hash-compare-test-rsa

February 24, 2020

[69]:
```
from hashcompare import Hash
from hashlib import sha1, sha256, blake2b, blake2s
from zlib import crc32
import time
import inspect
import numpy as np
import copy
```

[70]:
```
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives.asymmetric import padding
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.serialization import load_pem_private_key
from cryptography.hazmat.primitives.serialization import load_pem_public_key
```

[180]:
```
class RSA_Hash():

    def __init__(self, name):
        self.labryPubKey = load_pem_public_key(open('/Users/labry/Downloads/
↪labry_public.pem','rb').read(), default_backend())
        self.name = name
        self.sha = sha256()

    def update(self, data):
        self.sha.update(data)
        self.packet = self.sha.digest()
        #self.packet = data

    def digest(self):
            self.ciphertext = self.labryPubKey.encrypt(
                self.packet,
                padding.OAEP(
                    mgf=padding.MGF1(algorithm=hashes.SHA256()),
                    algorithm=hashes.SHA256(),
                    label=None
                )
            )
            return self.ciphertext
```

[182]:
```
x_range = [1,10,20,30,40,50,60,70,80,90, 100, 110]
#x_range = [15]
names_of_hashes = ['sha256','blake2b','blake2s','rsa-sha256']

rsa = RSA_Hash("rsa-sha256")
hash_mapper = [Hash(sha256(),names_of_hashes[0]),␣
 ↪Hash(blake2b(),names_of_hashes[1]),Hash(blake2s(),names_of_hashes[2]),␣
 ↪Hash(rsa,names_of_hashes[2])]
hash_mapper = np.array(hash_mapper)

simulated_packet = "abcdefghij"
packet= []

for weight in x_range:
    packet.append(simulated_packet * weight)

for idx, hash_f in enumerate(hash_mapper):

    for i in range(len(x_range)):
        hash_mapper[idx].set_start(time.time())
        for j in range(Hash.NUM_OF_ROUNDS):
            #packet = simulated_packet*i
            hash_f.update(packet[i].encode())
            hash_v = hash_f.digest()

        hash_mapper[idx].set_finish(time.time())

    print("{} {} performed {} operations in {} {} \n".format
        (idx, hash_mapper[idx].get_name(), Hash.NUM_OF_ROUNDS, ␣
 ↪len(hash_mapper[idx].get_duration()), hash_mapper[idx].get_duration()))
```

```
0 sha256 performed 10000 operations in 12 [0.021763086318969727,
0.019660234451293945, 0.022747039794921875, 0.026158809661865234,
0.028717756271362305, 0.033689022064208984, 0.03716397285461426,
0.040405988693237305, 0.042118072509765625, 0.04643821716308594,
0.04790306091308594, 0.05140113830566406]

1 blake2b performed 10000 operations in 12 [0.015569925308227539,
0.02309584617614746, 0.026061058044433594, 0.028560876846313477,
0.030650854110717773, 0.03776884078979492, 0.024763822555541992,
0.02611398696899414, 0.028738975524902344, 0.03510093688964844,
0.030112028121948242, 0.031687021255493164]

2 blake2s performed 10000 operations in 12 [0.01170206069946289,
0.01355886459350586, 0.015223979949951172, 0.017187833786010742,
0.018893957138061523, 0.0231931209564209, 0.02486395835876465,
0.02624821662902832, 0.028892993927001953, 0.030658960342407227,
```

```
0.04074811935424805, 0.040040016174316406]

3 blake2s performed 10000 operations in 12 [0.8271300792694092,
0.8154799938201904, 0.8206741809844971, 0.818490743637085, 0.8511502742767334,
0.8529708385467529, 0.8311119079589844, 0.8376638889312744, 0.8347969055175781,
0.8615458011627197, 0.8546669483184814, 0.8490149974822998]
```

