Week 8 Homework due on Friday Nov 2, 22:00 Hour

Group 5

Wong Ann Yi (1004000)
Liu Bowen (1004028)
Tan Chin Leong Leonard (1004041)

Exercise 1

Hash "51.505-Foundations-of-Cybersecurity-MSSD" and "51.505-Foundations-of-Cybersecurity-MSSd", respectively using SHA1. Observe the difference of these 2 hash values.

Answers:

The codes are written in Python 2.7:

^^
```
from Crypto.Cipher import AES
import hashlib
import hmac
def Sha1():
    str1= "51.505-Foundations-of-Cybersecurity-MSSD"
    str2= "51.505-Foundations-of-Cybersecurity-MSSd"
    return (hashlib.sha1(str1).hexdigest()), (hashlib.sha1(str2).hexdigest())
print Sha1()
```
^^

The results of the hash values are:
For "51.505-Foundations-of-Cybersecurity-MSSD", the hash value is:
`'d2d2b25b4ee22957a08c41bfcb12e3b16e810699'`

For the "51.505-Foundations-of-Cybersecurity-MSSd", the hash value is:
`'da7e687280aca8cd68a50ca5c1445f82da9783f7'`

Despite a small difference in the plain text, the resulting hash values of the two plain texts are completely different.

Exercise 2

Compute any official test vector of HMAC-SHA256
(see https://tools.ietf.org/html/rfc4868#section-2.7.2.1).

Answers:

From official test vector site, we select the following test case:
Test Case AUTH256-1:
Key = 0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b
0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b (32 bytes)

Data = 4869205468657265 ("Hi There")

PRF-HMAC-SHA-256 =
198a607eb44bfbc69903a0f1cf2bbdc5ba0aa3f3d9ae3c1c7a3b1696a0b68cf7

We use Python 2.7 to write the following codes:

^^

```
from Crypto.Cipher import AES
import hashlib
import hmac
def Hmac256():
    key =
('0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b').decode('hex')
    msg = ('4869205468657265').decode('hex')
    h = hmac.new( key, msg, hashlib.sha256 )
    return (h.hexdigest())
print Hmac256()
```

^^

The resulting hash value is:
`198a607eb44bfbc69903a0f1cf2bbdc5ba0aa3f3d9ae3c1c7a3b1696a0b68cf7`

The hash value is the same as the result from the official test vector site.

Exercise 3

Let us define a hash function Hn(.) that executes SHA-512 and outputs the n bits.
Find a collision of H8, H16, H24, H32, and H40. Measure how long it takes to find a collision.

Answers:

We will perform the first 3 hash function of 8, 16, 24 bits.

H8 (using Python 2.7 to code):

We hashed the plain text "MSSD" using SHA512. With the hash value (using the first 8, 16 & 24 bits), we use it to compare with a series of hash values generated by hashing integer(s) starting from 0 (and increment by 1 at each step) until we find a match.

H8:

```
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
def collision():
    start = time.time()
    msg1 = hashlib.sha512('mssd').hexdigest()[:2]
    counter = 0
    while(True):
        states = (hashlib.sha512(str(counter)).hexdigest())[:2] == msg1
        if (states == True):
            break;
        counter = counter+1
    end = time.time()
    return end-start, counter
print collision()
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

The resulting value is: `(4.1961669921875e-05, 14)`

The time taken is 4.196 x 10^(-5) seconds
The integer which has the same has value (using the first 8 bits) is 14

The H16:

```
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
def collision16():
    start = time.time()
    msg1 = hashlib.sha512('mssd').hexdigest()[:4]
    counter = 0
    while(True):
        states = (hashlib.sha512(str(counter)).hexdigest())[:4] == msg1
        if (states == True):
```

```
        break;
      counter = counter+1
   end = time.time()
   return end-start, counter
print collision16()
```

^^^

The resulting value is: `(0.018130064010620117, 9995)`

The time taken is: 0.0181 seconds

The integer which matches the same value (of the first 16 bits) is 9995

H24:

^^

```
def collision24():
   start = time.time()
   msg1 = hashlib.sha512('mssd').hexdigest()[:6]
   counter = 0
   while(True):
      states = (hashlib.sha512(str(counter)).hexdigest())[:6] == msg1
      if (states == True):
         break;
      counter = counter+1
   end = time.time()
   return end-start, counter
print collision24()
```

^^^

The resulting value is: `(0.03895902633666992, 18902)`

The time taken is 0.03895 seconds

The integer which matches the hash value (of the first 24 bits) is: 18902

Exercise 4

For H8, H16, H24, H32 and H40 find a preimage of the corresponding hashes: "\00", "\00"*2, "\00"*3, "\00"*4, and "\00"*5. Measure how long it takes to find a preimage.

Answers:

We will perform the first three hash function of 8, 16 and 24 bits.

We use Python 2.7 to develop the codes:

For H8:

```
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
def preImage8():
    start = time.time()
    counter = 0
    while(True):
        states = (hashlib.sha512(str(counter)).hexdigest())[:2] == '00'
        if (states == True):
            break;
        counter = counter+1
    end = time.time()
    return end-start, counter
print (preImage8())
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

The result is: `(0.0001499652862548828, 61)`
The preimage is: 61
The time taken is: 0.0001499 seconds

For H16:

```
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
def preImage16():
    start = time.time()
    counter = 0
    while(True):
        states = (hashlib.sha512(str(counter)).hexdigest())[:4] == '0000'
        if (states == True):
            break;
        counter = counter+1
    end = time.time()
    return end-start, counter
print (preImage16())
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

The result is: `(0.4179039001464844, 288946)`

The preimage is: 288946
The time taken is: 0.4179 seconds

For H24:

^^

```
def preImage24():
    start = time.time()
    counter = 0
    while(True):
        states = (hashlib.sha512(str(counter)).hexdigest())[:6] == '000000'
        if (states == True):
            break;
        counter = counter+1
    end = time.time()
    return end-start, counter
print (preImage24())
```

^^

The resulting result is: `(10.036580085754395, 6899310)`
The preimage is: 6899310
The time taken is: 10.03658 seconds