15.2. hmac — Keyed-Hashing for Message Authentication¶

New in version 2.2.

This module implements the HMAC algorithm as described by **RFC 2104**.

hmac.new(*key*[, *msg*[, *digestmod*]])¶

Return a new hmac object. If *msg* is present, the method call `update(msg)` is made. *digestmod* is the digest constructor or module for the HMAC object to use. It defaults to the `hashlib.md5()` constructor.

Note

The md5 hash has known weaknesses but remains the default for backwards compatibility. Choose a better one for your application.

An HMAC object has the following methods:

hmac.update(*msg*)¶
Update the hmac object with the string *msg*. Repeated calls are equivalent to a single call with the concatenation of all the arguments: `m.update(a); m.update(b)` is equivalent to `m.update(a + b)`.

hmac.digest()¶
Return the digest of the strings passed to the `update()` method so far. This string will be the same length as the *digest_size* of the digest given to the constructor. It may contain non-ASCII characters, including NUL bytes.

hmac.hexdigest()¶
Like `digest()` except the digest is returned as a string twice the length containing only hexadecimal digits. This may be used to exchange the value safely in email or other non-binary environments.

hmac.copy()¶
Return a copy (“clone”) of the hmac object. This can be used to efficiently compute the digests of strings that share a common initial substring.

See also

Module `hashlib`
The Python module providing secure hash functions.

Previous topic

15.1. `hashlib` — Secure hashes and message digests

Next topic

15.3. `md5` — MD5 message digest algorithm

This Page

- Show Source

Navigation

- index
- modules |
- next |
- previous |
- Python v2.6.4 documentation »
- The Python Standard Library »
- 15. Cryptographic Services »

Last updated on Feb 26, 2010. Created using Sphinx 0.6.3.