By : Mahmoud Hany Fathalla 20104375

Image encryption and Signature

Project Implementation Overview

The project implements two distinct approaches for image encryption and digital signing . Both methods perform identical cryptographic operations but differ in execution methodology.

Manual OpenSSL Approach

The manual approach uses step-by-step terminal commands for educational purposes . This method generates AES keys and IVs, then encrypts images in four different modes, demonstrating the practical application of each encryption mode.

Automated Python Implementation

The Python script automates the entire process, converting images and performing all encryption operations in a single execution. The automated approach creates organized output files.

Key Observations and Statistical Analysis

NIST Randomness Test Results

The most significant findings come from the statistical validation using NIST randomness tests. The ECB mode results show interesting patterns . Most tests pass with acceptable p-values, but Maurer's Universal Statistical Test consistently fails with a p-value of -1.0, indicating non-random behavior.

Comparison with Unencrypted Data

The contrast between encrypted and unencrypted data is striking. The original image data shows clear non-random patterns . Multiple tests fail catastrophically with p-values near zero, demonstrating the structured nature of image data.

Surprising and Interesting Findings

ECB Mode Vulnerability Demonstration

The most educationally valuable observation is ECB mode's statistical weakness. While most randomness tests pass, the Universal Statistical Test failure demonstrates why ECB is unsuitable for image encryption, as it preserves patterns from the original data.

Cross-Mode Performance Consistency

All non-ECB modes (CBC, CFB, OFB) show remarkably similar statistical performance, with most tests producing p-values well above the 0.01 significance threshold, indicating strong cryptographic randomness.

Binary Conversion Pipeline

The project includes a sophisticated testing pipeline that converts encrypted images to binary format for statistical analysis . This demonstrates the practical application of cryptographic validation techniques.

Notes

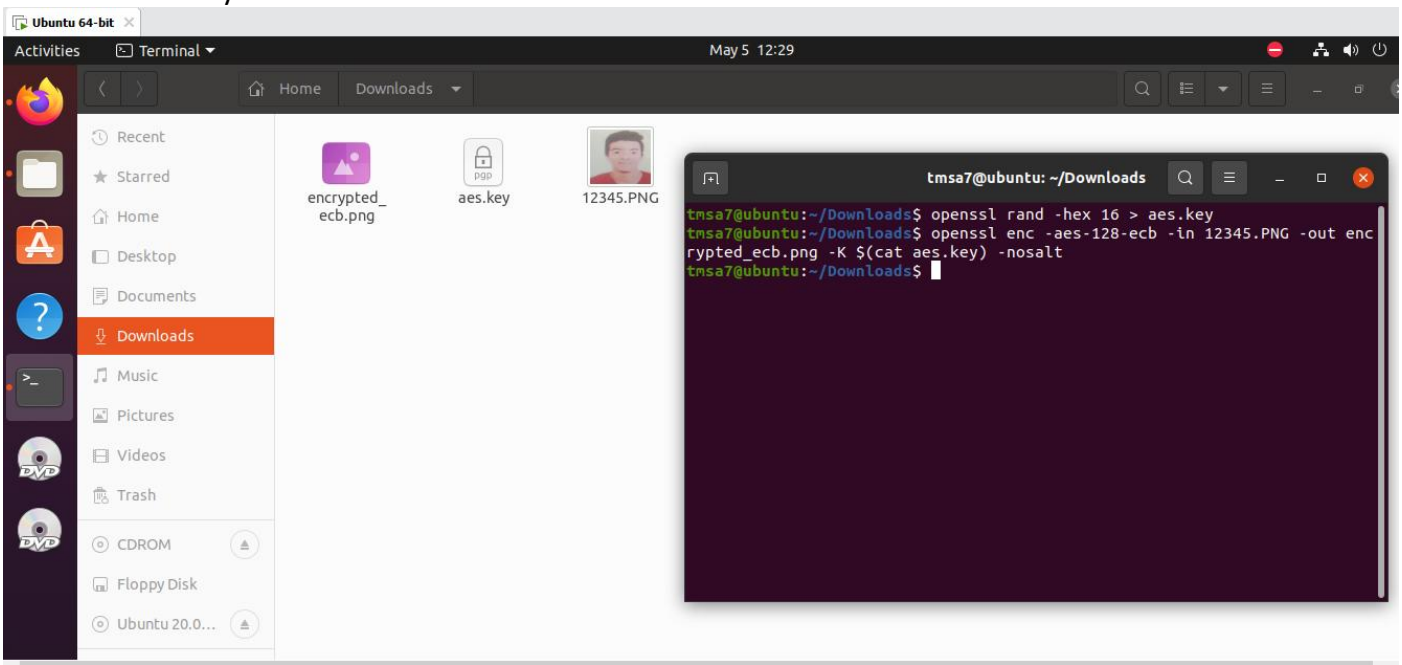
This project effectively demonstrates the practical differences between AES encryption modes through both implementation and statistical validation. The comprehensive testing approach using NIST standards provides scientific evidence of encryption effectiveness, making it an excellent educational tool for understanding cryptographic principles and their real-world implications.

By : Mahmoud Hany Fathalla 20104375

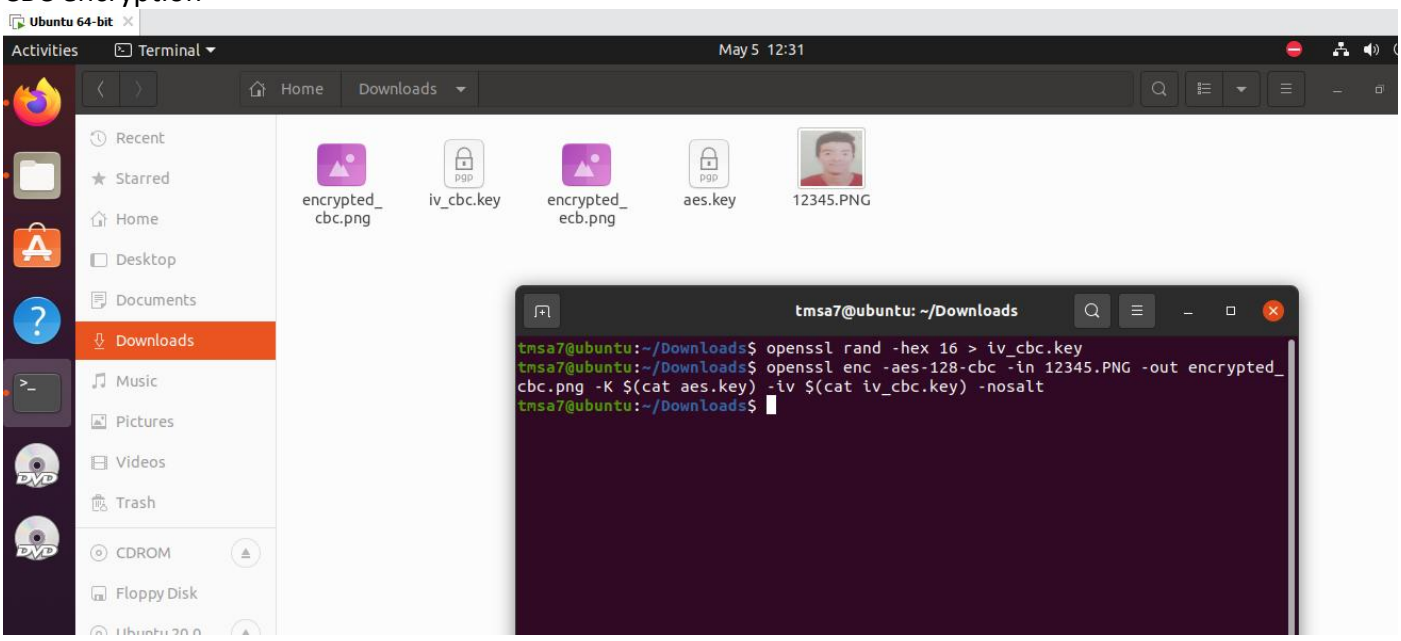
Manual OpenSSL Approach Implementation :

Task 1:

