作业1：

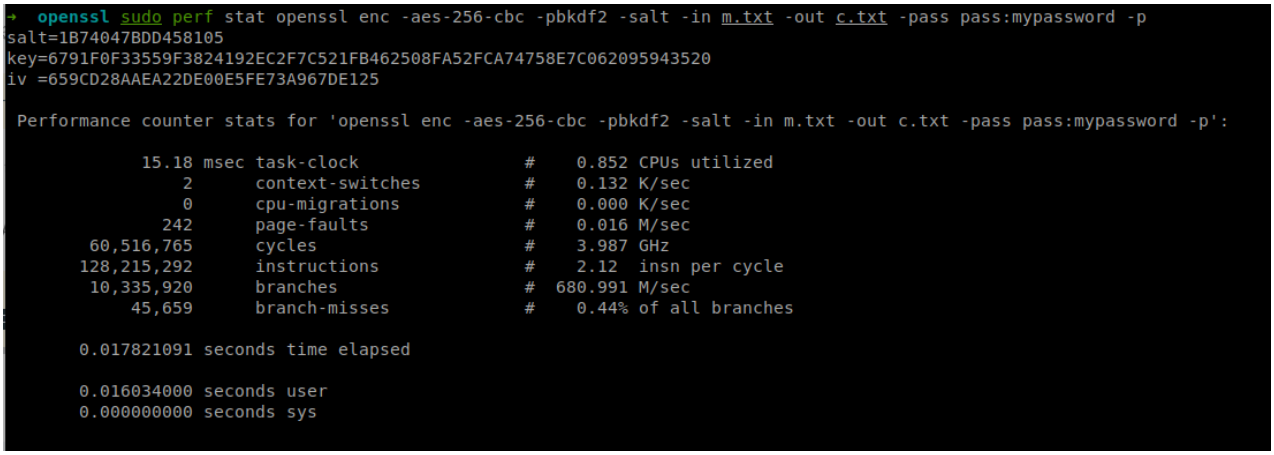
从网上搜集DES、AES、TEA和SM4的源码，并在本地环境编译

编写一个小程序，测试四种加密算法的加密速度（用cpu cycles方法，不要用时间）

分别测试CBC和GCM两种工作模式

AES-CBC

```
$ sudo perf stat openssl enc -aes-256-cbc -pbkdf2 -salt -in m.txt -out c.txt -pass pass:mypassword -p
```



AES-GCM

```
import json
from base64 import b64encode
```

```
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes

header = b"header"
data = b"secret"
key = get_random_bytes(16)
cipher = AES.new(key, AES.MODE_GCM)
cipher.update(header)
ciphertext, tag = cipher.encrypt_and_digest(data)

json_k = ['nonce', 'header', 'ciphertext', 'tag']
json_v = [
    b64encode(x).decode('utf-8')
    for x in [cipher.nonce, header, ciphertext, tag]
]
result = json.dumps(dict(zip(json_k, json_v)))
print(result)
```

```
→  aes-gcm sudo perf stat python3 e.py
Traceback (most recent call last):
  File "e.py", line 10, in <module>
    from Crypto.Cipher import AES
ModuleNotFoundError: No module named 'Crypto'

 Performance counter stats for 'python3 e.py':

            114.44 msec task-clock                #    0.990 CPU
s utilized
                13      context-switches          #    0.114 K/s
ec
                 0      cpu-migrations            #    0.000 K/s
ec
             4,160      page-faults               #    0.036 M/s
ec
       481,547,393      cycles                    #    4.208 GHz

       468,188,552      instructions              #    0.97  ins
n per cycle
       102,158,341      branches                  #  892.714 M/s
ec
         3,688,782      branch-misses             #    3.61% of
all branches

       0.115560746 seconds time elapsed

       0.098786000 seconds user
       0.015805000 seconds sys
```

DES-CBC

```
$ sudo perf stat openssl enc -des-cbc -pbkdf2 -salt -in m.txt -out c.txt -pass
pass:mypassword -p
```

```
→  openssl sudo perf stat openssl enc -des-cbc -pbkdf2 -salt -in m.txt -out c.txt -pass pass:mypassword -p
salt=C05DECAF1C093BF1
key=2EEC80E68C923E33
iv =E9F467749C14E17D

 Performance counter stats for 'openssl enc -des-cbc -pbkdf2 -salt -in m.txt -out c.txt -pass pass:mypassword -p':

            8.46 msec task-clock                #    0.850 CPUs utilized
              11      context-switches          #    0.001 M/sec
               0      cpu-migrations            #    0.000 K/sec
             242      page-faults               #    0.029 M/sec
      31,382,577      cycles                    #    3.711 GHz
      67,889,520      instructions              #    2.16  insn per cycle
       5,864,017      branches                  #  693.349 M/sec
          46,278      branch-misses             #    0.79% of all branches

      0.009946480 seconds time elapsed

      0.002911000 seconds user
      0.005822000 seconds sys
```

DES-GCM

无

原因：因为GCM是为区块长度为128bits设计的，而DES或者3DES的区块长度是64bits，也有猜测是因为GCM的MAC组件的安全性取决于最长的消息中的块数与多项式字段中的元素数之差。

