Danger

This is a "Hazardous Materials" module. You should **ONLY** use it if you're 100% absolutely sure that you know what you're doing because this module is full of land mines, dragons, and dinosaurs with laser guns.

Authenticated encryption

Authenticated encryption with associated data (AEAD) are encryption schemes which provide both confidentiality and integrity for their ciphertext. They also support providing integrity for associated data which is not encrypted.

class **cryptography.hazmat.primitives.ciphers.aead.ChaCha20Poly1305**(*key*)

Added in version 2.0.

The ChaCha20Poly1305 construction is defined in RFC 7539 section 2.8. It is a stream cipher combined with a MAC that offers strong integrity guarantees.

Parameters: **key** (bytes-like) – A 32-byte key. This **must** be kept secret.
Raises: cryptography.exceptions.UnsupportedAlgorithm – If the version of OpenSSL does not support ChaCha20Poly1305.

```
>>> import os
>>> from cryptography.hazmat.primitives.ciphers.aead import ChaCha20Poly1305
>>> data = b"a secret message"
>>> aad = b"authenticated but unencrypted data"
>>> key = ChaCha20Poly1305.generate_key()
>>> chacha = ChaCha20Poly1305(key)
>>> nonce = os.urandom(12)
>>> ct = chacha.encrypt(nonce, data, aad)
>>> chacha.decrypt(nonce, ct, aad)
b'a secret message'
```

classmethod **generate_key**()

Securely generates a random ChaCha20Poly1305 key.

Returns bytes: A 32 byte key.

encrypt(*nonce*, *data*, *associated_data*)

latest

Warning

Reuse of a `nonce` with a given `key` compromises the security of any message with that `nonce` and `key` pair.

Encrypts the `data` provided and authenticates the `associated_data`. The output of this can be passed directly to the `decrypt` method.

Parameters:
- **nonce** (bytes-like) – A 12 byte value. **NEVER REUSE A NONCE** with a key.
- **data** (bytes-like) – The data to encrypt.
- **associated_data** (bytes-like) – Additional data that should be authenticated with the key, but does not need to be encrypted. Can be `None`.

Returns bytes: The ciphertext bytes with the 16 byte tag appended.

Raises: OverflowError – If `data` or `associated_data` is larger than 2^{31} - 1 bytes.

`encrypt_into`(*nonce*, *data*, *associated_data*, *buf*)

Added in version 47.0.0.

Warning

Reuse of a `nonce` with a given `key` compromises the security of any message with that `nonce` and `key` pair.

Encrypts and authenticates the `data` provided as well as authenticating the `associated_data`. The output is written into the `buf` parameter.

Parameters:
- **nonce** (bytes-like) – A 12 byte value. **NEVER REUSE A NONCE** with a key.
- **data** (bytes-like) – The data to encrypt.
- **associated_data** (bytes-like) – Additional data that should be authenticated with the key, but does not need to be encrypted. Can be `None`.
- **buf** – A writable bytes-like object that must be exactly `len(data) + 16` bytes. The ciphertext with the 16 byte tag appended will be written to this buffer.

Returns int: The number of bytes written to the buffer (always `len(data) + 16`).

Raises:
- ValueError – If the buffer is not the correct size.
- OverflowError – If `data` or `associated_data` is larger than 2^{31} - 1 bytes.

`decrypt`(*nonce*, *data*, *associated_data*)

latest

Decrypts the `data` and authenticates the `associated_data`. If you called encrypt with `associated_data` you must pass the same `associated_data` in decrypt or the integrity

check will fail.

Parameters:
- **nonce** (bytes-like) – A 12 byte value. **NEVER REUSE A NONCE** with a key.
- **data** (bytes-like) – The data to decrypt (with tag appended).
- **associated_data** (bytes-like) – Additional data to authenticate. Can be `None` if none was passed during encryption.

Returns bytes: The original plaintext.

Raises: cryptography.exceptions.InvalidTag – If the authentication tag doesn't validate this exception will be raised. This will occur when the ciphertext has been changed, but will also occur when the key, nonce, or associated data are wrong.

`decrypt_into(nonce, data, associated_data, buf)`

Added in version 47.0.0.

Decrypts the `data` and authenticates the `associated_data`. If you called encrypt with `associated_data` you must pass the same `associated_data` in decrypt or the integrity check will fail. The output is written into the `buf` parameter.

