85f2b0ff8318d2877eec2f63b931bd47417a81a53
8327af927da3e490ce9df289fea4615a6eef004e5cec7a77f0f0478e663643a1ab75945a
0082e5b91ad84905c1632605d8377ed3d2cb688cf352d67466c37bfaa08c8c765077b
secret: 5d82ab17b6a25686329228495209c97da5be51262fa96170b082730817e92633
00315ef1064a19b2f2a33afc96720a73afa14d55826970d79ab3adf441ae1f8d
key: 38c2168887316bcbdb2d8e47d86c329f9dde21c19ff6ffe06e89e7ccde02945a
nonce: 30e7764a7a038f9830cab2d5

A.9.1.1. Encryptions

sequence number: 0
plaintext: 4265617574792069732074727574682c2074727574820626561757479
aad: 436f756e742d30
ciphertext: 324a22241a7886c3f8c69c982be9c5e05fec65612a1f720fc1a32ca469c3
c0979d19fca8edc4e2c0585f1acde1

sequence number: 1
plaintext: 4265617574792069732074727574682c2074727574820626561757479
aad: 436f756e742d31
ciphertext: 89f9624107c8a25e3d3e6b8ac277a36687a0cae634326589a7f89c0df897
46402bed1a2090cdc8c2c8a5757a21

sequence number: 2
plaintext: 4265617574792069732074727574682c2074727574820626561757479
aad: 436f756e742d32
ciphertext: c686d16cb37ff082745a6b1348871333d1b3374ce439aa0d3d95cfcfc78e
f9afe9c310b0141cc9d549734161ff

sequence number: 4
plaintext: 4265617574792069732074727574682c2074727574820626561757479
aad: 436f756e742d34
ciphertext: cc9e2936752e02fcc0be0f8aef030b9ba79a87004a1f311846887784575e
a8d274f43bba7e13569c6d4b31e689

A.10. DHKEM(P-521), HKDF-SHA512, AES-GCM-256

A.10.1. PSK Setup Information

mode: 1
kemID: 3
kdfID: 2
aeadID: 2
info: 4f6465206f6e20612047726563369616e2055726e
skR: 00953f712b5c288be50460d0372612f88c3da171196a4cd958b0f658e988aa56ce5
edc018e5899acda38649fc51049610bc8d1423189a9b5ff5f7e4fe1081df3c7b3
skI: 00c1842b428c23730b0d4eb3859b20bbbe8fef23068e4a6431647379e4a2b92e9c1
ab58682e0043b9df9c48df14863a6ded7a3a283258fe534aada53b68082eaa7a9
skE: 00416c4a6236475f01a0cd0a9a7a0d7229be88a63281fca350ea98d4438de57a23b
d5053572bf0f7bed8bc84aba6f7322cabc6b2540e789eb5823660eaad8e206fd9
psk: 6d656c6c6f6e
pskID: 456e6e796e20447572696e206172616e204d6f726961
pkR: 040108f48ce8fcd71e7e07270fa566cfd66d9b7ab124df141d700b82141264a0130
6b94265afc3c361c0c0dc514864d4b12e687697d24ddce2f0c77f9c257b8a2f5a380109c
b328c87f9833f729967a74675acc31bccdccf5016d442e7b33d2ecaf6f79d8a0e79d8259
b07cd173e320ca25c68acf5aa4f3793e7fc6d076baa953060847a02
pkI: 04016c7b375e83c3e9252a4768986fff255917ca307345b4a953031a0314b8209b5
b4d55521fb576db505c9afda9fdfe00ce6f0ad64f6547223179bf851a7cb7b9a71201b9b

