Модуль hashlib, алгоритмы хеширования в Python

Интерфейс для различных безопасных алгоритмов хеширования

Модуль hashlib реализует общий интерфейс для множества различных безопасных алгоритмов хеширования и дайджеста сообщений. В модуль включены алгоритмы безопасного хеширования, такие как FIPS SHA1, SHA224, SHA256, SHA384 и SHA512, определенные в FIPS 180-2, а также алгоритм MDA RSA, определенный в Интернете RFC 1321.

Термины "безопасный хеш" и "дайджест сообщения" являются взаимозаменяемыми:

- Старые алгоритмы назывались дайджестами сообщений.
- Современный термин - безопасный хеш.

Примечание.

- Если вам нужны хеш-функции adler32 или crc32, то они доступны в модуле zlib.
- Ссылка для тех, кто ищет встроенную функцию хеш-значения объекта hash().

Предупреждение.

Некоторые алгоритмы имеют известные недостатки хеш-коллизий.

Хеш-алгоритмы.

Модуль hashlib поддерживает одну функцию для каждого алгоритма хеширования, названную так-же как называется сам алгоритм. Например: для создания хеш-объекта для алгоритма хеширования SHA-256, то используйте функцию hashlib.sha256().

Все функции возвращают хеш-объект с одинаковым простым интерфейсом. Этот объект можно "кормить" байтоподобными объектами, используя метод HASH.update(). В любой момент можно запросить дайджест(хеш) о конкатенации данных, переданных в него, используя методы HASH.digest() или HASH.hexdigest().

Новое в Python 3.6: добавлены hashlib.blake2b() и hashlib.blake2s().

Новое в Python 3.9: теперь все конструкторы модуля hashlib принимают ключевой аргумент usedforsecurity, используемый для[illegible]ечения безопасности со значением по умолчанию True. Значение False позволяет использовать небезопасные и заблокированные алгоритмы хеширования в ограниченных средах. Значение False указывает, что алгоритм хеширования не

используется в контексте безопасности, например как не криптографическая функция одностороннего сжатия.

Модуль hashlib теперь использует SHA3 и SHAKE из OpenSSL 1.1.1 и новее.


```
>>> import hashlib
>>> hash = hashlib.sha256()
>>> hash.update(b"Nobody inspects")
>>> hash.update(b" the spammish repetition")
>>> hash.hexdigest()
# '031edd7d41651593c5fe5c006fa5752b37fddff7bc4e843aa6af0c950f4b9406'
>>> hash.digest_size
32
>>> hash.block_size
64
```

Примечания:

- Для лучшей производительности многопоточности, Python GIL запускается для данных размером более 2047 байт при создании объекта или при обновлении методом HASH.update().
- Передача строковых объектов в HASH.update() не поддерживается, поскольку хеши работают с байтами, а не с символами.

Функции алгоритмов хеширования, которые всегда присутствуют в модуле hashlib, являются hashlib.sha1(), hashlib.sha224(), hashlib.sha256(), hashlib.sha384(), hashlib.sha512(), hashlib.blake2b() и hashlib.blake2s().

> ***Внимание!*** *Алгоритм hashlib.md5() также доступен в версиях до Python3.7, но этот алгоритм хеширования полностью удален с версии Python3.8.*

Дополнительные функции алгоритмов также могут быть доступны в зависимости от библиотеки OpenSSL, которую Python использует на запущенной платформе. На большинстве платформ также доступны hashlib.sha3_224(), hashlib.sha3_256(), hashlib.sha3_384(), hashlib.sha3_512(), hashlib.shake_128(), hashlib.shake_256().

Примеры использования:

```
>>> import hashlib
# список алгоритмов хеширования, которые
# гарантированно присутствуют на любых платформах.
>>> hashlib.algorithms_guaranteed
# {'sha3_512', 'sha384', 'md5', 'shake_128',
# 'sha256', 'sha3_256', 'sha3_224', 'sha512',
# 'blake2s', 'sha3_384', 'sha224', 'shake_256',
# 'blake2b', 'sha1'}

# быстро вычислить хеш строки
>>> hashlib.sha224(b"Nobody inspects the spammish repetition").hexdigest()
# 'a4337bc45a8fc544c03f52dc550cd6e1e87021bc896588bd79e901e2'
```

Пример вычисление хеш суммы файла.

```
import hashlib

# Так как алгоритмы работают с байтами то и
# файл надо открывать на чтение байтов 'rb'
with open('/usr/bin/python3', 'rb') as f:
    hsh = hashlib.sha1()
    while True:
        data = f.read(2048)
        if not data:
            break
        hsh.update(data)
    rez = hsh.hexdigest()

>>> print(rez)
# ab7701b3f3073c2f23744d0d558ed47aa59f400d
```

Содержание раздела:

Вверх

- КРАТКИЙ ОБЗОР МАТЕРИАЛА.

Вверх