Parameters:
- **nonce** (bytes-like) – A 12 byte value. **NEVER REUSE A NONCE** with a key.
- **data** (bytes-like) – The data to decrypt (with tag appended).
- **associated_data** (bytes-like) – Additional data to authenticate. Can be `None` if none was passed during encryption.
- **buf** – A writable bytes-like object that must be exactly `len(data) - 16` bytes. The plaintext will be written to this buffer.

Returns int: The number of bytes written to the buffer (always `len(data) - 16`).

Raises:
- ValueError – If the buffer is not the correct size.
- cryptography.exceptions.InvalidTag – If the authentication tag doesn't validate this exception will be raised. This will occur when the ciphertext has been changed, but will also occur when the key, nonce, or associated data are wrong.

class `cryptography.hazmat.primitives.ciphers.aead.AESGCM(key)`

Added in version 2.0.

Note

This class only supports 128-bit tags. If you need tag truncation (which is generally **not recommended**) you should use the `GCM` class with `Cipher`.

The AES-GCM construction is composed of the `AES` block cipher utilizing [illegible] Mode (GCM).

Parameters: **key** (bytes-like) – A 128, 192, or 256-bit key. This **must** be kept secret.

latest

```
>>> import os
>>> from cryptography.hazmat.primitives.ciphers.aead import AESGCM
>>> data = b"a secret message"
>>> aad = b"authenticated but unencrypted data"
>>> key = AESGCM.generate_key(bit_length=128)
>>> aesgcm = AESGCM(key)
>>> nonce = os.urandom(12)
>>> ct = aesgcm.encrypt(nonce, data, aad)
>>> aesgcm.decrypt(nonce, ct, aad)
b'a secret message'
```

classmethod **generate_key**(*bit_length*)

Securely generates a random AES-GCM key.

Parameters: **bit_length** – The bit length of the key to generate. Must be 128, 192, or 256.

Returns bytes: The generated key.

encrypt(*nonce*, *data*, *associated_data*)

Warning

Reuse of a `nonce` with a given `key` compromises the security of any message with that `nonce` and `key` pair.

Encrypts and authenticates the `data` provided as well as authenticating the `associated_data`. The output of this can be passed directly to the `decrypt` method.

Parameters:

- **nonce** (bytes-like) – NIST recommends a 96-bit IV length for best performance but it can be up to 2^{64} - 1 bits. **NEVER REUSE A NONCE** with a key.
- **data** (bytes-like) – The data to encrypt.
- **associated_data** (bytes-like) – Additional data that should be authenticated with the key, but is not encrypted. Can be `None`.

Returns bytes: The ciphertext bytes with the 16 byte tag appended.

Raises: OverflowError – If `data` or `associated_data` is larger than 2^{31} - 1 bytes.

encrypt_into(*nonce*, *data*, *associated_data*, *buf*)

Added in version 47.0.0.

Warning

Reuse of a `nonce` with a given `key` compromises the security of any that `nonce` and `key` pair.

latest

Encrypts and authenticates the `data` provided as well as authenticating the `associated_data`. The output is written into the `buf` parameter.

Parameters:
- **nonce** (bytes-like) – NIST recommends a 96-bit IV length for best performance but it can be up to 2^{64} - 1 bits. **NEVER REUSE A NONCE** with a key.
- **data** (bytes-like) – The data to encrypt.
- **associated_data** (bytes-like) – Additional data that should be authenticated with the key, but is not encrypted. Can be `None`.
- **buf** – A writable bytes-like object that must be exactly `len(data) + 16` bytes. The ciphertext with the 16 byte tag appended will be written to this buffer.

Returns int: The number of bytes written to the buffer (always `len(data) + 16`).

Raises:
- **ValueError** – If the buffer is not the correct size.
- **OverflowError** – If `data` or `associated_data` is larger than 2^{31} - 1 bytes.

`decrypt(nonce, data, associated_data)`

Decrypts the `data` and authenticates the `associated_data`. If you called encrypt with `associated_data` you must pass the same `associated_data` in decrypt or the integrity check will fail.

Parameters:
- **nonce** (bytes-like) – NIST recommends a 96-bit IV length for best performance but it can be up to 2^{64} - 1 bits. **NEVER REUSE A NONCE** with a key.
- **data** (bytes-like) – The data to decrypt (with tag appended).
- **associated_data** (bytes-like) – Additional data to authenticate. Can be `None` if none was passed during encryption.