5b3de83c31691ff10d41e3c38aca4ebf0a858a19d5dd01d247425e6697ef10ad9da8e946
7104c32d7ec5605b5835c074da5ce9000edaa8fbc4623d53431815b
pkE: 04014e52461ba98b13a2e0f47c39cffbbb78769d13f6eca78250fa2cb88ec2aa449
4cc9930c9fdc7057cef32dc714365a46a9abc03c8a53a22aba3c690f90b7602b0640058c
7100e423f782a9e650526506c6e02da601d06b92ebd68276e1d108e10da338918491c3d4
7043e4ba92a34dcff5a8afdb74ded907412b7704c569c6bbdb37ce8
enc: 04014e52461ba98b13a2e0f47c39cffbbb78769d13f6eca78250fa2cb88ec2aa449
4cc9930c9fdc7057cef32dc714365a46a9abc03c8a53a22aba3c690f90b7602b0640058c
7100e423f782a9e650526506c6e02da601d06b92ebd68276e1d108e10da338918491c3d4
7043e4ba92a34dcff5a8afdb74ded907412b7704c569c6bbdb37ce8
zz: 012281cc1b8c8c3d2397aafdbc02e8d20a0ce95da7f4f334739a74413a20c4bc551d
2e2c9b94162946ff8edc46ce43f3522c0dec9adfed372206a7fd0c2e2c598002
context: 010003000200020401 4e52461ba98b13a2e0f47c39cffbbb78769d13f6eca78
250fa2cb88ec2aa4494cc9930c9fdc7057cef32dc714365a46a9abc03c8a53a22aba3c69
0f90b7602b0640058c7100e423f782a9e650526506c6e02da601d06b92ebd68276e1d108
e10da338918491c3d47043e4ba92a34dcff5a8afdb74ded907412b7704c569c6bbdb37ce
8040108f48ce8fcd71e7e07270fa566cfd66d9b7ab124df141d700b82141264a01306b94
265afc3c361c0c0dc514864d4b12e687697d24ddce2f0c77f9c257b8a2f5a380109cb328
c87f9833f729967a74675acc31bccdccf5016d442e7b33d2ecaf6f79d8a0e79d8259b07c
d173e320ca25c68acf5aa4f3793e7fc6d076baa953060847a02000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000f19e7afbe93b9d8b9837fe0a40ada462caf9a031824
8f66dd7832fac65a58dcacbf170937f825b35d22fd19125483b1f2f6993549423617d8ab
9f65322d627b6490ce9df289fea4615a6eef004e5cec7a77f0f0478e663643a1ab75945a
0082e5b91ad84905c1632605d8377ed3d2cb688cf352d67466c37bfaa08c8c765077b
secret: 9978ffaa9ac0e2d311ed52db2232f77ac3491790d08c09b951843401a50100fe
a33c9aacc9a804832048f979ab3429af78d11de196a1fc55ef463dcda05b6081
key: 3f37b55ed1305cfbb36ecae3ae2926677bc46fe7c3ed99adce3b8d6cbd01d300
nonce: fe49897600606833182a3b11

A.10.1.1. Encryptions

sequence number: 0
plaintext: 4265617574792069732074727574682c2074727574820626561757479
aad: 436f756e742d30
ciphertext: d4b50a1608cbc0eab2091657b8310c073045c982f582c3ca26d8c5139f60
89d82aa62ffbf536db27de1877f935

sequence number: 1
plaintext: 4265617574792069732074727574682c2074727574820626561757479
aad: 436f756e742d31
ciphertext: 0547d0baac17c97b70b086f27574a3ca3e9303eb9083103f155ecb2e0877
6cbdb66c67be7647287de2be34c8e1

sequence number: 2
plaintext: 4265617574792069732074727574682c2074727574820626561757479
aad: 436f756e742d32
ciphertext: 43e7ed5027c26321533daac80b1abc3127cac53a6ca30b077e63b514d582
8bd72c5284a420dc7c949d9a686938

sequence number: 4
plaintext: 4265617574792069732074727574682c2074727574820626561757479
aad: 436f756e742d34
ciphertext: 17896c16e9915155240b1a5eff176aba8d5319e06879e64fdae655741794
a2cdbbf3bd1f665502eb2b8b22a593