Generate AES Key & ECB ENCRYPTION

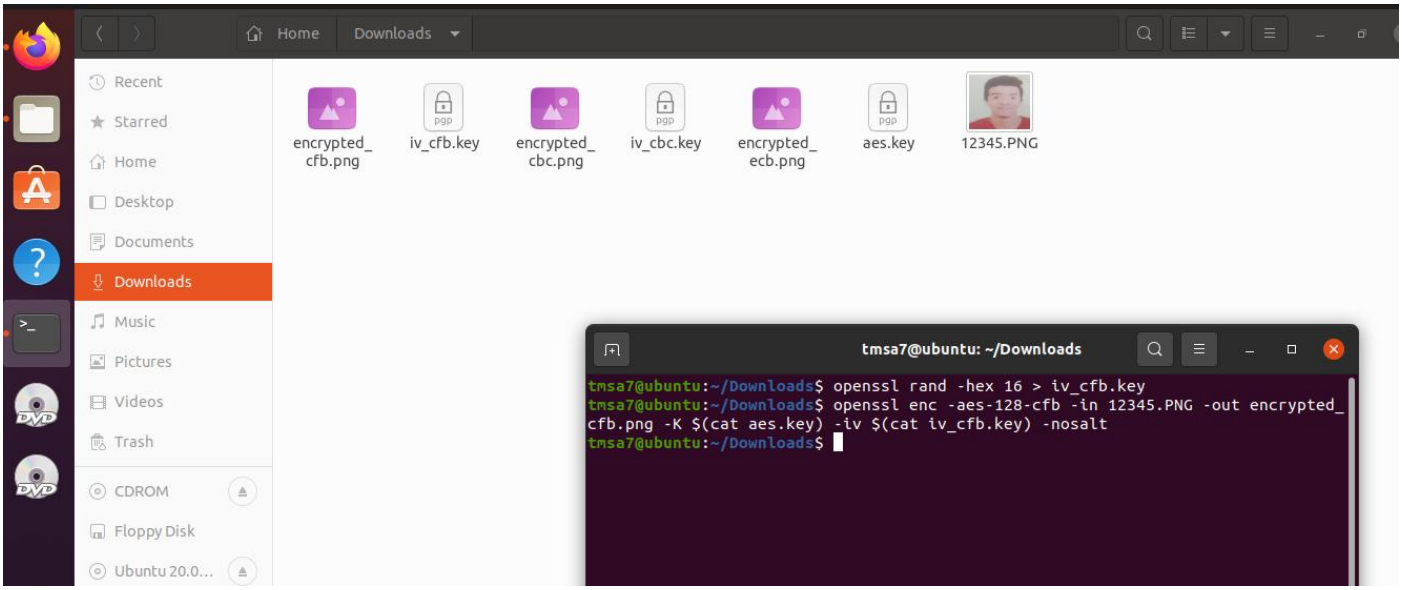


CBC encryption

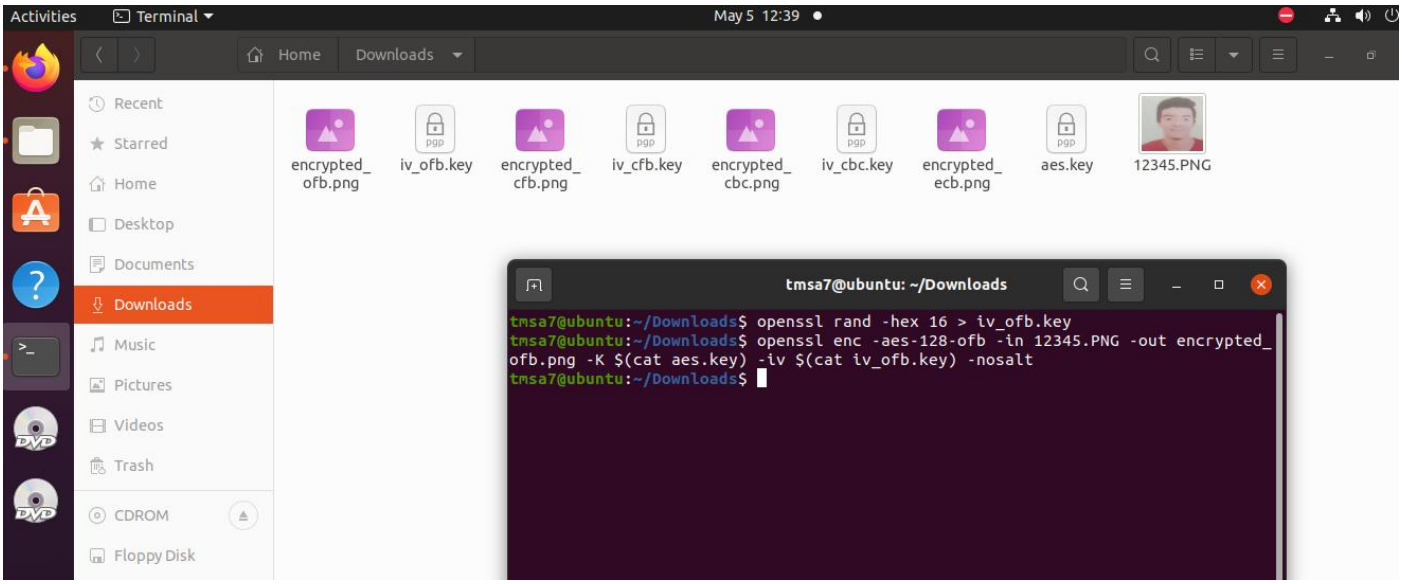


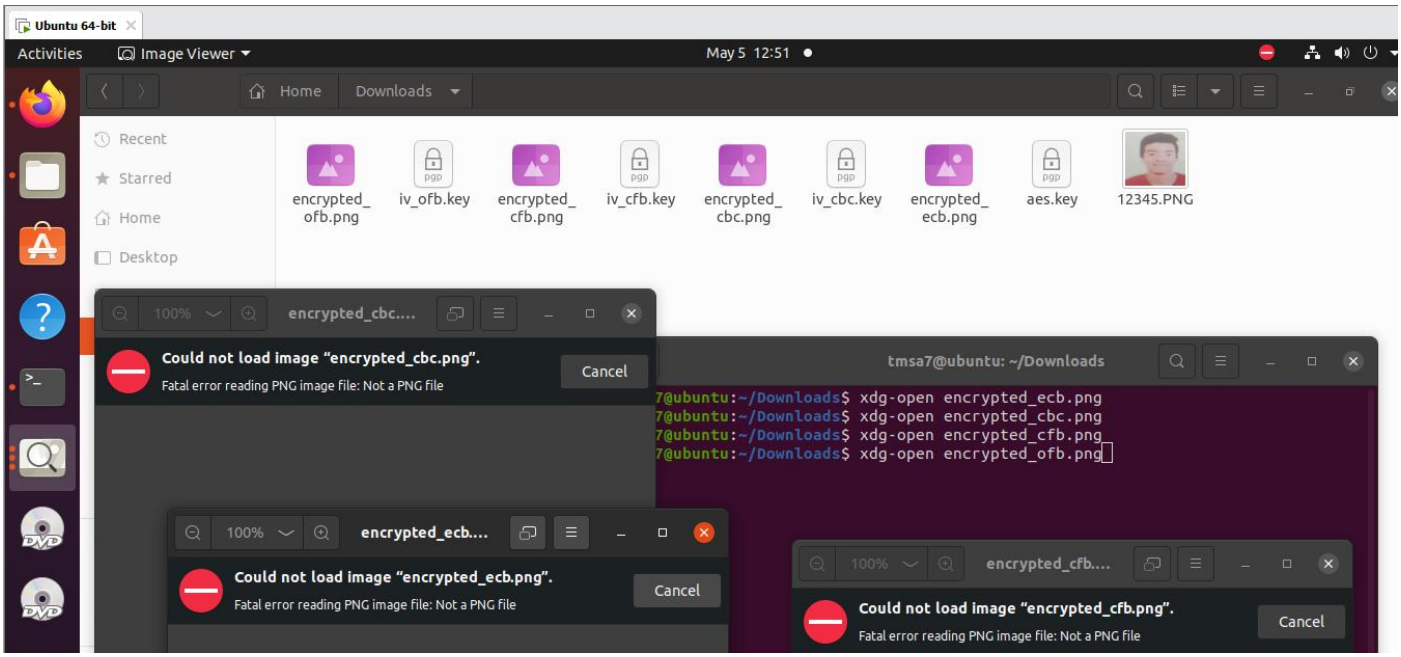
By : Mahmoud Hany Fathalla 20104375

CFB ENCRYPTION



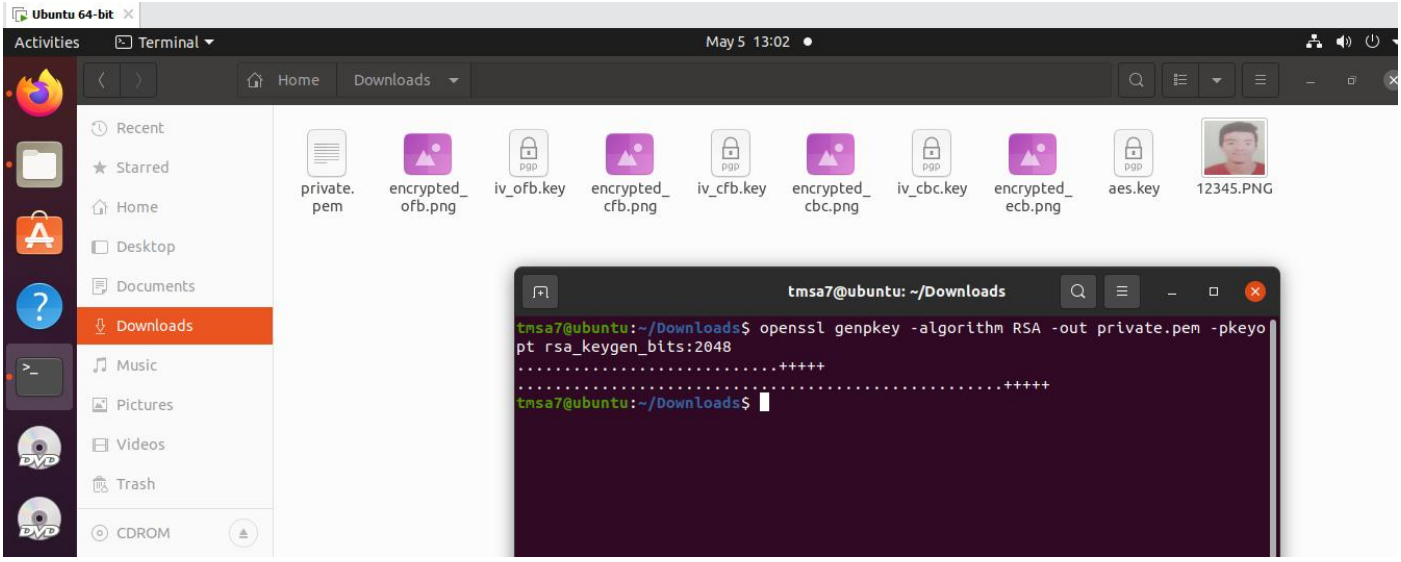
OFB ENCRYPTION

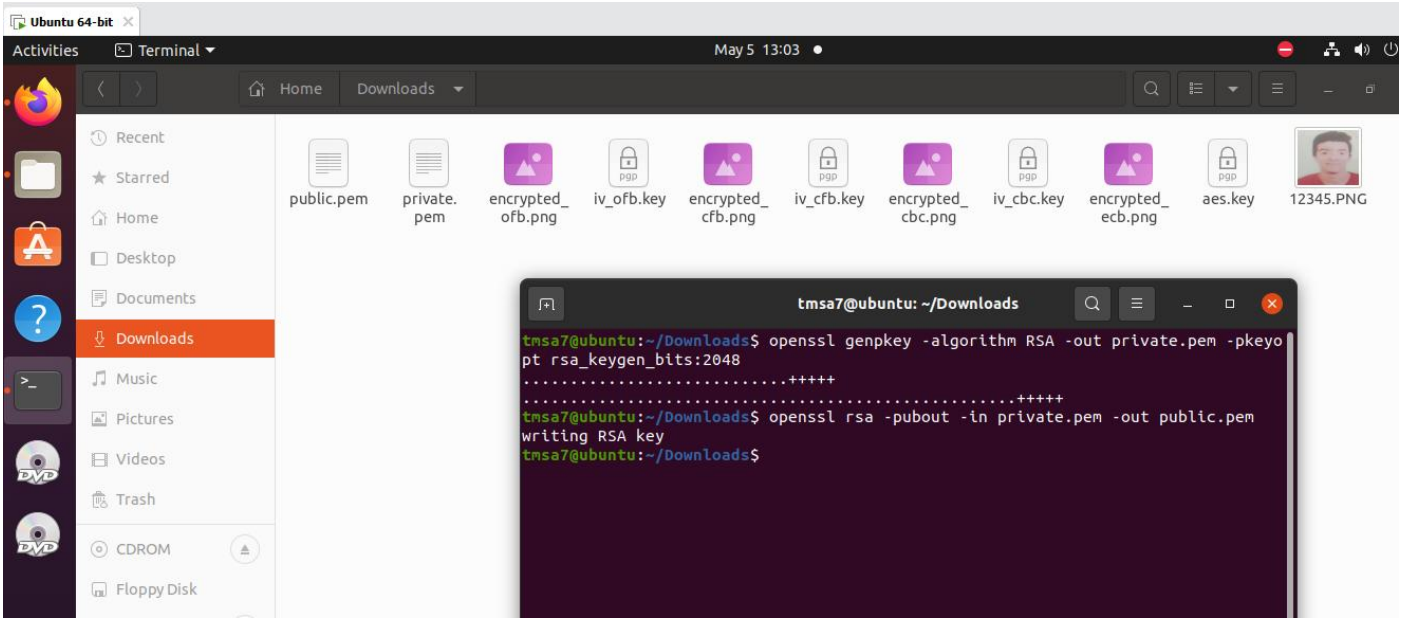




By : Mahmoud Hany Fathalla 20104375

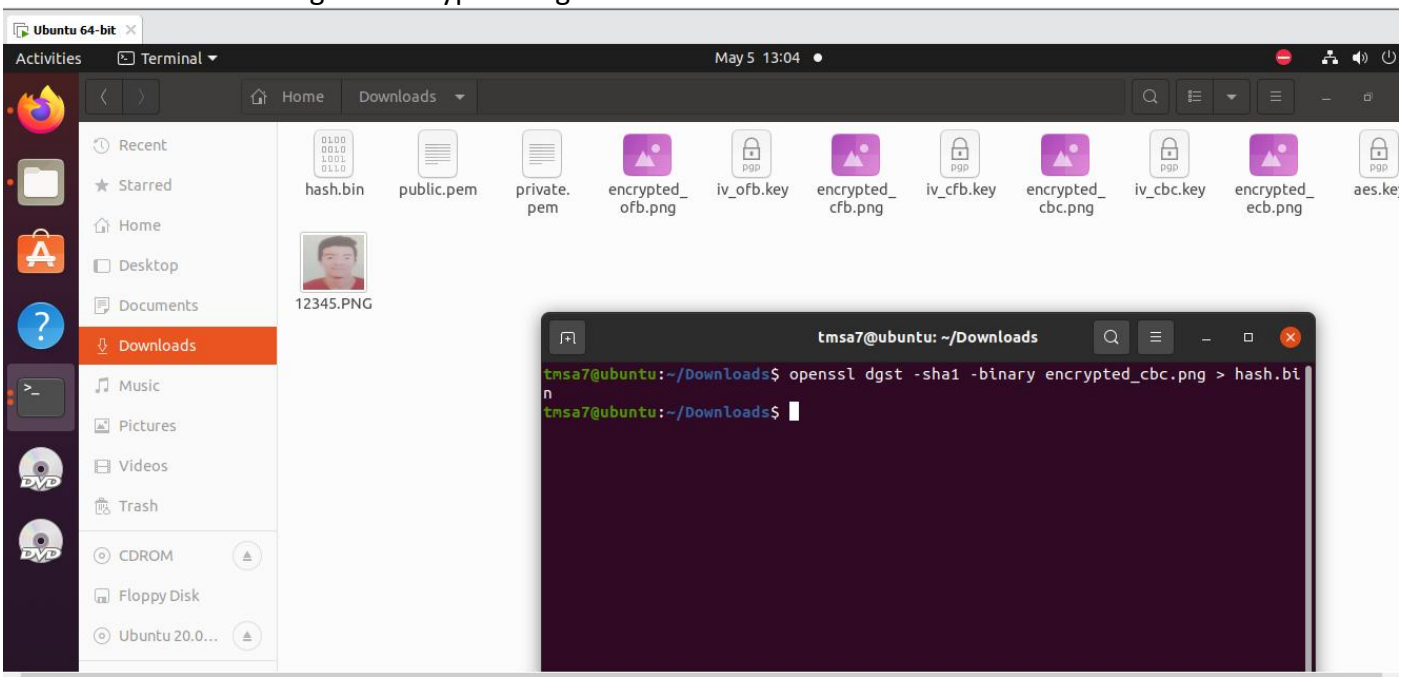
Task 2:
Generate RSA private & Public keys