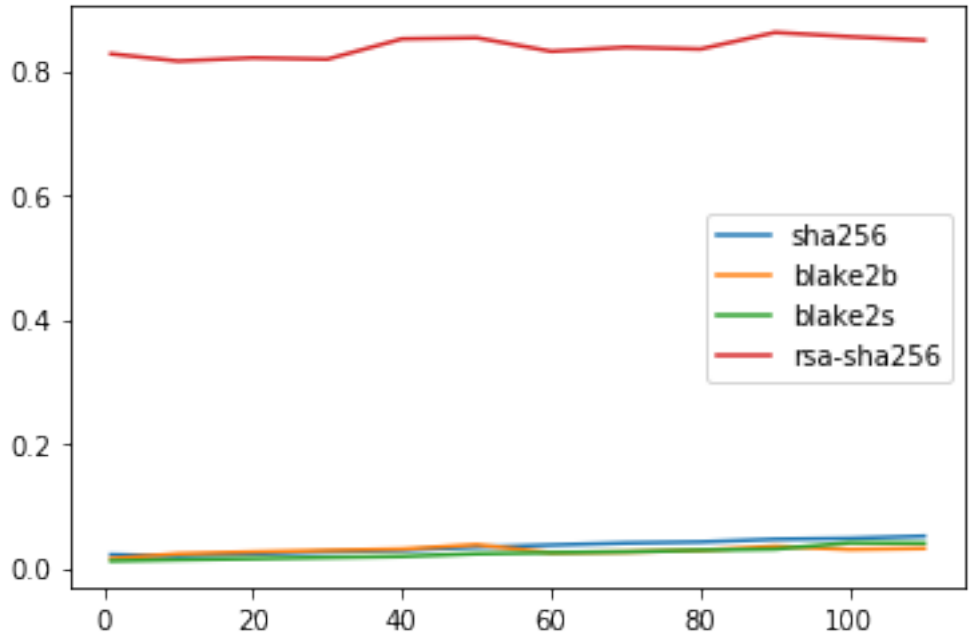
[183]:
```
default_duration = np.array(hash_mapper[0].get_duration())

import pandas as pd
import matplotlib.pyplot as plt
default_duration = pd.DataFrame(default_duration)
```

[191]:
```
#print(len(x_range))
zero_data = np.zeros(shape=(len(names_of_hashes),1))
zero_series = pd.Series(zero_data[:,0])
#print(zero_series)
plot_result2 = pd.DataFrame({names_of_hashes[0]: zero_series,␣
 ↪names_of_hashes[1]: zero_series,
                  names_of_hashes[2]: zero_series}, index=x_range)

#print(plot_result2)
for idx, hash_f in enumerate(hash_mapper):
    plot_result2[names_of_hashes[idx]] = np.array(hash_mapper[idx].
 ↪get_duration()[0])
    #print(hash_mapper[idx].get_duration())

plot_result2.plot()
print(plot_result2)

com_sha256_rsa = hash_mapper[3].get_duration()[0]/hash_mapper[0].
 ↪get_duration()[0]
com_blake2b_rsa = hash_mapper[3].get_duration()/hash_mapper[1].get_duration()
com_blake2s_rsa = hash_mapper[3].get_duration()/hash_mapper[2].get_duration()
com_sha256_rsa, com_blake2b_rsa, com_blake2s_rsa
```

```
       sha256   blake2b   blake2s  rsa-sha256
1    0.021763  0.015570  0.011702    0.827130
10   0.019660  0.023096  0.013559    0.815480
20   0.022747  0.026061  0.015224    0.820674
30   0.026159  0.028561  0.017188    0.818491
40   0.028718  0.030651  0.018894    0.851150
50   0.033689  0.037769  0.023193    0.852971
60   0.037164  0.024764  0.024864    0.831112
70   0.040406  0.026114  0.026248    0.837664
80   0.042118  0.028739  0.028893    0.834797
```