A.11. DHKEM(P-521), HKDF-SHA512, AES-GCM-256

A.11.1. Auth Setup Information

mode: 2
kemID: 3
kdfID: 2
aeadID: 2
info: 4f6465206f6e20612047726563696616e2055726e
skR: 006ae74c6d37982c4a6087500b66948a715ca971e7aa43260bae4683d78818cfc72
8b0d72d6834f4d401f35db13f932e414b44d03071805fcbd513a57130e18e8323
skI: 01d9581f65c8cf1a90f1711fe377c15e68f534be11ea5e0158a8adebaa04f0be9c8
0d0f2517abf0cd117d9ca2073b604743076cee2405f4db2825ace05e0eae83354
skE: 0195e8805187cf89fc17007d90a75dff2dc9ae824aea70adf001ef539832932f0d8
cd7d3bbc94e712fb64d0e5f0acbb7cb79e5bde9d24304c8b4ed0091c8905da986
psk: 6d656c6c6f6e
pskID: 456e6e796e20447572696e206172616e204d6f726961
pkR: 040054116ccfb36d9cd99e59b100ef9dbc70a6992b38632ee7650659275cdcab37b
ce7e74f2381cc7292ba418051432c3a8aeb706b3c05fcb886b3a95a306ae9863f5900e9b
6db3150e7241fec23c607db539e1a2b2c1898b5d2b78ec1a3254bf022dee6e8c2f6265f7
1ac8e6003614accb8532dd58d5a07a959bde06b763f2f41a9c3ac32
pkI: 0400fdb5f8a16b80e8a61c967f4f4f6b83740fc3521cd1654ee7e1cc42f4f645a57
e68942303a48d37e09b7cf2be067f03ed0b7076fe0eda129bc65209c83bafbc0b5d012ba

9db99b61236f645076e7f8b10c8763517dfcefd07241e90aa6a595209fc6aafc227fd9d8
70c8c6b8d037dd5386513608f7779887e47db733fe97f74169d21c7
pkE: 0401c72c45d970511e760274a3cb8b4457132b32a0aff685e7d3bc859e2d1a08fcd
181576356f10f5ca4508492f684f4f1966dd8253f6c3227378a94fc026a019aa02400998
21eaf35b86f1cb4ab03400e80ae5a395f3e3b90d892fb0b074138ed15d83ac134bec6653
06f5878b1210fcb61801e1fe12d36ce85ce777e53a4817b6fc424c7
enc: 0401c72c45d970511e760274a3cb8b4457132b32a0aff685e7d3bc859e2d1a08fcd
181576356f10f5ca4508492f684f4f1966dd8253f6c3227378a94fc026a019aa02400998
21eaf35b86f1cb4ab03400e80ae5a395f3e3b90d892fb0b074138ed15d83ac134bec6653
06f5878b1210fcb61801e1fe12d36ce85ce777e53a4817b6fc424c7
zz: 0050f396090a25a9ceaec1441f09355575cd07bf869ea98176be4c78e621a300dcd9
9ddbc3f401d85639c82062da083cc14d19d7ac7f0856ca61a5781f96b0aef03c00cf0031
10e5ff76c521f02d5757974a9fc7cfac607d1d37d40de2aa8b7d4121dc5b425fb7db5006
1cf9518524b4c69e91ef8e937bae2d5b907cef592c112af20ed3
context: 020003000200020401c72c45d970511e760274a3cb8b4457132b32a0aff685e
7d3bc859e2d1a08fcd181576356f10f5ca4508492f684f4f1966dd8253f6c3227378a94f
c026a019aa0240099821eaf35b86f1cb4ab03400e80ae5a395f3e3b90d892fb0b074138e
d15d83ac134bec665306f5878b1210fcb61801e1fe12d36ce85ce777e53a4817b6fc424c
7040054116ccfb36d9cd99e59b100ef9dbc70a6992b38632ee7650659275cdcab37bce7e
74f2381cc7292ba418051432c3a8aeb706b3c05fcb886b3a95a306ae9863f5900e9b6db3
150e7241fec23c607db539e1a2b2c1898b5d2b78ec1a3254bf022dee6e8c2f6265f71ac8
e6003614accb8532dd58d5a07a959bde06b763f2f41a9c3ac320400fdb5f8a16b80e8a61
c967f4f4f6b83740fc3521cd1654ee7e1cc42f4f645a57e68942303a48d37e09b7cf2be0
67f03ed0b7076fe0eda129bc65209c83bafbc0b5d012ba9db99b61236f645076e7f8b10c
8763517dfcefd07241e90aa6a595209fc6aafc227fd9d870c8c6b8d037dd5386513608f7
779887e47db733fe97f74169d21c7cf83e1357eefb8bdf1542850d66d8007d620e4050b5
715dc83f4a921d36ce9ce47d0d13c5d85f2b0ff8318d2877eec2f63b931bd47417a81a53
8327af927da3e490ce9df289fea4615a6eef004e5cec7a77f0f0478e663643a1ab75945a
0082e5b91ad84905c1632605d8377ed3d2cb688cf352d67466c37bfaa08c8c765077b
secret: c0359db265419a1734df0b692284c1f1b8df60fde983221dbae7e423004e5f69
bc0a5cd932def77bc3300c8a9f26d28381c1c436e2343de07729e814b4049007
key: c6104b46ff7837431247494088 42f8e6b7e7a4a8661756744463d810cf5e99a5
nonce: 7928a07ba38e10c7fb17379b