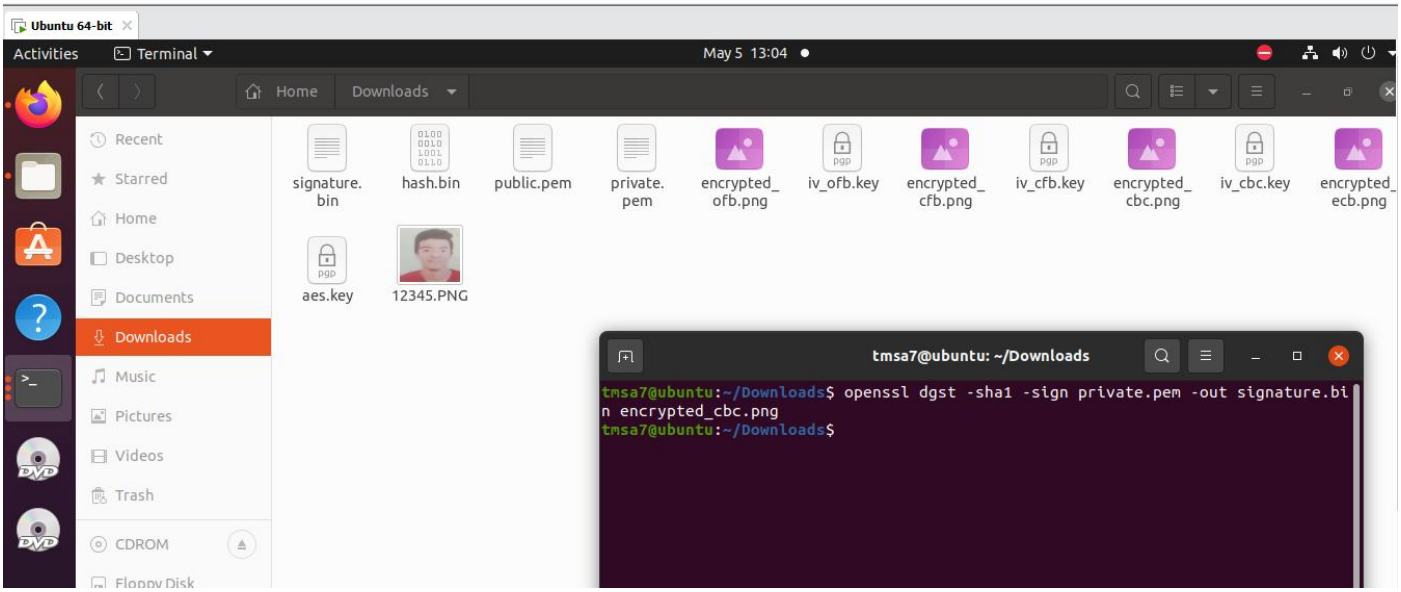


By : Mahmoud Hany Fathalla 20104375

Get the SHA1 Hash using cbc encrypted img

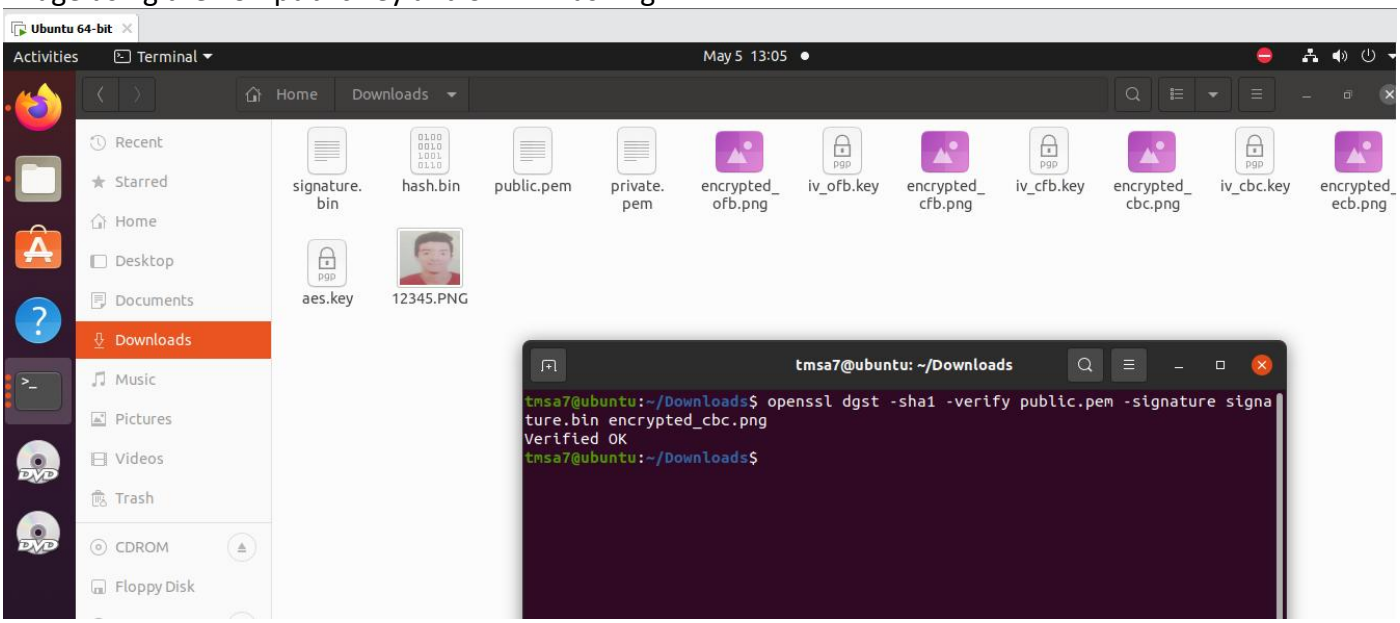


Get the Signature



By : Mahmoud Hany Fathalla 20104375

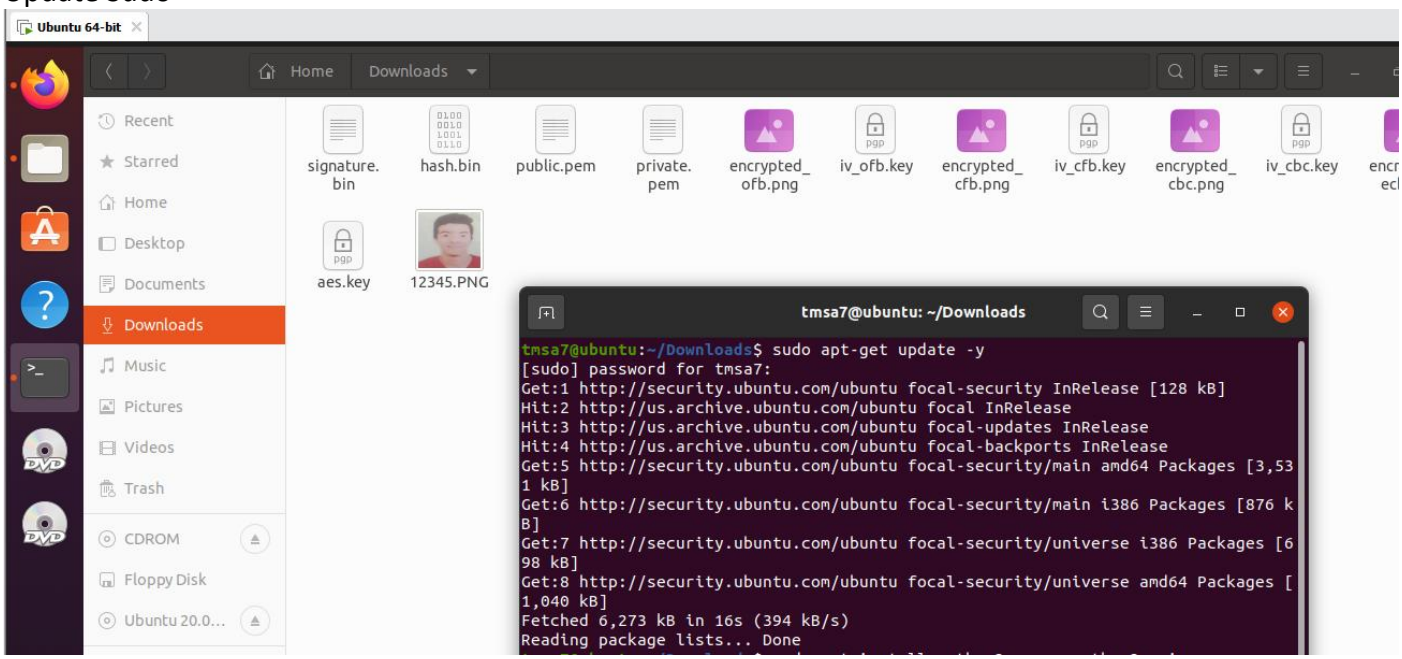
Uses command-line verification with . This command verifies the signature against the CBC-encrypted image using the RSA public key and SHA-1 hashing.



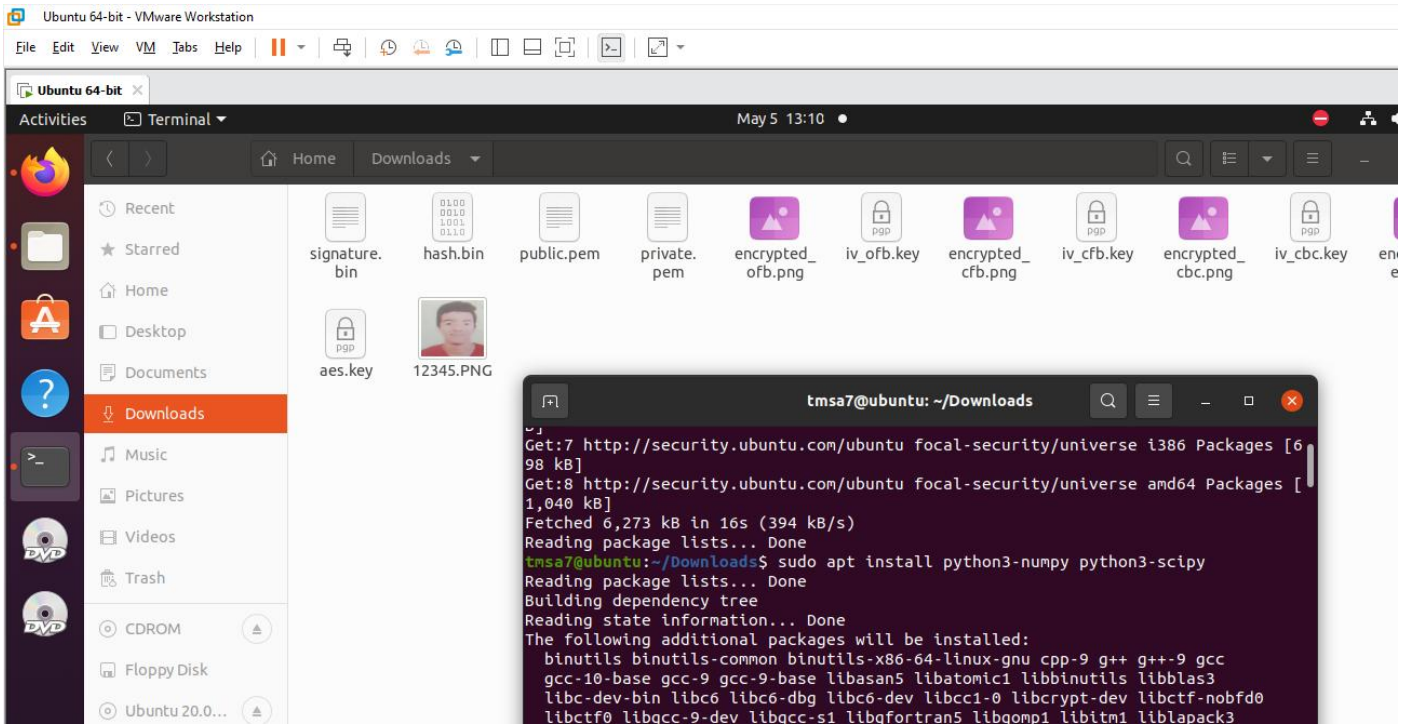
By : Mahmoud Hany Fathalla 20104375

Task 3 (Bonus)

Update Sudo

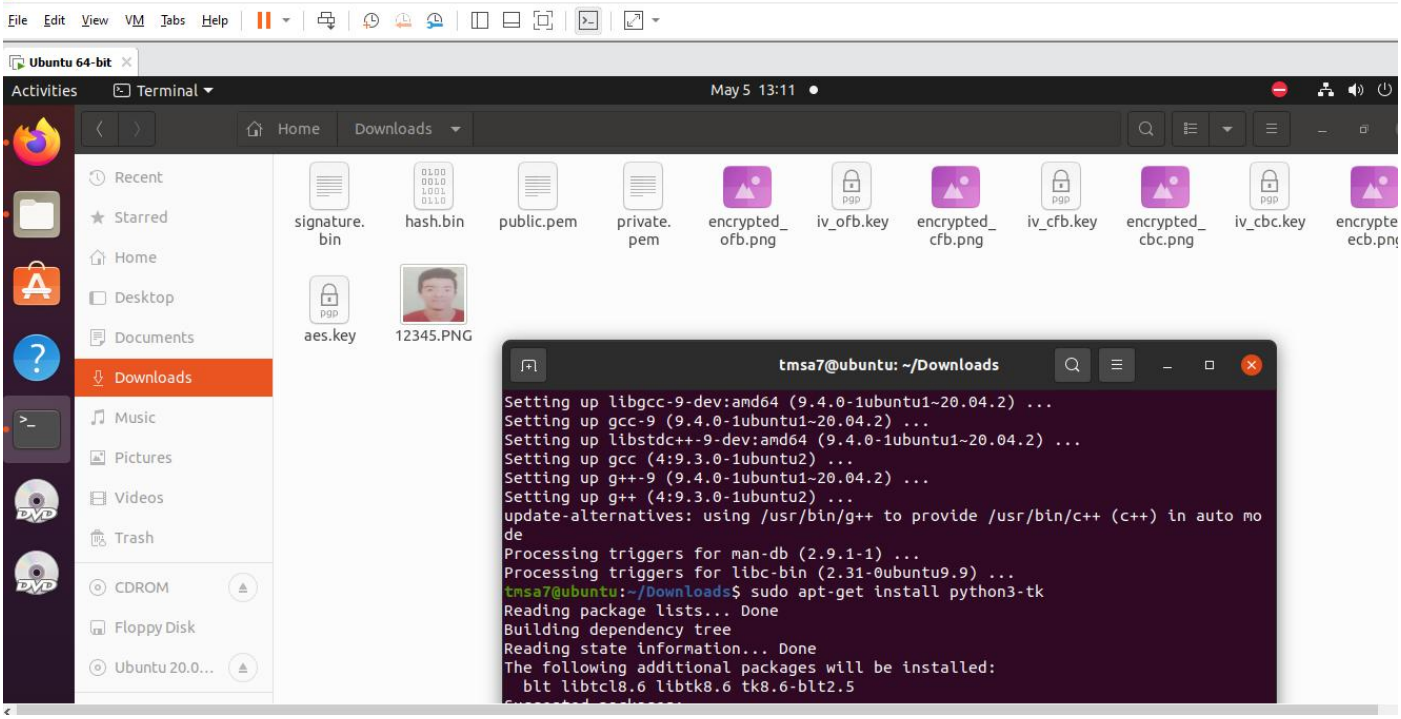


Install Numby

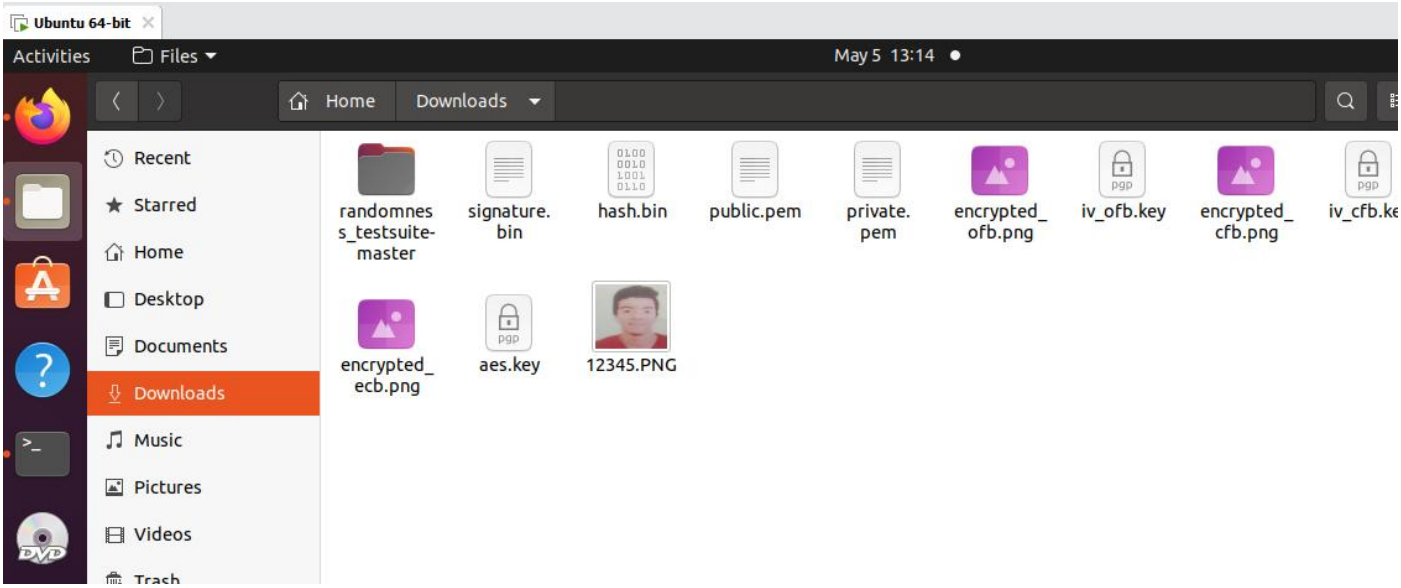


By : Mahmoud Hany Fathalla 20104375

Instal Tkenter

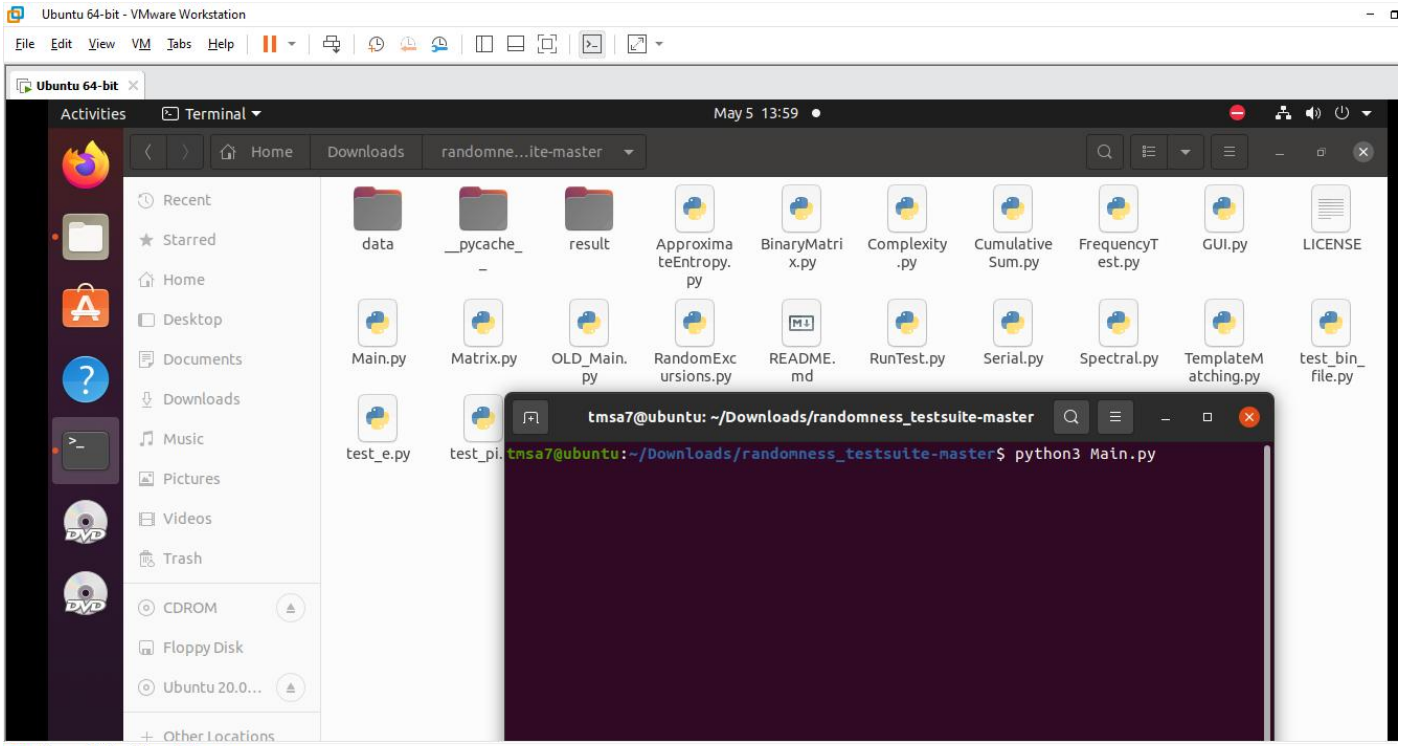


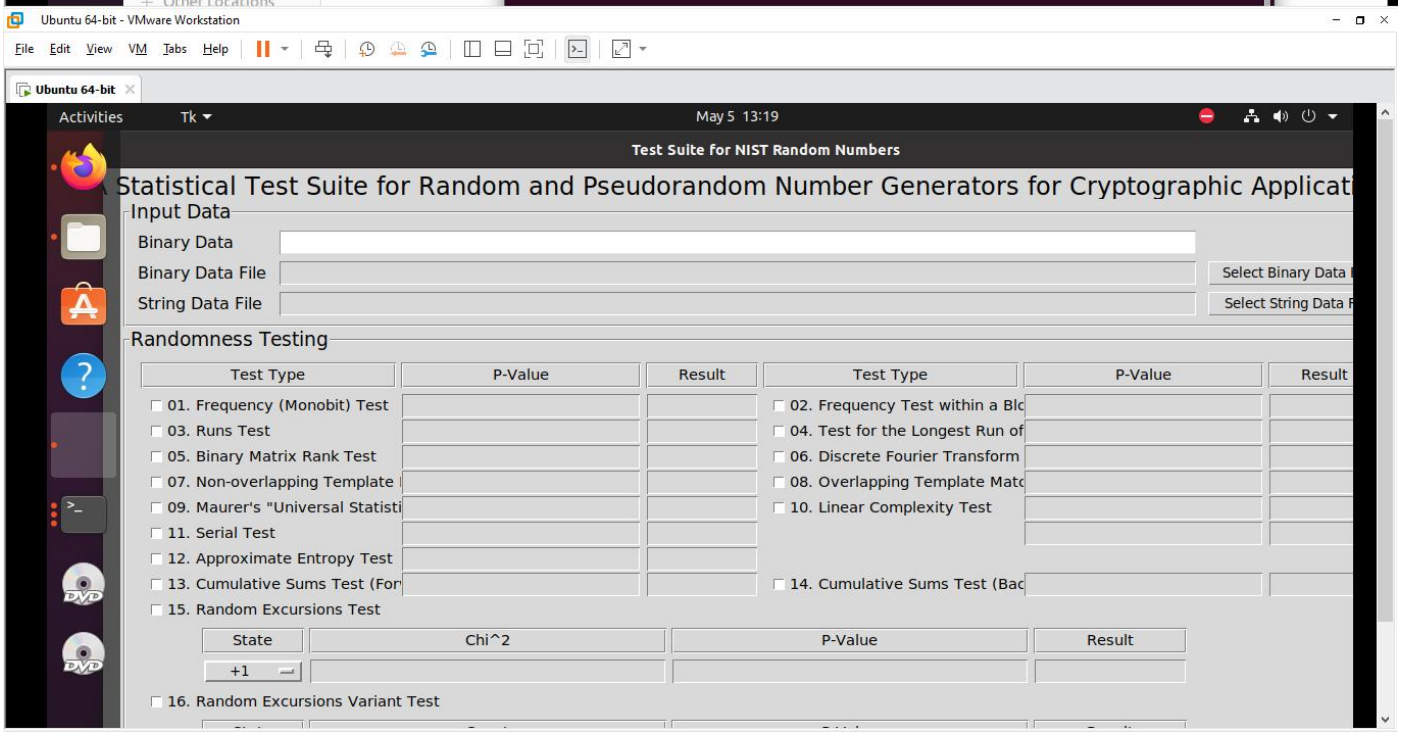
Clone The Ranomness_testsuite_master from git



By : Mahmoud Hany Fathalla 20104375

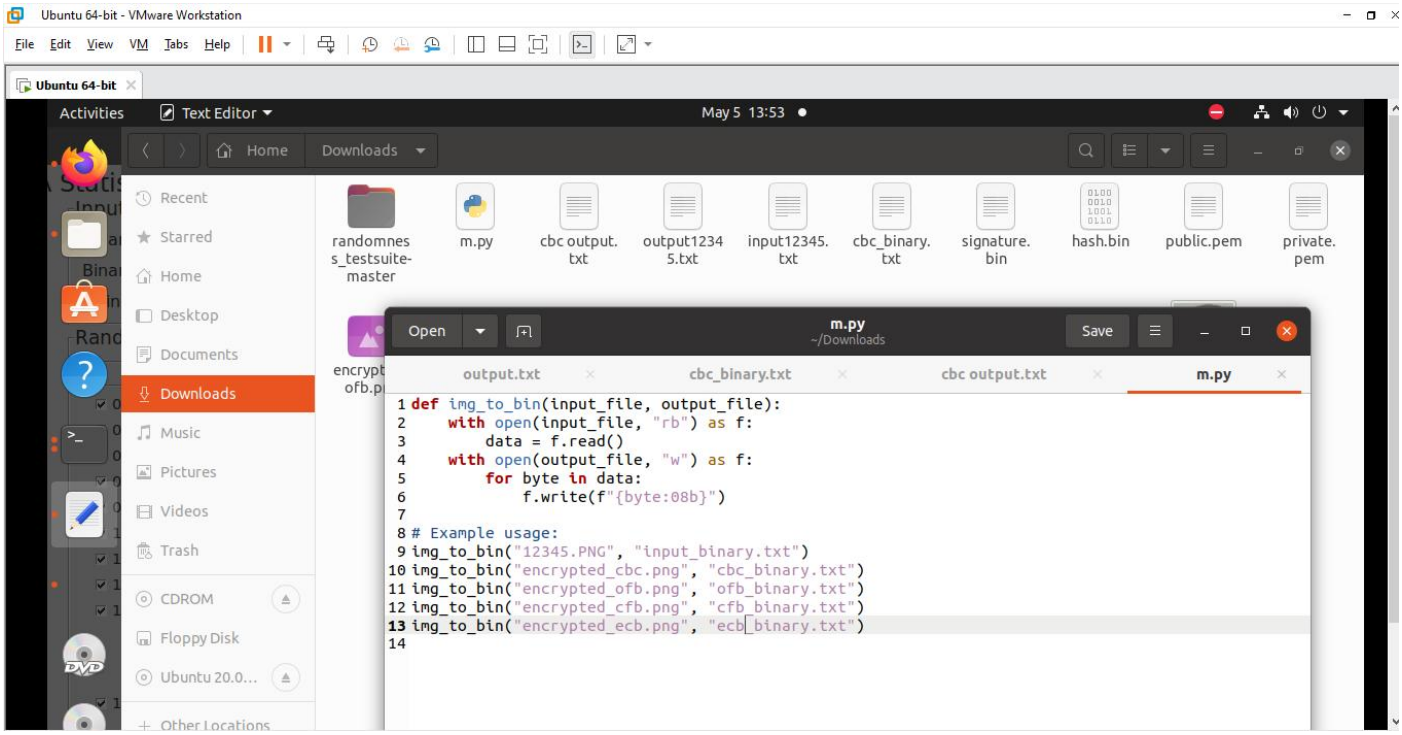
Run the GUI



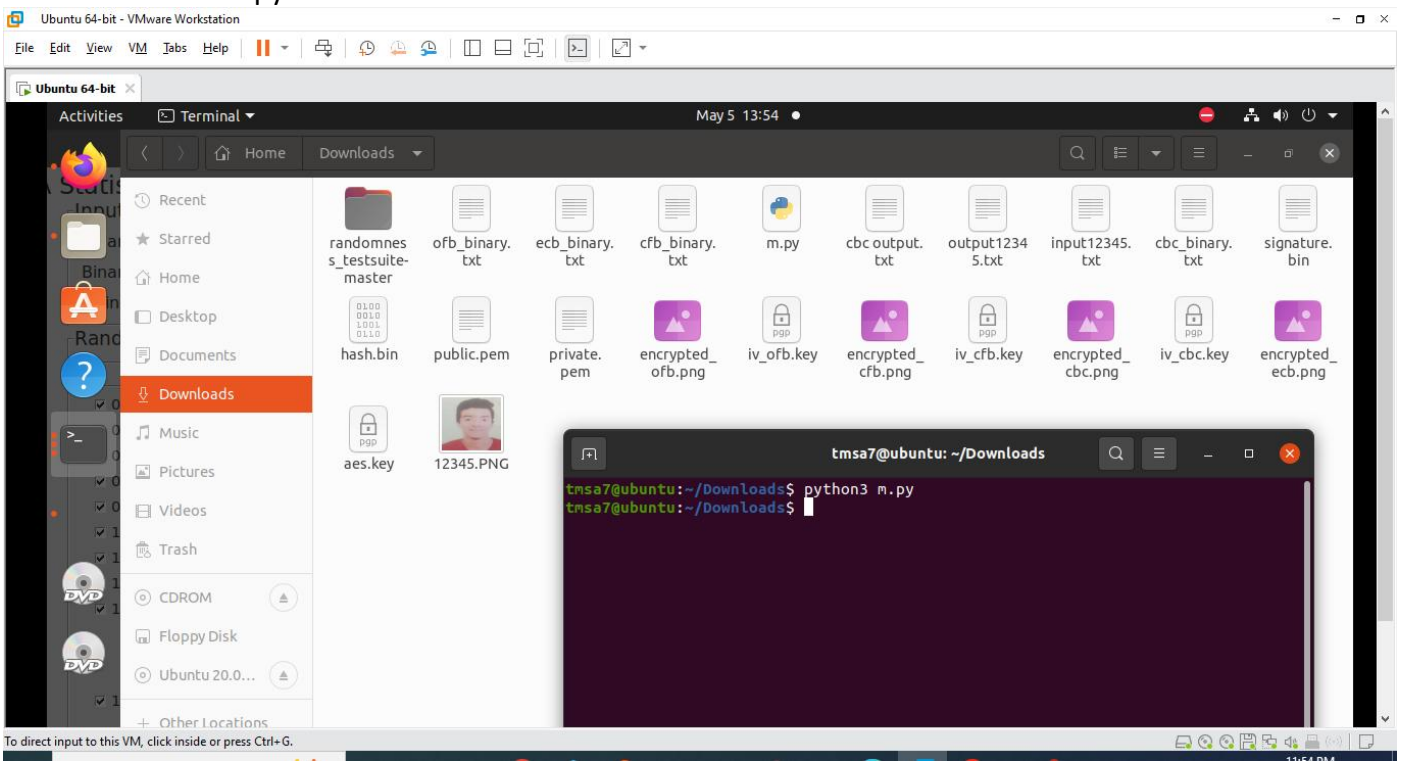


By : Mahmoud Hany Fathalla 20104375

Wrote a python script to convert the png and encrypted imgs to binary to apply test on it