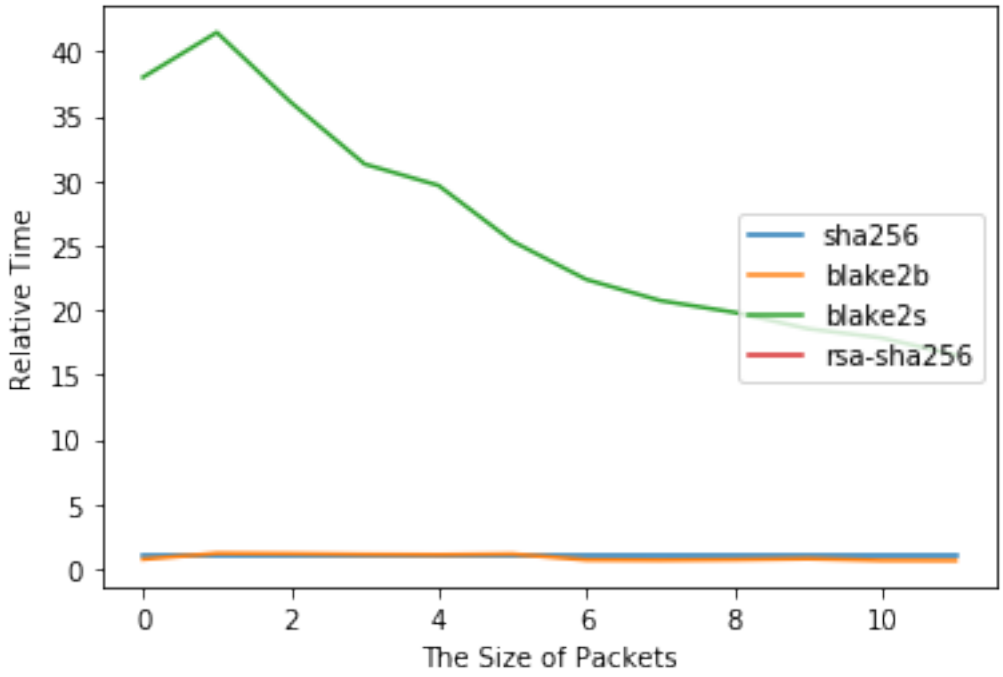
```
      90   0.046438  0.035101  0.030659    0.861546
      100  0.047903  0.030112  0.040748    0.854667
      110  0.051401  0.031687  0.040040    0.849015
```

[191]:
```
(0     38.006102
 1     41.478651
 2     36.078285
 3     31.289296
 4     29.638467
 5     25.318955
 6     22.363376
 7     20.731182
 8     19.820397
 9     18.552517
 10    17.841594
 11    16.517436
 Name: 0, dtype: float64,
            0
 0   53.123574
 1   35.308513
 2   31.490440
 3   28.657760
 4   27.769219
 5   22.583982
 6   33.561535
 7   32.077212
 8   29.047553
 9   24.544809
 10  28.382909
 11  26.793778,
            0
 0   70.682429
 1   60.143679
 2   53.906678
 3   47.620355
 4   45.048809
 5   36.776889
 6   33.426371
 7   31.913173
 8   28.892710
 9   28.100946
 10  20.974390
 11  21.204162)
```



[192]:
```
plot_result = pd.DataFrame(columns=names_of_hashes)
relative_performance = {}
for idx, hash_f in enumerate(hash_mapper):
    hash_mapper[idx].duration = pd.DataFrame(hash_mapper[idx].get_duration())
    hash_mapper[idx].duration.name = hash_mapper[idx].get_name()

    relative_performance.update({idx: hash_mapper[idx].get_duration()/
 ↪default_duration})
    alist = []
    for item in relative_performance[idx].values:
        alist.append(item)
    col_name = hash_mapper[idx].get_name()
    data = pd.Series(alist,name=col_name)
    plot_result[col_name] = data.astype(float)
```

[193]:
```
ax = plot_result.plot()
ax.set_xlabel("The Size of Packets")
ax.set_ylabel("Relative Time")
ax.legend(loc='center right')
plt.show()
```


Relative Time
The Size of Packets
sha256
blake2b
blake2s
rsa-sha256
0
5
10
15
20
25
30
35
40
0
2
4
6
8
10

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: