Министерство образования Республики Беларусь

Белорусский государственный университет информатики и радиоэлектроники

Кафедра информатики

ОТЧЕТ

по лабораторной работе №5

**Хэш-функции.**

Выполнил:

Александров А.А.

Проверил:

Артемьев В.С.

Минск 2019

**Задание:** реализовать программное средство контроля целостности сообщений с помощью вычисления хэш-функции и алгоритма HMAC

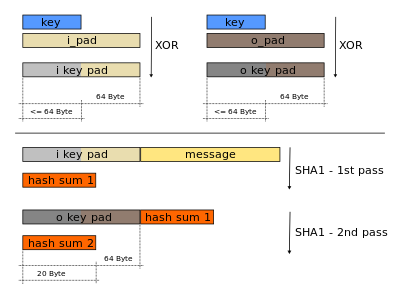
**Введение**

HMAC (hash-based message authentication code) - код аутентификации (проверки подлинности) сообщений, использующий хеш-функции, механизм проверки целостности информации, позволяющий гарантировать то, что данные, передаваемые или хранящиеся в ненадёжной среде, не были изменены посторонними лицами. Механизм HMAC использует MAC, описан в RFC 2104, в стандартах организаций ANSI, IETF, ISO и NIST.

MAC — стандарт, описывающий способ обмена данными и способ проверки целостности передаваемых данных с использованием секретного ключа. Два клиента, использующие HMAC, как правило, разделяют общий секретный ключ. HMAC — надстройка над MAC; механизм обмена данными с использованием секретного ключа (как в MAC) и хеш-функций. В зависимости от используемой хеш-функции выделяют HMAC-MD5, HMAC-SHA1, HMAC-RIPEMD128, HMAC-RIPEMD160 и т. п.

**Алгоритм**

* b, block\_size — размер блока в байтах;
* H, hash — хеш-функция;
* ipad — блок вида ( 0x36 0x36 0x36 ... 0x36 ), где байт 0x36 повторяется b раз; 0x36 — константа, магическое число, приведённое в RFC 2104; «i» от «inner»[[1]](https://ru.wikipedia.org/wiki/HMAC" \l "cite_note-rfc2104-1);
* К, key — секретный ключ (общий для отправителя и получателя);
* K0 — изменённый ключ K (уменьшенный или увеличенный до размера блока (до b байт));
* L — размер в байтах строки, возвращаемой хеш-функцией H; L зависит от выбранной хеш-функции и обычно меньше размера блока;



HMAC использует два прохода хэш - вычислений. Секретный ключ первый используется для получения двух ключей - внутренний и внешний. Первый проход алгоритма производит внутренний хэш, полученный из сообщения и внутреннего ключа. Второй проход производит окончательный HMAC-код, полученный из внутренней хеш - результата и внешнего ключа. Таким образом, алгоритм обеспечивает лучший иммунитет против атак расширения длины.

Итеративной хэш - функция разбивает сообщение на блоки фиксированного размера и итерацию над ними с функцией сжатия. Так, например, SHA-256 работает на 512-битовых блоков. Размер выходного HMAC та же, что и основной хеш - функции (например, 256 и 1600 бит в случае SHA-256 и SHA-3, соответственно), хотя оно может быть укорочены, если это желательно.

HMAC не шифрует сообщение. Вместо этого сообщения (зашифрованные или нет), должны быть отправлены вместе с HMAC. Сторона с секретным ключом может проверить код аутентификации, и, если он является подлинным, полученные и вычисленные хэши совпадают.

Определение и анализ конструкции HMAC был впервые опубликован в 1996 году в статье Mihir Bellare , Ран Канетти, и Хьюго Кравчик, и они также писали RFC 2104 в 1997 1996 документе также определен вариант под названием NMAC. FIPS PUB 198 обобщается и стандартизирует использование HMACs. HMAC используется в рамках протокола IPsec и TLS протоколы и JSON веб - токенов.

**Уязвимости**

В 2006 году Jongsung Ким, Алекс Бирюков, Барт Preneel иSeokhie Hong показал, как отличить HMAC с уменьшеннымиверсиями MD5 и SHA-1 или полные версии Haval, MD4 и SHA-0 от случайной функции или HMAC со случайной функцией.Уязвимость позволяет злоумышленникуразработать атаку подделки на HMAC. В 2009 году Xiaoyun Wang и др. представил отличительнуюатаку на HMAC-MD5 без использования связанных ключей. Онможет отличить экземпляр HMAC с MD5 от экземпляра сослучайной функцией с 297 запросов с вероятностью 0,87.

В 2011 году информационный RFC 6151 резюмировал соображения безопасности в MD5 и HMAC-MD5.Для HMAC-MD5 RFC, следует , что, хотя безопасность MD5серьезно скомпрометирована, - известныев настоящее время «атаки на HMAC-MD5, кажется, неуказывают на практическую уязвимость при использовании вкачестве кода аутентификации сообщения» , но это такжедобавляет , что «для нового дизайна протокола, CipherSuite сHMAC-MD5 не должны быть включены» .

**Результаты работы программы**



**Код программы**

from hashlib import sha1

trans\_5C = bytes((x ^ 0x5C) for x in range(256))

trans\_36 = bytes((x ^ 0x36) for x in range(256))

digest\_size = None

class HMAC:

blocksize = 64 # 512-bit HMAC; can be changed in subclasses.

def \_\_init\_\_(self, key, msg=None, digestmod=''):

if not isinstance(key, (bytes, bytearray)):

raise TypeError("key: expected bytes or bytearray, but got %r" % type(key).\_\_name\_\_)

if not digestmod:

raise TypeError("Missing required parameter 'digestmod'.")

self.digest\_cons = digestmod

self.outer = self.digest\_cons()

self.inner = self.digest\_cons()

self.digest\_size = self.inner.digest\_size

blocksize = self.inner.block\_size

self.block\_size = blocksize

if len(key) > blocksize:

key = self.digest\_cons(key).digest()

key = key.ljust(blocksize, b'\0')

self.outer.update(key.translate(trans\_5C))

self.inner.update(key.translate(trans\_36))

if msg is not None:

self.update(msg)

def update(self, msg):

self.inner.update(msg)

def \_current(self):

h = self.outer.copy()

h.update(self.inner.digest())

return h

def digest(self):

h = self.\_current()

return h.digest()

def hexdigest(self):

h = self.\_current()

return h.hexdigest()

def hmac(key, msg=None, digestmod=''):

return HMAC(key, msg, digestmod)

text = b"MEY\_TEXT"

key = b"SECRET\_KEY"

hashed = hmac(key=key, msg=text, digestmod=sha1)

signature = hashed.hexdigest()

print(signature)