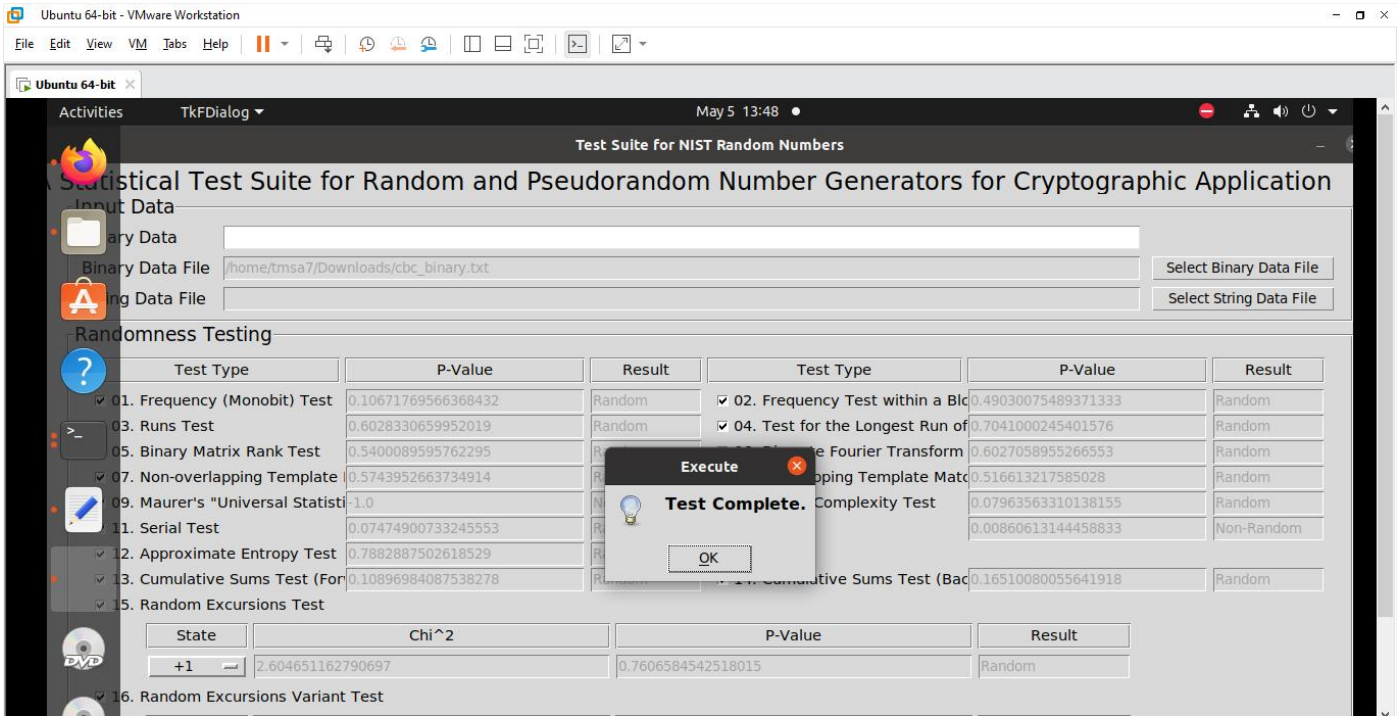


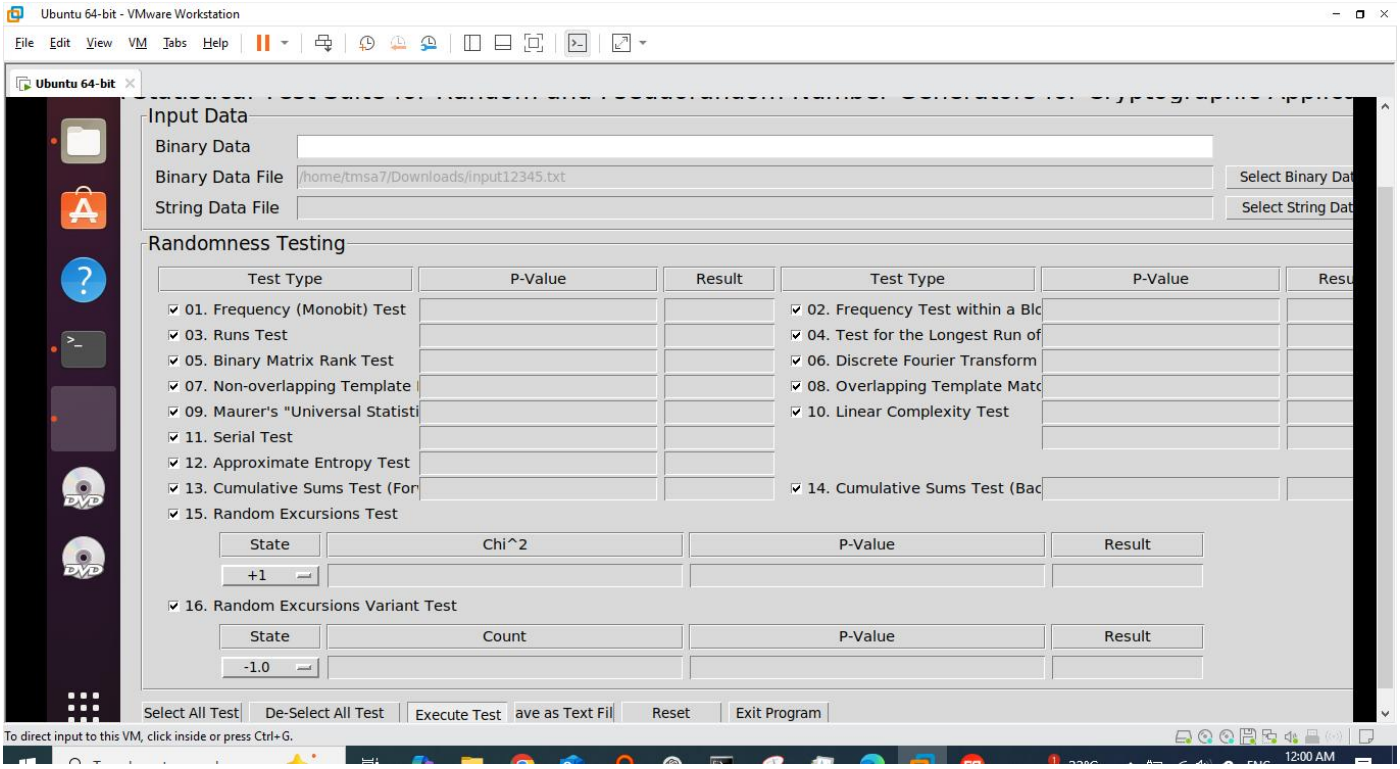
Run the converter .py



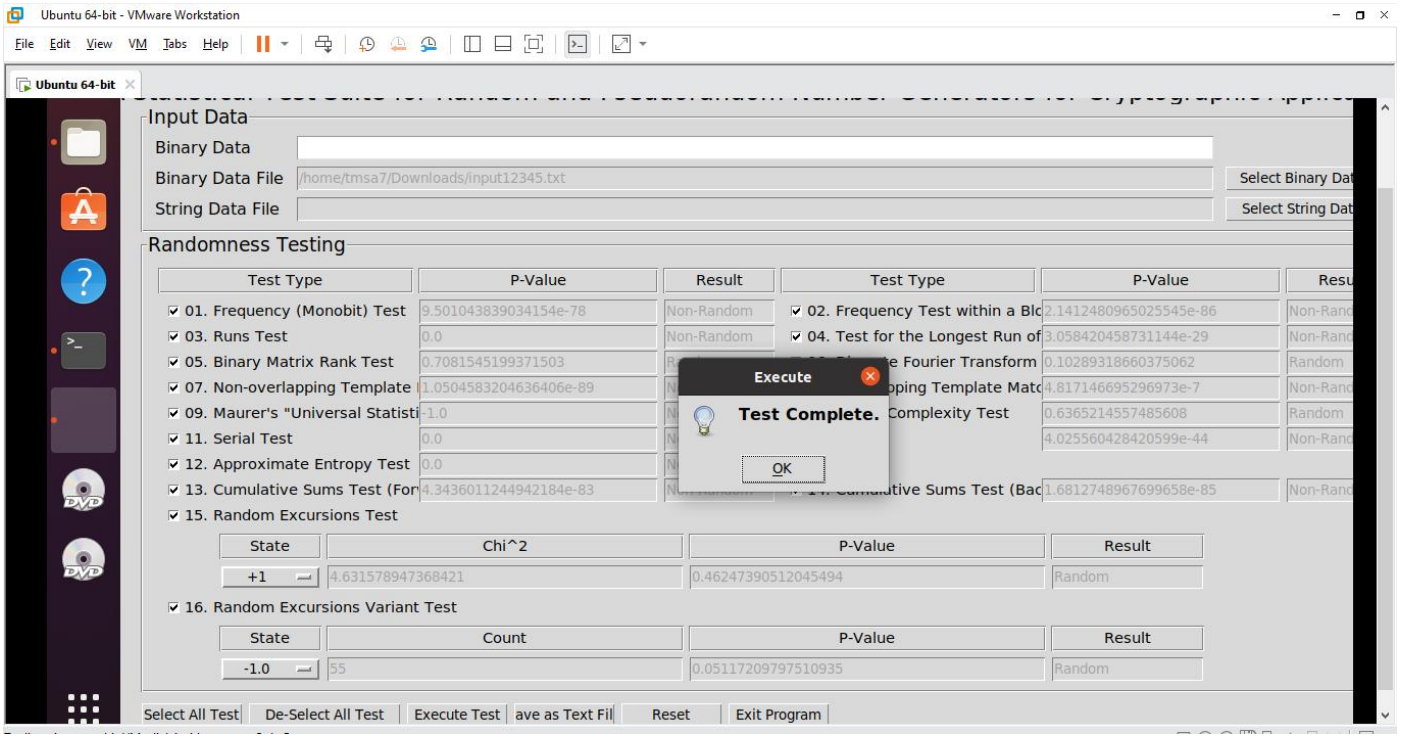
By : Mahmoud Hany Fathalla 20104375

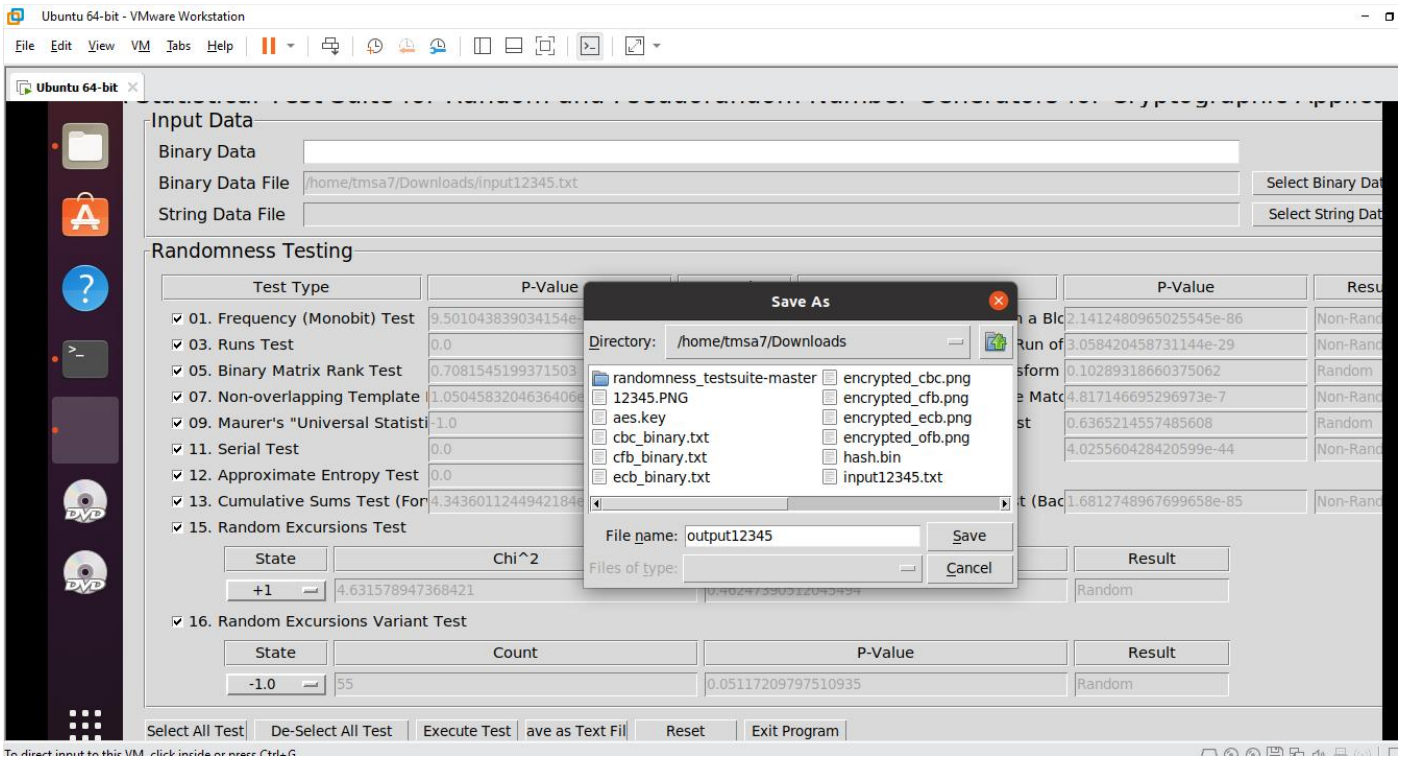
Selecting the input binary img and all the encrypted binary imgs and applay testing on them then save the results as ouput.txt file



