AES

AES (Advanced Encryption Standard) is a symmetric block cipher standardized by NIST . It has a fixed data block size of 16 bytes. Its keys can be 128, 192, or 256 bits long.

AES is very fast and secure, and it is the de facto standard for symmetric encryption.

As an example, encryption can be done as follows:

```
>>> from Crypto.Cipher import AES
>>>
>>> key = b'Sixteen byte key'
>>> cipher = AES.new(key, AES.MODE_EAX)
>>>
>>> nonce = cipher.nonce
>>> ciphertext, tag = cipher.encrypt_and_digest(data)
```

The recipient can obtain the original message using the same key and the incoming triple `(nonce, ciphertext, tag)`:

```
>>> from Crypto.Cipher import AES
>>>
>>> key = b'Sixteen byte key'
>>> cipher = AES.new(key, AES.MODE_EAX, nonce=nonce)
>>> plaintext = cipher.decrypt(ciphertext)
>>> try:
>>>     cipher.verify(tag)
>>>     print("The message is authentic:", plaintext)
>>> except ValueError:
>>>     print("Key incorrect or message corrupted")
```

Module's constants for the modes of operation supported with AES:

var MODE_ECB: Electronic Code Book (ECB)

var MODE_CBC: Cipher-Block Chaining (CBC)

var MODE_CFB: Cipher FeedBack (CFB)

var MODE_OFB: Output FeedBack (OFB)

var MODE_CTR: CounTer Mode (CTR)

var MODE_OPENPGP:

OpenPGP Mode

var MODE_CCM: Counter with CBC-MAC (CCM) Mode

var MODE_EAX: EAX Mode

var MODE_GCM: Galois Counter Mode (GCM)

var MODE_SIV: Syntethic Initialization Vector (SIV)

var MODE_OCB: Offset Code Book (OCB)

Crypto.Cipher.AES.new(*key*, *mode*, **args*, ***kwargs*)

Create a new AES cipher.

Parameters:

- key (*bytes/bytearray/memoryview*) –
  The secret key to use in the symmetric cipher.

  It must be 16, 24 or 32 bytes long (respectively for *AES-128*, *AES-192* or *AES-256*).

  For `MODE_SIV` only, it doubles to 32, 48, or 64 bytes.
- mode (One of the supported `MODE_*` constants) – The chaining mode to use for encryption or decryption. If in doubt, use `MODE_EAX`.

Keyword Arguments:

- iv (*bytes*, *bytearray*, *memoryview*) – (Only applicable for `MODE_CBC`, `MODE_CFB`, `MODE_OFB`, and `MODE_OPENPGP` modes).

  The initialization vector to use for encryption or decryption.

  For `MODE_CBC`, `MODE_CFB`, and `MODE_OFB` it must be 16 bytes long.

  For `MODE_OPENPGP` mode only, it must be 16 bytes long for encryption and 18 bytes for decryption (in the latter case, it is actually the *encrypted* IV which was prefixed to the ciphertext).

  If not provided, a random byte string is generated (you must then read its value with the `iv` attribute).
- nonce (*bytes*, *bytearray*, *memoryview*) – (Only applicable for `MODE_CCM`, `MODE_EAX`, `MODE_GCM`, `MODE_SIV`, `MODE_OCB`, and `MODE_CTR`).

  A value that must never be reused for any other encryption done with this key (except possibly for `MODE_SIV`, see below).

  For `MODE_EAX`, `MODE_GCM` and `MODE_SIV` there are no restrictions on its length (recommended: 16 bytes).

  For `MODE_CCM`, its length must be in the range [7..13]. Bear in mind that with CCM there is a trade-off between nonce length and maximum message

size. Recommendation: 11 bytes.

For `MODE_OCB`, its length must be in the range [1..15] (recommended: 15).

For `MODE_CTR`, its length must be in the range [0..15] (recommended: 8).

For `MODE_SIV`, the nonce is optional, if it is not specified, then no nonce is being used, which renders the encryption deterministic.

If not provided, for modes other than `MODE_SIV`, a random byte string of the recommended length is used (you must then read its value with the `nonce` attribute).

- segment_size (*integer*) – (Only `MODE_CFB`).The number of bits the plaintext and ciphertext are segmented in. It must be a multiple of 8. If not specified, it will be assumed to be 8.
- mac_len : (*integer*) – (Only `MODE_EAX`, `MODE_GCM`, `MODE_OCB`, `MODE_CCM`) Length of the authentication tag, in bytes.

  It must be even and in the range [4..16]. The recommended value (and the default, if not specified) is 16.
- msg_len : (*integer*) – (Only `MODE_CCM`). Length of the message to (de)cipher. If not specified, `encrypt` must be called with the entire message. Similarly, `decrypt` can only be called once.
- assoc_len : (*integer*) – (Only `MODE_CCM`). Length of the associated data. If not specified, all associated data is buffered internally, which may represent a problem for very large messages.
- initial_value : (*integer* or *bytes/bytearray/memoryview*) – (Only `MODE_CTR`). The initial value for the counter. If not present, the cipher will start counting from 0. The value is incremented by one for each block. The counter number is encoded in big endian mode.
- counter : (*object*) – Instance of `Crypto.Util.Counter`, which allows full customization of the counter block. This parameter is incompatible to both `nonce` and `initial_value`.
- use_aesni : (*boolean*) – Use Intel AES-NI hardware extensions (default: use if available).

Return: an AES object, of the applicable mode.