Returns bytes: The original plaintext.

Raises: **cryptography.exceptions.InvalidTag** – If the authentication tag doesn't validate this exception will be raised. This will occur when the ciphertext has been changed, but will also occur when the key, nonce, or associated data are wrong.

`decrypt_into(nonce, data, associated_data, buf)`

Added in version 47.0.0.

Decrypts the `data` and authenticates the `associated_data`. If you called encrypt with `associated_data` you must pass the same `associated_data` in decrypt or the integrity check will fail. The output is written into the `buf` parameter.

Parameters:
- **nonce** (bytes-like) – NIST recommends a 96-bit IV length for best performance but it can be up to 2^{64} - 1 bits. **NEVER REUSE A NONCE** with a key.
- **data** (bytes-like) – The data to decrypt (with tag appended).
- **associated_data** (bytes-like) – Additional data to authenticate. Can be `None` if none was passed during encryption.
- **buf** – A writable bytes-like object that must be exactly `len(data) - 16` bytes. The plaintext will be written to this buffer.

Returns int: The number of bytes written to the buffer (always `len(data) - 16`).

Raises:
- **ValueError** – If the buffer is not the correct size.
- **cryptography.exceptions.InvalidTag** – If the authentication tag doesn't validate this exception will be raised. This will occur when the ciphertext has been changed, but will also occur when the key, nonce, or associated data are wrong.

---

class **cryptography.hazmat.primitives.ciphers.aead.AESGCMSIV**(*key*)

Added in version 42.0.0.

The AES-GCM-SIV construction is defined in **RFC 8452** and is composed of the `AES` block cipher utilizing Galois Counter Mode (GCM) and a synthetic initialization vector (SIV).

Parameters: **key** (bytes-like) – A 128, 192, or 256-bit key. This **must** be kept secret.

Raises: **cryptography.exceptions.UnsupportedAlgorithm** – If the version of OpenSSL does not support AES-GCM-SIV.

```
>>> import os
>>> from cryptography.hazmat.primitives.ciphers.aead import AESGCMSIV
>>> data = b"a secret message"
>>> aad = b"authenticated but unencrypted data"
>>> key = AESGCMSIV.generate_key(bit_length=128)
>>> aesgcmsiv = AESGCMSIV(key)
>>> nonce = os.urandom(12)
>>> ct = aesgcmsiv.encrypt(nonce, data, aad)
>>> aesgcmsiv.decrypt(nonce, ct, aad)
b'a secret message'
```

classmethod **generate_key**(*bit_length*)

Securely generates a random AES-GCM-SIV key.

Parameters: **bit_length** – The bit length of the key to generate. Must be 128, 192, or 256.

Returns bytes: The generated key.

latest

encrypt(*nonce*, *data*, *associated_data*)

Encrypts and authenticates the `data` provided as well as authenticating the `associated_data`. The output of this can be passed directly to the `decrypt` method.

Parameters:
- **nonce** (bytes-like) – A 12-byte value.
- **data** (bytes-like) – The data to encrypt.
- **associated_data** (bytes-like) – Additional data that should be authenticated with the key, but is not encrypted. Can be `None`.

Returns bytes: The ciphertext bytes with the 16 byte tag appended.

Raises: OverflowError – If `data` or `associated_data` is larger than 2^{32} - 1 bytes.

`encrypt_into(nonce, data, associated_data, buf)`

Added in version 47.0.0.

Encrypts and authenticates the `data` provided as well as authenticating the `associated_data`. The output is written into the `buf` parameter.

Parameters:
- **nonce** (bytes-like) – A 12-byte value.
- **data** (bytes-like) – The data to encrypt.
- **associated_data** (bytes-like) – Additional data that should be authenticated with the key, but is not encrypted. Can be `None`.
- **buf** – A writable bytes-like object that must be exactly `len(data) + 16` bytes. The ciphertext with the 16 byte tag appended will be written to this buffer.

Returns int: The number of bytes written to the buffer (always `len(data) + 16`).

Raises:
- ValueError – If the buffer is not the correct size.
- OverflowError – If `data` or `associated_data` is larger than 2^{32} - 1 bytes.

`decrypt(nonce, data, associated_data)`

Decrypts the `data` and authenticates the `associated_data`. If you called encrypt with `associated_data` you must pass the same `associated_data` in decrypt or the integrity check will fail.

