15.1. `hashlib` — Secure hashes and message digests¶

New in version 2.5.

This module implements a common interface to many different secure hash and message digest algorithms. Included are the FIPS secure hash algorithms SHA1, SHA224, SHA256, SHA384, and SHA512 (defined in FIPS 180-2) as well as RSA's MD5 algorithm (defined in Internet **RFC 1321**). The terms secure hash and message digest are interchangeable. Older algorithms were called message digests. The modern term is secure hash.

Note

If you want the adler32 or crc32 hash functions they are available in the `zlib` module.

Warning

Some algorithms have known hash collision weaknesses, see the FAQ at the end.

There is one constructor method named for each type of *hash*. All return a hash object with the same simple interface. For example: use `sha1()` to create a SHA1 hash object. You can now feed this object with arbitrary strings using the `update()` method. At any point you can ask it for the *digest* of the concatenation of the strings fed to it so far using the `digest()` or `hexdigest()` methods.

Constructors for hash algorithms that are always present in this module are `md5()`, `sha1()`, `sha224()`, `sha256()`, `sha384()`, and `sha512()`. Additional algorithms may also be available depending upon the OpenSSL library that Python uses on your platform.

For example, to obtain the digest of the string `'Nobody inspects the spammish repetition'`:

```
>>> import hashlib
>>> m = hashlib.md5()
>>> m.update("Nobody inspects")
>>> m.update(" the spammish repetition")
>>> m.digest()
'\xbbd\x9c\x83\xdd\x1e\xa5\xc9\xd9\xde\xc9\xa1\x8d\xf0\xff\xe9'
>>> m.digest_size
16
>>> m.block_size
64
```

More condensed:

```
>>> hashlib.sha224("Nobody inspects the spammish repetition").hexdigest()
'a4337bc45a8fc544c03f52dc550cd6e1e87021bc896588bd79e901e2'
```

A generic `new()` constructor that takes the string name of the desired algorithm as its first parameter also exists to allow access to the above listed hashes as well as any other algorithms that your OpenSSL library may offer. The named constructors are much faster than `new()` and should be preferred.

Using `new()` with an algorithm provided by OpenSSL:

```
>>> h = hashlib.new('ripemd160')
>>> h.update("Nobody inspects the spammish repetition")
>>> h.hexdigest()
'cc4a5ce1b3df48aec5d22d1f16b894a0b894eccc'
```

The following values are provided as constant attributes of the hash objects returned by the constructors:

`hash.digest_size`¶

The size of the resulting hash in bytes.

`hash.block_size`¶

The internal block size of the hash algorithm in bytes.

A hash object has the following methods:

`hash.update(arg)`¶
Update the hash object with the string *arg*. 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)`.

`hash.digest()`¶
Return the digest of the strings passed to the `update()` method so far. This is a string of `digest_size` bytes which may contain non-ASCII characters, including null bytes.

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

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

See also

Module `hmac`
A module to generate message authentication codes using hashes.

Module `base64`
Another way to encode binary hashes for non-binary environments.

http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf
The FIPS 180-2 publication on Secure Hash Algorithms.

http://www.cryptography.com/cnews/hash.html
Hash Collision FAQ with information on which algorithms have known issues and what that means regarding their use.

Previous topic

15. Cryptographic Services

Next topic

15.2. `hmac` — Keyed-Hashing for Message Authentication

This Page

- Show Source

Navigation

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