A.11.1.1. Encryptions

sequence number: 0
plaintext: 4265617574792069732074727574682c2074727574820626561757479
aad: 436f756e742d30
ciphertext: 1a29ac1c5618a2672e66ea0affae0132985abddcd097cadd43761052d9cb
5aea800349a20c6d5e8d8138ef9d05

sequence number: 1
plaintext: 4265617574792069732074727574682c2074727574820626561757479
aad: 436f756e742d31
ciphertext: 0e6ef91c9e99b752677f636b84a40d59f4004afc408fc11e01611ec46b55
e37c978817d02aa0d9e292a4d71b36

sequence number: 2
plaintext: 4265617574792069732074727574682c2074727574820626561757479
aad: 436f756e742d32
ciphertext: ee6263f5ab9e33c956262ceb94f42356f30c7f6313aaf2940bb7dde78b4f
9759e55f7e2dfbd076a4c3cebab79a

sequence number: 4
plaintext: 4265617574792069732074727574682c2074727574820626561757479
aad: 436f756e742d34
ciphertext: ae46e55cc95e5dfd3a319573875ca78e228caf19b98e1beb6867edb6e88e
83b5a3c1a122e4be86feb1555fb6db

A.12. DHKEM(P-521), HKDF-SHA512, AES-GCM-256

A.12.1. AuthPSK Setup Information

mode: 3
kemID: 3
kdfID: 2
aeadID: 2
info: 4f6465206f6e20612047726563696616e2055726e
skR: 010bf0c9af1d5dda4c97151d6a9425c8f590aa7c0adff53c06d23380ae82bbd32f6
4ddf0c344d221d2f7657711d73de7fa25a75bd8fa75662029e118087276a5ef0b
skI: 01800af579dbd0c91a09fa0ff3ddf21d9a1447649528d777e00962c748159e6de03
880a70331fdab67761e25f78977128f81c7fee9897eea74b5ee80e94414b6aed6
skE: 013e65c19df676b1d7da923475c72c359fbdd91f224d68785bdf5891bbadfd136a3
3cc8c31408b9652be389f52e8a19d9401aedaf549a0654f246c277af48f59b2ba
psk: 6d656c6c6f6e
pskID: 456e6e796e20447572696e206172616e204d6f726961
pkR: 0400a1735a659c6b281e603f170b5e886ccfff35d274b042d43a815447fc80208f5
99704d63a67355b4d04bfdc367f60961f1b36b1e11bf3ed7599f58948f364a1023501ec6
b5acd7ce1cc2c3ec6dba8d72b08e472809623ac2df65dcb658dbd7d5f599c9ac624517d7
80b49b3d7619f5647be76a56a73fe2c3fc2ae620159cb1b7a437a94
pkI: 0400f0fe8e86b8d86d02a4dc5db6988f6343067174caff2e2700834b98d25045013
eb24cf00c03dd506d562625d0fe5c576910d176705cd4ba1fcd10f5a98f466a0a2200a62