Parameters:
- **nonce** (bytes-like) – A 12-byte value.
- **data** (bytes-like) – The data to decrypt (with tag appended).
- **associated_data** (bytes-like) – Additional data to authenticate. Can be `None` if none was passed during encryption.

Returns bytes: The original plaintext.

Raises: cryptography.exceptions.InvalidTag – If the authentication tag doesn't validate this exception will be raised. This will occur when the ciphertext has been changed, but will also occur when the key, nonce, or associated data are wrong.

`decrypt_into(nonce, data, associated_data, buf)`

Added in version 47.0.0.

Decrypts the `data` and authenticates the `associated_data`. If you called encrypt with `associated_data` you must pass the same `associated_data` in decrypt or the integrity check will fail. The output is written into the `buf` parameter.

Parameters:
- **nonce** (bytes-like) – A 12-byte value.
- **data** (bytes-like) – The data to decrypt (with tag appended).
- **associated_data** (bytes-like) – Additional data to authenticate. Can be `None` if none was passed during encryption.
- **buf** – A writable bytes-like object that must be exactly `len(data) - 16` bytes. The plaintext will be written to this buffer.

Returns int: The number of bytes written to the buffer (always `len(data) - 16`).

Raises:
- **ValueError** – If the buffer is not the correct size.
- **cryptography.exceptions.InvalidTag** – If the authentication tag doesn't validate this exception will be raised. This will occur when the ciphertext has been changed, but will also occur when the key, nonce, or associated data are wrong.

---

class **cryptography.hazmat.primitives.ciphers.aead.AESOCB3**(*key*)

Added in version 36.0.0.

The OCB3 construction is defined in **RFC 7253**. It is an AEAD mode that offers strong integrity guarantees and good performance.

Parameters: **key** (bytes-like) – A 128, 192, or 256-bit key. This **must** be kept secret.

Raises: **cryptography.exceptions.UnsupportedAlgorithm** – If the version of OpenSSL does not support AES-OCB3.

```
>>> import os
>>> from cryptography.hazmat.primitives.ciphers.aead import AESOCB3
>>> data = b"a secret message"
>>> aad = b"authenticated but unencrypted data"
>>> key = AESOCB3.generate_key(bit_length=128)
>>> aesocb = AESOCB3(key)
>>> nonce = os.urandom(12)
>>> ct = aesocb.encrypt(nonce, data, aad)
>>> aesocb.decrypt(nonce, ct, aad)
b'a secret message'
```

classmethod **generate_key**(*bit_length*)

Securely generates a random AES-OCB3 key.

Parameters: **bit_length** – The bit length of the key to generate. Must be 128, 192, or 256.

Returns bytes: The generated key.

encrypt(*nonce*, *data*, *associated_data*)

Warning

Reuse of a `nonce` with a given `key` compromises the security of any message with that `nonce` and `key` pair.

Encrypts and authenticates the `data` provided as well as authenticating the `associated_data`. The output of this can be passed directly to the `decrypt` method.

Parameters:
- **nonce** (bytes-like) – A 12-15 byte value. **NEVER REUSE A NONCE** with a key.
- **data** (bytes-like) – The data to encrypt.
- **associated_data** (bytes-like) – Additional data that should be authenticated with the key, but is not encrypted. Can be `None`.

Returns bytes: The ciphertext bytes with the 16 byte tag appended.

Raises: OverflowError – If `data` or `associated_data` is larger than 2^{31} - 1 bytes.

encrypt_into(*nonce*, *data*, *associated_data*, *buf*)

Added in version 47.0.0.

Warning

Reuse of a `nonce` with a given `key` compromises the security of any message with that `nonce` and `key` pair.

Encrypts and authenticates the `data` provided as well as authenticating the `associated_data`. The output is written into the `buf` parameter.

Parameters:
- **nonce** (bytes-like) – A 12-15 byte value. **NEVER REUSE A NONCE** with a key.
- **data** (bytes-like) – The data to encrypt.
- **associated_data** (bytes-like) – Additional data that should be authenticated with the key, but is not encrypted. Can be `None`.
- **buf** – A writable bytes-like object that must be exactly `len(data) + 16` bytes. The ciphertext with the 16 byte tag appended will be written to this buffer.

Returns int: The number of bytes written to the buffer (always `len(data) + 16`).

Raises:
- ValueError – If the buffer is not the correct size.
- OverflowError – If `data` or `associated_data` is larger than 2^{31} - 1 bytes.

latest

decrypt(*nonce*, *data*, *associated_data*)

Decrypts the `data` and authenticates the `associated_data`. If you called encrypt with

`associated_data` you must pass the same `associated_data` in decrypt or the integrity check will fail.

Parameters:
- **nonce** (bytes-like) – A 12 byte value. **NEVER REUSE A NONCE** with a key.
- **data** (bytes-like) – The data to decrypt (with tag appended).
- **associated_data** (bytes-like) – Additional data to authenticate. Can be `None` if none was passed during encryption.

Returns bytes: The original plaintext.

Raises: cryptography.exceptions.InvalidTag – If the authentication tag doesn't validate this exception will be raised. This will occur when the ciphertext has been changed, but will also occur when the key, nonce, or associated data are wrong.

`decrypt_into(nonce, data, associated_data, buf)`

Added in version 47.0.0.

Decrypts the `data` and authenticates the `associated_data`. If you called encrypt with `associated_data` you must pass the same `associated_data` in decrypt or the integrity check will fail. The output is written into the `buf` parameter.

Parameters:
- **nonce** (bytes-like) – A 12-15 byte value. **NEVER REUSE A NONCE** with a key.
- **data** (bytes-like) – The data to decrypt (with tag appended).
- **associated_data** (bytes-like) – Additional data to authenticate. Can be `None` if none was passed during encryption.
- **buf** – A writable bytes-like object that must be exactly `len(data) - 16` bytes. The plaintext will be written to this buffer.

Returns int: The number of bytes written to the buffer (always `len(data) - 16`).

Raises:
- ValueError – If the buffer is not the correct size.
- cryptography.exceptions.InvalidTag – If the authentication tag doesn't validate this exception will be raised. This will occur when the ciphertext has been changed, but will also occur when the key, nonce, or associated data are wrong.

class `cryptography.hazmat.primitives.ciphers.aead.AESSIV(key)`

Added in version 37.0.0.

The SIV (synthetic initialization vector) construction is defined in RFC 5297. Depending on how it is used, SIV allows either deterministic authenticated encryption or nonce-based, misuse-resistant authenticated encryption.

Parameters: **key** (bytes-like) – A 256, 384, or 512-bit key (double sized from typical AES). This **must** be kept secret.

Raises: cryptography.exceptions.UnsupportedAlgorithm – If the version of OpenSSL does not support AES-SIV.

```
>>> import os
>>> from cryptography.hazmat.primitives.ciphers.aead import AESSIV
>>> data = b"a secret message"
>>> nonce = os.urandom(16)
>>> aad = [b"authenticated but unencrypted data", nonce]
>>> key = AESSIV.generate_key(bit_length=512)  # AES256 requires 512-bit keys for SIV
>>> aessiv = AESSIV(key)
>>> ct = aessiv.encrypt(data, aad)
>>> aessiv.decrypt(ct, aad)
b'a secret message'
```

classmethod **generate_key**(*bit_length*)

Securely generates a random AES-SIV key.

Parameters: **bit_length** – The bit length of the key to generate. Must be 256, 384, or 512. AES-SIV splits the key into an encryption and MAC key, so these lengths correspond to AES 128, 192, and 256.

Returns bytes: The generated key.

encrypt(*data*, *associated_data*)

Note

SIV performs nonce-based authenticated encryption when a component of the associated data is a nonce. The final associated data in the list is used for the nonce.

Random nonces should have at least 128-bits of entropy. If a nonce is reused with SIV authenticity is retained and confidentiality is only compromised to the extent that an attacker can determine that the same plaintext (and same associated data) was protected with the same nonce and key.

If you do not supply a nonce encryption is deterministic and the same (plaintext, key) pair will always produce the same ciphertext.

Encrypts and authenticates the `data` provided as well as authenticating the `associated_data`. The output of this can be passed directly to the `decrypt` method.

Parameters:
- **data** (bytes-like) – The data to encrypt.
- **associated_data** (*list*) – An optional `list` of `bytes-like objects`. This is additional data that should be authenticated with the key, but is not encrypted. Can be `None`. In SIV mode the final element of this list is treated as a `nonce`.

Returns bytes: The ciphertext bytes with the 16 byte tag **prepended**.

Raises: OverflowError – If `data` or an `associated_data` el than 2^{31} - 1 bytes.

`encrypt_into(data, associated_data, buf)`

Added in version 47.0.0.

Note

SIV performs nonce-based authenticated encryption when a component of the associated data is a nonce. The final associated data in the list is used for the nonce.

Random nonces should have at least 128-bits of entropy. If a nonce is reused with SIV authenticity is retained and confidentiality is only compromised to the extent that an attacker can determine that the same plaintext (and same associated data) was protected with the same nonce and key.

If you do not supply a nonce encryption is deterministic and the same (plaintext, key) pair will always produce the same ciphertext.

Encrypts and authenticates the `data` provided as well as authenticating the `associated_data`. The output is written into the `buf` parameter.

Parameters:
- **data** (bytes-like) – The data to encrypt.
- **associated_data** (*list*) – An optional `list` of `bytes-like objects`. This is additional data that should be authenticated with the key, but is not encrypted. Can be `None`. In SIV mode the final element of this list is treated as a `nonce`.
- **buf** – A writable bytes-like object that must be exactly `len(data) + 16` bytes. The ciphertext with the 16 byte tag **prepended** will be written to this buffer.

Returns int: The number of bytes written to the buffer (always `len(data) + 16`).

Raises:
- **ValueError** – If the buffer is not the correct size.
- **OverflowError** – If `data` or an `associated_data` element is larger than 2^{31} - 1 bytes.

`decrypt(data, associated_data)`

Decrypts the `data` and authenticates the `associated_data`. If you called encrypt with `associated_data` you must pass the same `associated_data` in decrypt or the integrity check will fail.

Parameters:
- **data** (*bytes*) – The data to decrypt (with tag **prepended**).
- **associated_data** (*list*) – An optional `list` of `bytes-like objects`. This is additional data that should be authenticated with the key, but is not encrypted. Can be `None` if none was used during encryption.

Returns bytes: The original plaintext.

Raises: cryptography.exceptions.InvalidTag – If the authentication tag doesn't validate this exception will be raised. This will occur when the ciphertext has been changed, but will also occur when the key or associated data are wrong.

`decrypt_into`(*data*, *associated_data*, *buf*)

Added in version 47.0.0.

Decrypts the `data` and authenticates the `associated_data`. If you called encrypt with `associated_data` you must pass the same `associated_data` in decrypt or the integrity check will fail. The output is written into the `buf` parameter.

Parameters:
- **data** (*bytes*) – The data to decrypt (with tag **prepended**).
- **associated_data** (*list*) – An optional `list` of `bytes-like objects`. This is additional data that should be authenticated with the key, but is not encrypted. Can be `None` if none was used during encryption.
- **buf** – A writable bytes-like object that must be exactly `len(data) - 16` bytes. The plaintext will be written to this buffer.

Returns int: The number of bytes written to the buffer (always `len(data) - 16`).

Raises:
- ValueError – If the buffer is not the correct size.
- cryptography.exceptions.InvalidTag – If the authentication tag doesn't validate this exception will be raised. This will occur when the ciphertext has been changed, but will also occur when the key or associated data are wrong.

---

class `cryptography.hazmat.primitives.ciphers.aead.AESCCM`(*key*, *tag_length=16*)

Added in version 2.0.

The AES-CCM construction is composed of the `AES` block cipher utilizing Counter with CBC-MAC (CCM) (specified in RFC 3610).

Parameters:
- **key** (bytes-like) – A 128, 192, or 256-bit key. This **must** be kept secret.
- **tag_length** (*int*) – The length of the authentication tag. This defaults to 16 bytes and it is **strongly** recommended that you do not make it shorter unless absolutely necessary. Valid tag lengths are 4, 6, 8, 10, 12, 14, and 16.

Raises: cryptography.exceptions.UnsupportedAlgorithm – If the version of OpenSSL does not support AES-CCM.

```
>>> import os
>>> from cryptography.hazmat.primitives.ciphers.aead import AESCCM
>>> data = b"a secret message"
>>> aad = b"authenticated but unencrypted data"
>>> key = AESCCM.generate_key(bit_length=128)
>>> aesccm = AESCCM(key)
>>> nonce = os.urandom(13)
>>> ct = aesccm.encrypt(nonce, data, aad)
>>> aesccm.decrypt(nonce, ct, aad)
b'a secret message'
```

classmethod **generate_key**(*bit_length*)

Securely generates a random AES-CCM key.