参考链接

TEA-CBC

测试源代码来源（c语言）：Github

```
→  drakon-tea git:(master) ✗ sudo perf stat ./tea
Encrypted: ��
d���&f/#
Decrypted: Hello world!

 Performance counter stats for './tea':

              0.34 msec task-clock                #    0.417 CPUs utilized
                 0      context-switches          #    0.000 K/sec
                 0      cpu-migrations            #    0.000 K/sec
                53      page-faults               #    0.156 M/sec
         1,463,705      cycles                    #    4.304 GHz
           767,752      instructions              #    0.52  insn per cycle
           149,662      branches                  #  440.039 M/sec
             7,010      branch-misses             #    4.68% of all branches

       0.000815375 seconds time elapsed

       0.000852000 seconds user
       0.000000000 seconds sys
```

TEA-GCM

待补充

SM4-CBC

```
$ sudo perf stat openssl enc -sm4-cbc -pbkdf2 -salt -in m.txt -out c.txt -pass
pass:mypassword -p
```

```
→  openssl sudo perf stat openssl enc -sm4-cbc -pbkdf2 -salt -in m.txt -out c.txt -pass pass:mypassword -p
salt=279984BA2F8A8E3C
key=16C7CAC6CBBC833D39E5B39D9935F51C
iv =91E7B251924FB6B609CAA86D55EBE953

 Performance counter stats for 'openssl enc -sm4-cbc -pbkdf2 -salt -in m.txt -out c.txt -pass pass:mypassword -p':

              7.23 msec task-clock                #    0.940 CPUs utilized
                 0      context-switches          #    0.000 K/sec
                 0      cpu-migrations            #    0.000 K/sec
               241      page-faults               #    0.033 M/sec
        28,090,025      cycles                    #    3.884 GHz
        68,644,562      instructions              #    2.44  insn per cycle
         5,982,227      branches                  #  827.131 M/sec
            43,634      branch-misses             #    0.73% of all branches

       0.007693684 seconds time elapsed

       0.007699000 seconds user
       0.000000000 seconds sys
```

SM4-GCM

待补充

从测试过程中观察，各个加密算法中对性能影响最大的子模块（函数）是哪个？

无测试，待补充。

cipher参考链接

PyCryptodome

perf参考链接

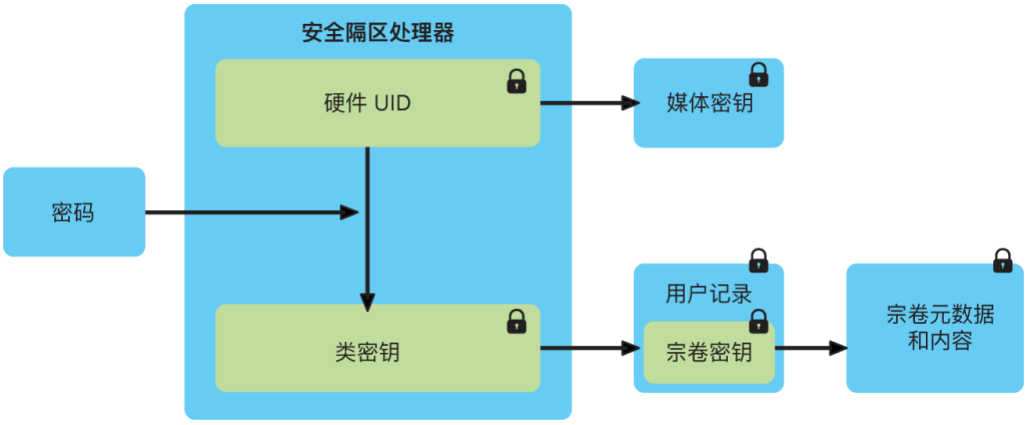
- How to Use Perf Performance Analysis Tool on Ubuntu 20.04

- 系统级性能分析工具perf的介绍与使用
- perf - check the performance arguments of a program
- 虚拟机Linux使用perf stat提示cycles not supported
- Linux perf-top basics: understand the %

作业2

手机端的随机数产生源:

- 软件随机数生成:
  - 操作系统随机数生成
    - 如**Apple**提供了内核中的受信任软件**CPRNG**, 聚合系统中的原始熵并为内核和用户控件的使用者提供安全的随机数. 作为熵源的因素包括:
      - 安全隔区的硬件 RNG
      - 启动过程中所收集基于时序的时间误差
      - 从硬件中断收集的熵
      - 用于启动过程中保持熵的种子文件
      - Intel 随机指令，即 RDSEED 和 RDRAND（仅限 macOS）
  - 应用内置随机数生成
    - 如微信中的摇骰子功能, 经过测试, 在无互联网连接的时候经过一段等待时间后也会得出结果, 再次连接互联网, 重发此消息, 又会得到不同结果. 我们认为在正常情况下摇骰子应为向服务器请求结果, 但离线时也会得出结果, 说明内置了随机数生成的功能.

- 硬件随机数生成:
  - 如Apple设备会将密钥放置在一个安全隔区片上系统(SoC)中加密,这之中就包含了一个随机数发生器
  - 

安全隔区处理器。
  - 三星于2020年发布的`Galaxy A Quantum`手机配备了QRNG芯片(QRNG),利用CMOS图像传感器捕获的光源散粒噪声产生随机序列

我们认为 三星的QRNG和苹果内置的HRNG可以被认为是"真"随机数