9f9f62f6053b554bf09b2a547b844f3e040c2b92c548babbc73cf05e77f23d0cffa9e5df
d0a57f9be64bf453ec48cbd00f2e47349572fead07dc73658256331
pkE: 0400f86941ff8711fc30860d2d8c2e2894e3301a072940ff54c54c46a1c6b1ed46d
2003fc2ec948aaf78bce02dd71fc3af5c25468a394a935386716ac1d89d0663225900432
3a47f9edf3a5fa40b666077f49d26d2a42730bfda86f74a23154023232bfee03040eed85
c18f476375b018a5b5e848238f06efd9d59c92a00462c34aa4bca58
enc: 0400f86941ff8711fc30860d2d8c2e2894e3301a072940ff54c54c46a1c6b1ed46d
2003fc2ec948aaf78bce02dd71fc3af5c25468a394a935386716ac1d89d0663225900432
3a47f9edf3a5fa40b666077f49d26d2a42730bfda86f74a23154023232bfee03040eed85
c18f476375b018a5b5e848238f06efd9d59c92a00462c34aa4bca58
zz: 01175e2421582f1a01e492e7dc3acbb7426a9f592f53192d8ef67f9bb2a5187c9dc7
59b31ad3531aa9ebdfd9cf9c456538f216385c628efba5f22194eca8a5661e93017d4165
7e70cd7d44eff9b6ec79df014aa74c2643134e83f35d29ea3a2d40d31821ca6546e74e67
c61263d3ac8bbeaf58a61fd12f71dee776b1858a9da5509d90d4
context: 0300030002000204 00f86941ff8711fc30860d2d8c2e2894e3301a072940ff5
4c54c46a1c6b1ed46d2003fc2ec948aaf78bce02dd71fc3af5c25468a394a935386716ac
1d89d06632259004323a47f9edf3a5fa40b666077f49d26d2a42730bfda86f74a2315402
3232bfee03040eed85c18f476375b018a5b5e848238f06efd9d59c92a00462c34aa4bca5
80400a1735a659c6b281e603f170b5e886ccfff35d274b042d43a815447fc80208f59970
4d63a67355b4d04bfdc367f60961f1b36b1e11bf3ed7599f58948f364a1023501ec6b5ac
d7ce1cc2c3ec6dba8d72b08e472809623ac2df65dcb658dbd7d5f599c9ac624517d780b4
9b3d7619f5647be76a56a73fe2c3fc2ae620159cb1b7a437a940400f0fe8e86b8d86d02a
4dc5db6988f6343067174caff2e2700834b98d25045013eb24cf00c03dd506d562625d0f
e5c576910d176705cd4ba1fcd10f5a98f466a0a2200a629f9f62f6053b554bf09b2a547b
844f3e040c2b92c548babbc73cf05e77f23d0cffa9e5dfd0a57f9be64bf453ec48cbd00f
2e47349572fead07dc73658256331f19e7afbe93b9d8b9837fe0a40ada462caf9a031824
8f66dd7832fac65a58dcacbf170937f825b35d22fd19125483b1f2f6993549423617d8ab
9f65322d627b6490ce9df289fea4615a6eef004e5cec7a77f0f0478e663643a1ab75945a
0082e5b91ad84905c1632605d8377ed3d2cb688cf352d67466c37bfaa08c8c765077b
secret: 292ecdeeee0ea44e81594cc8a08d515efcbe3904979841866760cc2a1a765e54
e1f7189e502194481afc2d6c206d36a79bfe01ad5ae1a2ff1a208cd3bce6a3c1
key: 87a4d83695a7175153a89b82074ca3d3c70af9437714ee0e31c28faae24e0c90
nonce: 24ddc8cc50a9626caf84a204

A.12.1.1. Encryptions

sequence number: 0
plaintext: 4265617574792069732074727574682c2074727574820626561757479
aad: 436f756e742d30
ciphertext: 5569e7435fff9cb4cd306734c4caf42a6dfb3c38cc3ecf16d696d13e733f
cc671ea73297c671462e835d7b8557

sequence number: 1
plaintext: 4265617574792069732074727574682c2074727574820626561757479
aad: 436f756e742d31
ciphertext: e74619c530a9c7d54803ae3f585b0b41e114b1f59de11ceb1858674240dd
b5cb9aa670de1136a7094d03cf7424

sequence number: 2
plaintext: 4265617574792069732074727574682c2074727574820626561757479
aad: 436f756e742d32
ciphertext: ba291704b425866d69a82912f8aaed74c8796a733e3dc4d04b6eb6cfbcfa
2b73f4d4268dc2dca6fc6100018a71

sequence number: 4
plaintext: 4265617574792069732074727574682c2074727574820626561757479
aad: 436f756e742d34
ciphertext: 80bb5f19a7b934e26f4c74aff965152a0b275488d479041005069d1f43d4
73bd5d5ecdef200393135c8bfcd47b

Authors' Addresses

Richard L. Barnes
Cisco

Email: rlb@ipv.sx

Karthik Bhargavan
Inria

Email: karthikeyan.bhargavan@inria.fr