Parameters: **bit_length** – The bit length of the key to generate. Must be 128, 192, or 256.

Returns bytes: The generated key.

encrypt(*nonce*, *data*, *associated_data*)

Warning

Reuse of a `nonce` with a given `key` compromises the security of any message with that `nonce` and `key` pair.

Encrypts and authenticates the `data` provided as well as authenticating the `associated_data`. The output of this can be passed directly to the `decrypt` method.

Parameters:

- **nonce** (bytes-like) – A value of between 7 and 13 bytes. The maximum length is determined by the length of the ciphertext you are encrypting and must satisfy the condition: `len(data) < 2 ** (8 * (15 - len(nonce)))` **NEVER REUSE A NONCE** with a key.
- **data** (bytes-like) – The data to encrypt.
- **associated_data** (bytes-like) – Additional data that should be authenticated with the key, but is not encrypted. Can be `None`.

Returns bytes: The ciphertext bytes with the tag appended.

Raises: **OverflowError** – If `data` or `associated_data` is larger than 2^{31} - 1 bytes.

encrypt_into(*nonce*, *data*, *associated_data*, *buf*)

Added in version 47.0.0.

Warning

latest

Reuse of a `nonce` with a given `key` compromises the security of any message with

that `nonce` and `key` pair.

Encrypts and authenticates the `data` provided as well as authenticating the `associated_data`. The output is written into the `buf` parameter.

Parameters:
- **nonce** (bytes-like) – A value of between 7 and 13 bytes. The maximum length is determined by the length of the ciphertext you are encrypting and must satisfy the condition: `len(data) < 2 ** (8 * (15 - len(nonce)))` **NEVER REUSE A NONCE** with a key.
- **data** (bytes-like) – The data to encrypt.
- **associated_data** (bytes-like) – Additional data that should be authenticated with the key, but is not encrypted. Can be `None`.
- **buf** – A writable bytes-like buffer into which the ciphertext and tag will be written. This must be exactly `len(data) + tag_length` bytes long.

Returns int: The number of bytes written to the buffer (this is always `len(data) + tag_length`).

Raises:
- **OverflowError** – If `data` or `associated_data` is larger than 2^{31} - 1 bytes.
- **ValueError** – If `buf` is not the correct length.

`decrypt`(*nonce*, *data*, *associated_data*)

Decrypts the `data` and authenticates the `associated_data`. If you called encrypt with `associated_data` you must pass the same `associated_data` in decrypt or the integrity check will fail.

Parameters:
- **nonce** (bytes-like) – A value of between 7 and 13 bytes. This is the same value used when you originally called encrypt. **NEVER REUSE A NONCE** with a key.
- **data** (bytes-like) – The data to decrypt (with tag appended).
- **associated_data** (bytes-like) – Additional data to authenticate. Can be `None` if none was passed during encryption.

Returns bytes: The original plaintext.

Raises: cryptography.exceptions.InvalidTag – If the authentication tag doesn't validate this exception will be raised. This will occur when the ciphertext has been changed, but will also occur when the key, nonce, or associated data are wrong.

`decrypt_into`(*nonce*, *data*, *associated_data*, *buf*)

Added in version 47.0.0.

Decrypts the `data` and authenticates the `associated_data`. If you calle
`associated_data` you must pass the same `associated_data` in decrypt o
check will fail. The output is written into the `buf` parameter.

Parameters:

- **nonce** (bytes-like) – A value of between 7 and 13 bytes. This is the same value used when you originally called encrypt. **NEVER REUSE A NONCE** with a key.
- **data** (bytes-like) – The data to decrypt (with tag appended).
- **associated_data** (bytes-like) – Additional data to authenticate. Can be `None` if none was passed during encryption.
- **buf** – A writable bytes-like object that must be exactly `len(data) - tag_length` bytes. The plaintext will be written to this buffer.

Returns int: The number of bytes written to the buffer (always `len(data) - tag_length`).

Raises:

- **ValueError** – If the buffer is not the correct size.
- **cryptography.exceptions.InvalidTag** – If the authentication tag doesn't validate this exception will be raised. This will occur when the ciphertext has been changed, but will also occur when the key, nonce, or associated data are wrong.