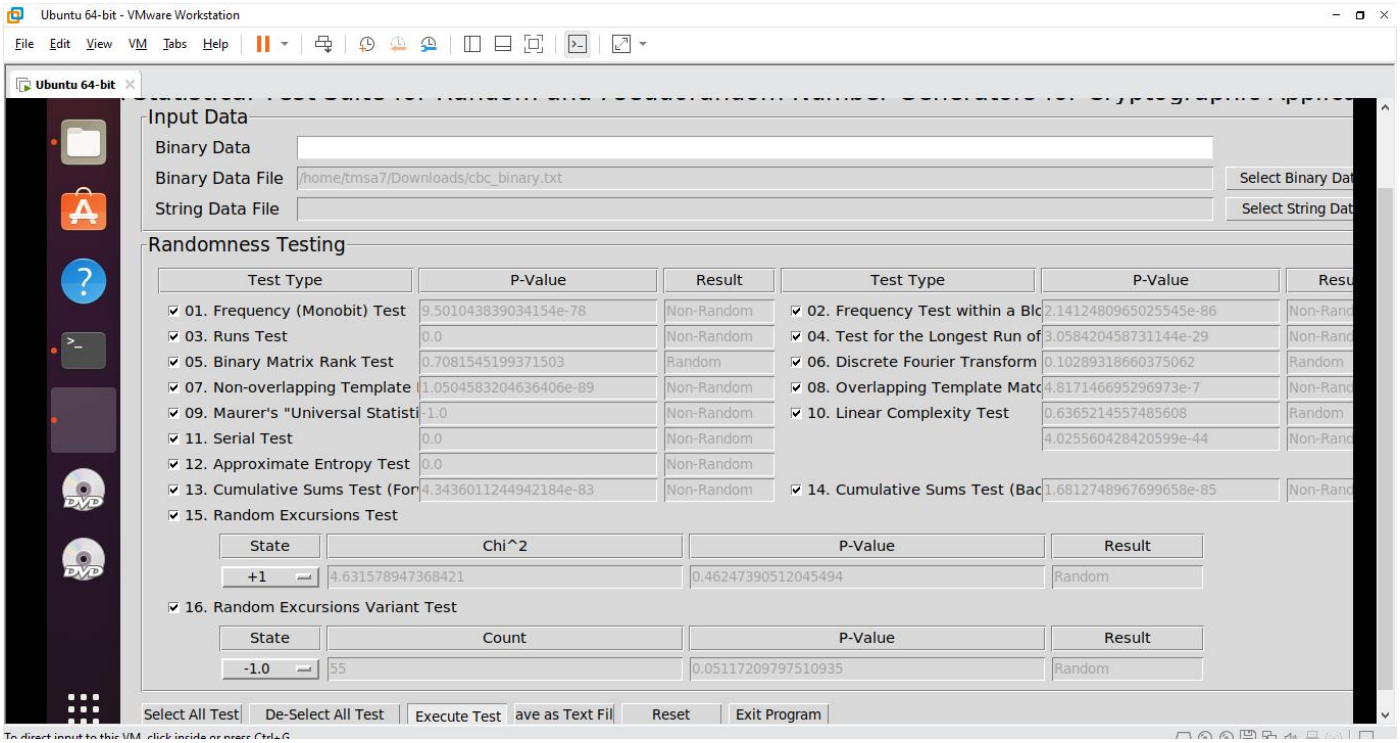


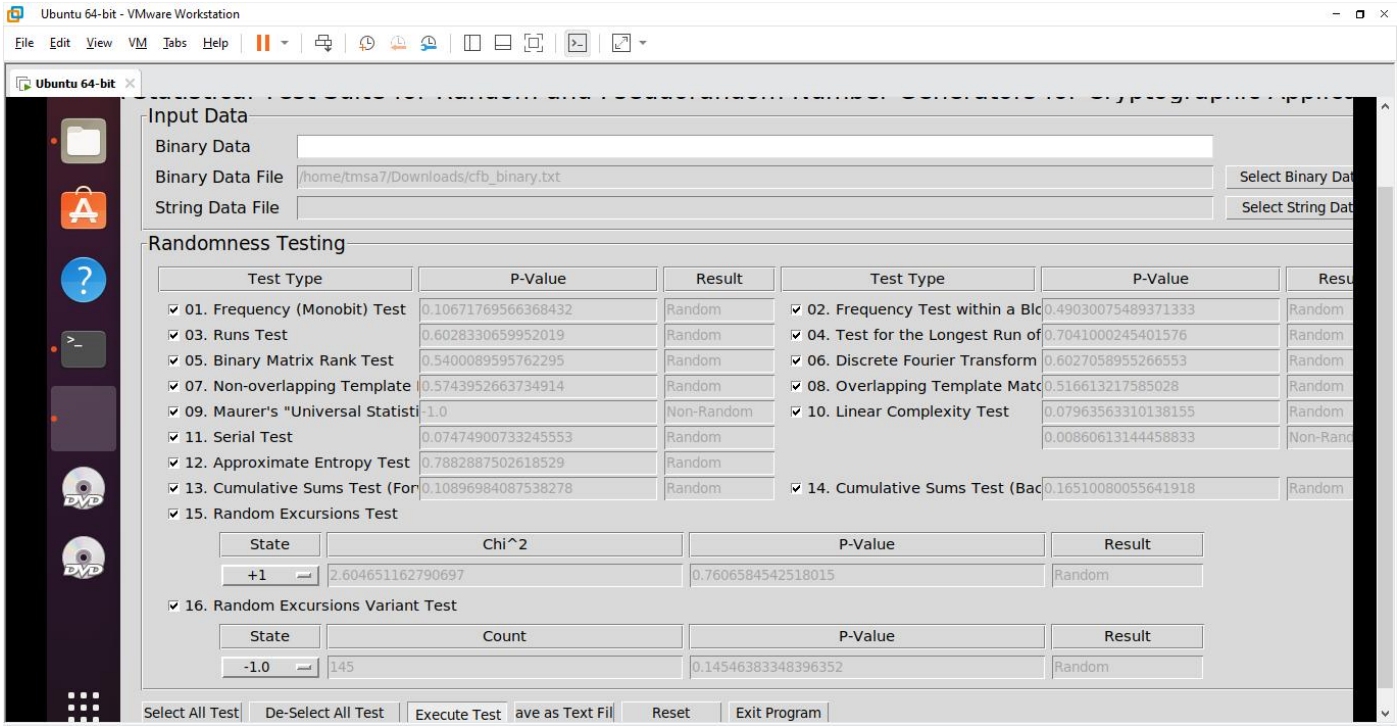
By : Mahmoud Hany Fathalla 20104375



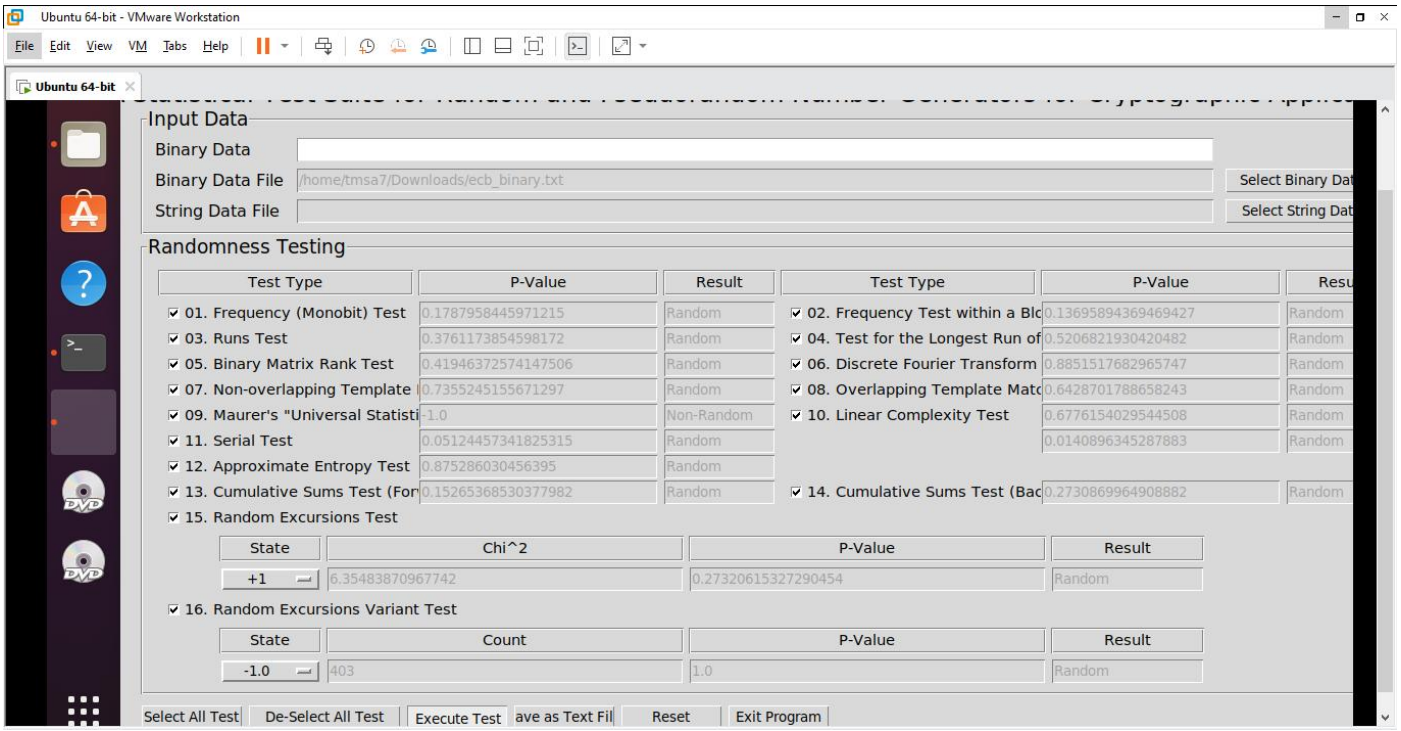


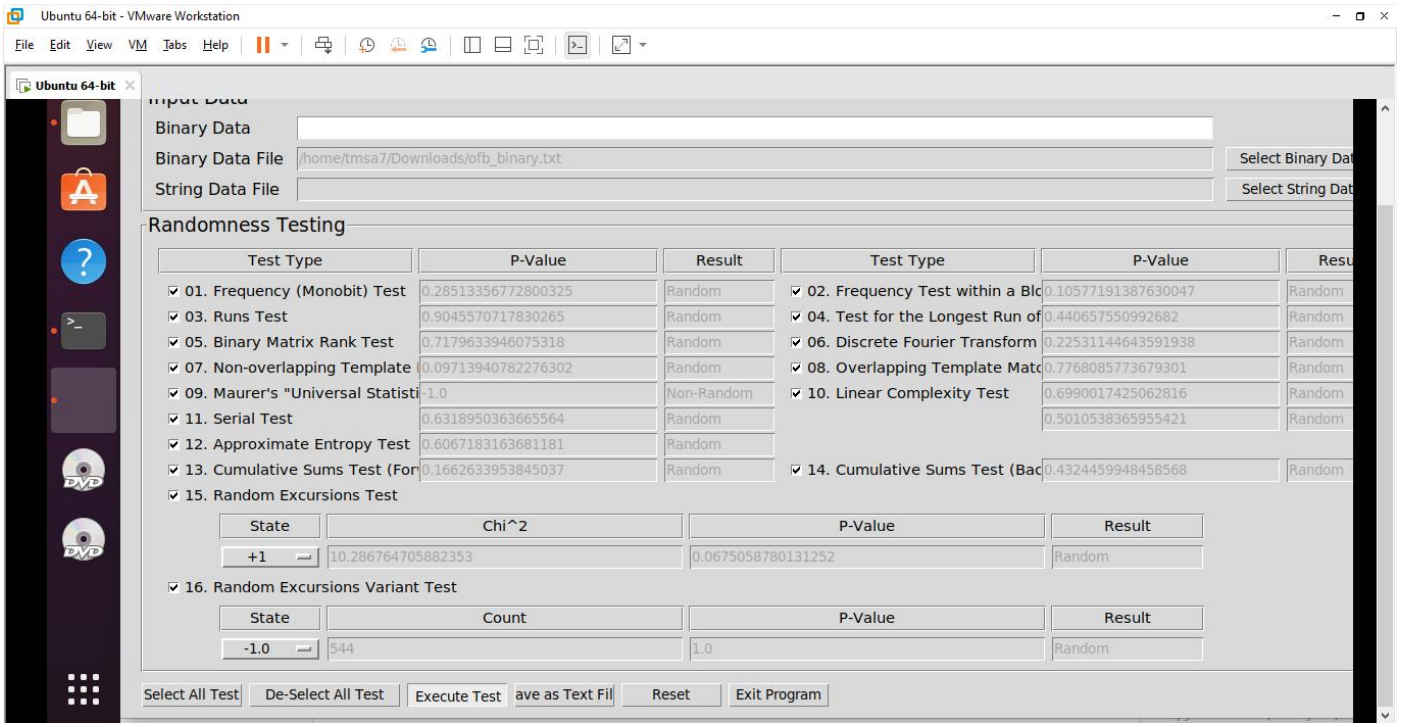
By : Mahmoud Hany Fathalla 20104375





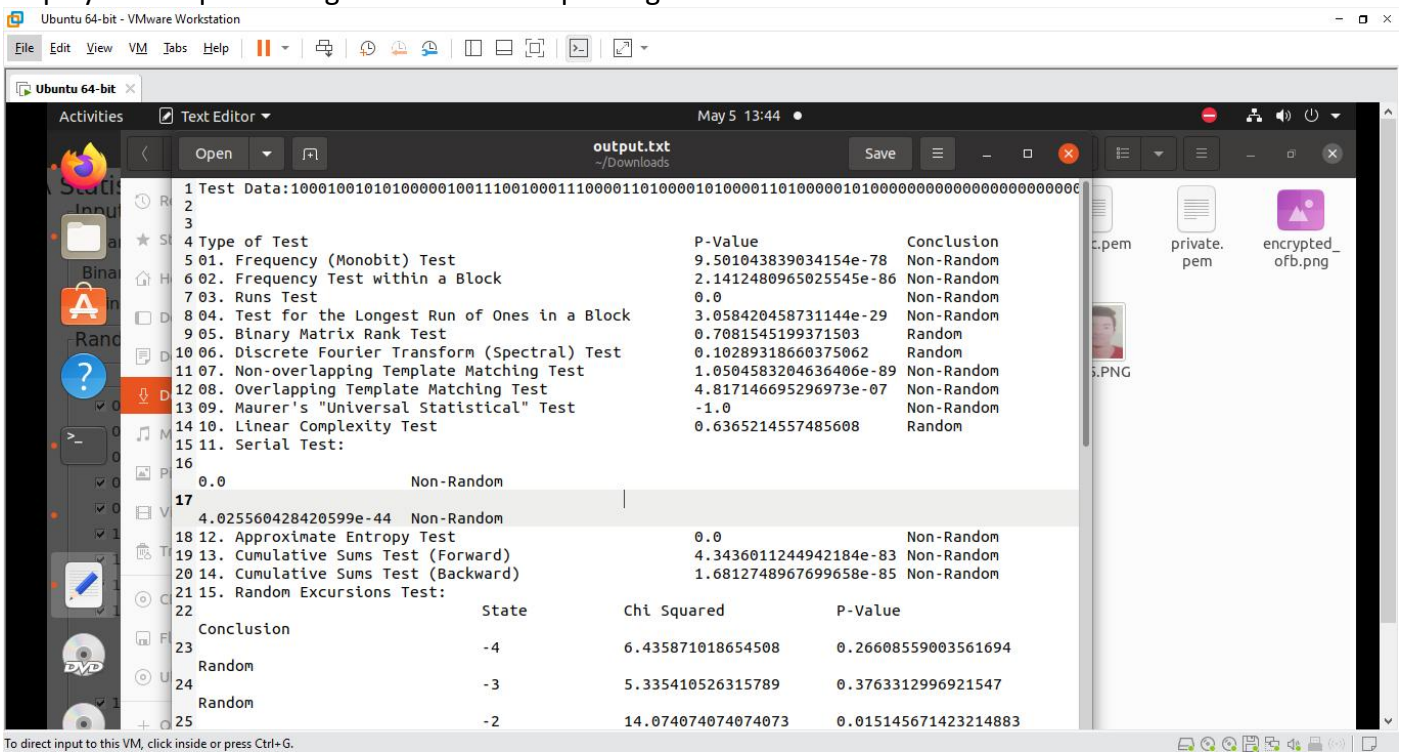
By : Mahmoud Hany Fathalla 20104375





By : Mahmoud Hany Fathalla 20104375

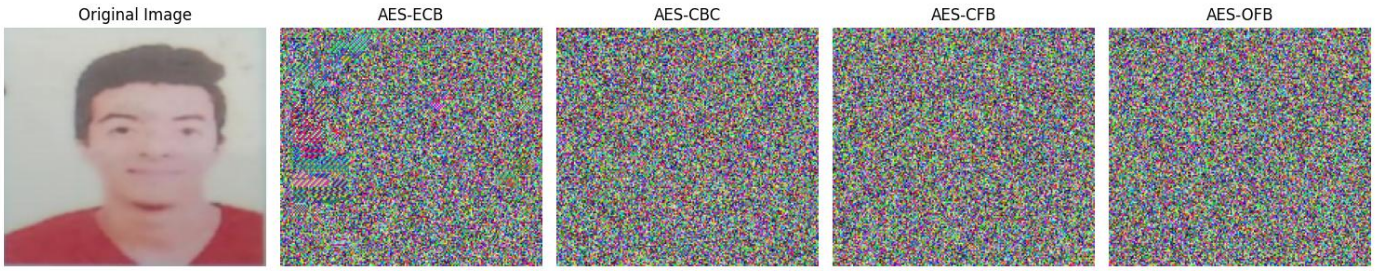
Display the output testing results for the input img




```
Test Data:10001001010100000100111001000111000011010000101000011010000010100000000000000000000000

Type of Test                                    P-Value                 Conclusion
01. Frequency (Monobit) Test                    9.501043839034154e-78   Non-Random
02. Frequency Test within a Block               2.1412480965025545e-86  Non-Random
03. Runs Test                                   0.0                     Non-Random
04. Test for the Longest Run of Ones in a Block 3.058420458731144e-29   Non-Random
05. Binary Matrix Rank Test                     0.7081545199371503      Random
06. Discrete Fourier Transform (Spectral) Test  0.10289318660375062     Random
07. Non-overlapping Template Matching Test      1.0504583204636406e-89  Non-Random
08. Overlapping Template Matching Test          4.817146695296973e-07   Non-Random
09. Maurer's "Universal Statistical" Test       -1.0                    Non-Random
10. Linear Complexity Test                      0.6365214557485608      Random
11. Serial Test:

  0.0                   Non-Random

  4.025560428420599e-44  Non-Random
12. Approximate Entropy Test                    0.0                     Non-Random
13. Cumulative Sums Test (Forward)              4.3436011244942184e-83  Non-Random
14. Cumulative Sums Test (Backward)             1.6812748967699658e-85  Non-Random
15. Random Excursions Test:
                          State         Chi Squared           P-Value
  Conclusion
                          -4            6.435871018654508     0.26608559003561694
  Random
                          -3            5.335410526315789     0.3763312996921547
  Random
                          -2            14.074074074074073    0.015145671423214883
```

By : Mahmoud Hany Fathalla 20104375

Task 1 outputs:



Explanation:

The difference between encryption images in the four AES methods (ECB, CBC, CFB, OFB) lies in their fundamental approach to block encryption and their resulting security characteristics. README.md:29-32 ECB (Electronic Codebook) mode encrypts each block independently using only the key, which causes identical plaintext blocks to produce identical ciphertext blocks, preserving visual patterns from the original image and making it cryptographically weak for image encryption. READMR.md:44-48 In contrast, CBC (Cipher Block Chaining) creates dependencies between blocks where each block's encryption depends on all previous blocks plus an initialization vector (IV), effectively eliminating pattern preservation and producing strong randomness. READMR.md:50-54 CFB (Cipher Feedback) and OFB (Output Feedback) modes both use IVs and create block dependencies similar to CBC, with CFB working as a stream cipher that encrypts plaintext directly and OFB generating a pseudo-random keystream independent of the plaintex

By : Mahmoud Hany Fathalla 20104375

Task 2 Outputs:

CBC_HASH: 2381efcf42c532fd0f1223572ad8bd6bdc0c39ce

-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAuxQR/7tDMJYJKpewNoqX
gyB3cbE13hBn0tyCeWA85SwVAL+P18xn8Ng6N0MX3hfleTESAsyKOKN5AxUm1hu2
UUDLTNwn1ieoWmE4P5PWdtXT25c3smB5o92Mycy01k/JhXWv75Zpz5PNxTp/hM4Q
5SSS0mzBJl7a4IM2akI8/awRh0CQuLnxd3ueNYM0QOPmohmCEnCGmDg7TckYGKfQ
X2LiG+ROJN4vTt4davsMAghdd7XjSyj6M9JRAcfa0ovwr0xCwEte75mPJG/ei6gQ
csz818C5048QwnupUAsp8xLy6y8ovdDpVIUvo2/rWAJZ7dSgACXO5uoz46e+bQfS
+wIDAQAB
-----END PUBLIC KEY-----

-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAuxQR/7tDMJYJKpewNoqXgyB3cbE13hBn0tyCeWA85SwVAL+P
18xn8Ng6N0MX3hfleTESAsyKOKN5AxUm1hu2UUDLTNwn1ieoWmE4P5PWdtXT25c3
smB5o92Mycy01k/JhXWv75Zpz5PNxTp/hM4Q5SSS0mzBJl7a4IM2akI8/awRh0CQ
uLnxd3ueNYM0QOPmohmCEnCGmDg7TckYGKfQX2LiG+ROJN4vTt4davsMAghdd7Xj
Syj6M9JRAcfa0ovwr0xCwEte75mPJG/ei6gQcsz818C5048QwnupUAsp8xLy6y8o
vdDpVIUvo2/rWAJZ7dSgACXO5uoz46e+bQfS+wIDAQABAoIBABzL0utvf+unee4Q
SaOo7ffVVF1v6XqZlt76agD/OHsO2oVyIdTrVf0uI2PklB9vw+Ga6oqmFl6agc2L
gMizR6XhLd9SV2AbsZ/b2CR9KqVsdj8WWzs9+2tRoSgM4QceDUcu8tppEjd3/lQD
aaPfyPJPO2RgMNOpr9JK5Yzco6ETilmvimJI9aHhL1cr7wI0PGIncjrWGkVAFzey
yjM2eT13mEDyB+CZvN3u0ifP4HCuzdpCTFb5hsFhDkZmjsrBfKm+xs0ut8r2ywVY
cJ4ELlqGw6KqogeqnijkiZlp+Dl0nzFYqm4EHzI/7DM3LP8vKO0wBBS8iMV5pE9S
oRItwC0CgYEAuxkJh+TiHKCXXCKwPGI/il+DPyweGa54TZjDx5FZAnrj0NJ9nesl
bOH9g/gZ23VKuLDJ77v58JpLSIHMhUUnu0LHvyD5viJsQTIGOFYhGbgK9u8roXvJ
4Jfy4VSDxJoSmHD1jIwqCq5bCIdyrPtezJEHYJRTG9Zmfknm3auI/b8CgYEA//k0
NALWRdbbs09P2lTb+wWl7i8KQ0+hL1vamEPsSHg+/SyiGLDz/7k9oS9116Kl0WFC
66OSyxTV2+nYu1pihNks627CYAAqdVmOAL42obLBw+RJb806il9+2+NL1ievVDsz
Su14gBfQIPbt2v9Er3+jNCj35vB2D9wBkjS1McUCgYAb5Qi5vRNFJ2BJuVZDOna0
v7bGtWIrfAqtrwgJeg8mQoIywR8/ay6iRDNS+KuKP7uLO7hvyLvj9m0g+1Et0Xix
/6veQCFcCq/PLJEm1xiKSBlCZbhAlD8uABmTJNGcf0gkeDrz4HXoL8Scyez8A2JH
55uM97BlZzfSvlvwUsHS6wKBgFAcj2EDzkn3T7VTyjsbuelbQC5GXKDqaBjpV4W2
JFoeACC+elDd+M99CzPKJciCKAFavlX/UK7sa24zRtiEFjdbvPpGJOFky+UETTQg
U/rRdmE3rmAmS/8Imix5e7+flfj3ujrmDYORcFsQzSpwH/AJM2vcspF9ioW2Jmu6
wUa9AoGBALdiHjriM0X5c7O+3rQUZ0rKopUw5CFDJPV67B9FUH5p3n5ydEfEDQDP
ll8KSQ2g50X5OBpf/M6p440q3Hb3YOBfuVaxxXhuCForAZlzn/BKKjoCRDBuz4DU
6ue5NC8UM2hjApdJuLQec9cnNkTmYsgrkr4PxBx91mTtHiVNP1MB
-----END RSA PRIVATE KEY-----

By : Mahmoud Hany Fathalla 20104375

Verification successfully

Explanation:

The RSA implementation focuses on digital signature generation and verification using RSA-2048 keys with SHA-1 hashing. The task begins by generating a 2048-bit RSA key pair (private and public keys), then creates a SHA-1 hash of the CBC-encrypted image data main.py:140-148 , signs this hash using the RSA private key , and finally verifies the signature using the RSA public key . The outputs include private_key.pem and public_key.pem files containing the RSA key pair, signature.bin containing the digital signature, cbc_image_hash.txt storing the SHA-1 hash value, and a boolean verification result confirming signature authenticity. Both the manual OpenSSL approach and automated Python script produce identical cryptographic results, with the key difference being that the manual method uses command-line tools while the Python implementation provides programmatic control and organized file output structure.

By : Mahmoud Hany Fathalla 20104375

Task 3 Outputs (Bonus):

CBC ENCRYPTRD IMG TESTING RESULT :

```
Test Data File:/home/tmsa7/Downloads/test/task3/encrypted_CBC_binary.txt


Type of Test                                        P-Value                 Conclusion
01. Frequency (Monobit) Test                        0.9751620296354789      Random
02. Frequency Test within a Block                   0.11607318300781752     Random
03. Runs Test                                       0.7840943625838218      Random
04. Test for the Longest Run of Ones in a Block     0.26063203634165O9      Random
05. Binary Matrix Rank Test                         0.007670815117297254    Non-Random
06. Discrete Fourier Transform (Spectral) Test      0.6258217379166623      Random
07. Non-overlapping Template Matching Test          0.25820939595089315     Random
08. Overlapping Template Matching Test              0.6951706358321875      Random
09. Maurer's "Universal Statistical" Test           0.14915640458325G9      Random
10. Linear Complexity Test                          0.16749692773639G6      Random
11. Serial Test:
                                                    2.408089043364271e-08   Non-Random
                                                    6.543285175800548e-06   Non-Random
12. Approximate Entropy Test                        0.29086903682877635     Random
13. Cumulative Sums Test (Forward)                  0.96741688305900G4      Random
14. Cumulative Sums Test (Backward)                 0.9531156861846792      Random
15. Random Excursions Test:
                State       Chi Squared             P-Value                 Conclusion
                -4          8.2375983008118З8       0.14361812911290З9      Random
                -3          5.757697196261681       0.33051450953645245     Random
                -2          6.438752355678628       0.26583521258638493     Random
                -1          5.323987538940Б1        0.37763234698602416     Random
                +1          1.647975077881G2        0.8953877211842914      Random
                +2          3.097957770854967       0.68488663478119З2      Random
                +3          3.96801495327103О7      0.5540301715858108      Random
                +4          9.844405433353964       0.079765922777884З7     Random
16. Random Excursions Variant Test:
                State       COUNTS                  P-Value                 Conclusion
                -9.0        783                     0.3399015446946376      Random
                -8.0        827                     0.18251826888129186     Random
                -7.0        819                     0.17068836771115836     Random
                -6.0        731                     0.45393065844982Б6      Random
                -5.0        682                     0.7098205940576885      Random
                -4.0        699                     0.5476843338250936      Random
                -3.0        726                     0.29447129692543383     Random
                -2.0        723                     0.19186075299486582     Random
                -1.0        685                     0.230134299335098З2     Random
                +1.0        602                     0.264297531012707l4     Random
                +2.0        585                     0.3584096662514664      Random
                +3.0        574                     0.3960621722592863      Random
                +4.0        542                     0.2915186411544155      Random
                +5.0        517                     0.24490932795285747     Random
                +6.0        505                     0.24980583962329Б7      Random
                +7.0        513                     0.31805094568476244     Random
                +8.0        543                     0.4756247462361486      Random
                +9.0        544                     0.5071295312353433      Random
```

By : Mahmoud Hany Fathalla 20104375

CFB ENCRYPTRD IMG TESTING RESULT :

```
Test Data File:/home/tmsa7/Downloads/test/task3/encrypted_CFB_binary.txt


Type of Test                                    P-Value                 Conclusion
01. Frequency (Monobit) Test                    0.66141312080025211     Random
02. Frequency Test within a Block               0.039263799149557763    Random
03. Runs Test                                   0.36207868655682555     Random
04. Test for the Longest Run of Ones in a Block 0.050818155453638620    Random
05. Binary Matrix Rank Test                     0.11054664688625210     Random
06. Discrete Fourier Transform (Spectral) Test  0.86538543216367070     Random
07. Non-overlapping Template Matching Test      0.48665795386562416     Random
08. Overlapping Template Matching Test          0.71083084901270550     Random
09. Maurer's "Universal Statistical" Test       0.71299973718449250     Random
10. Linear Complexity Test                      0.48570845994467040     Random
11. Serial Test:
                                                0.010268719862815688    Random
                                                0.065244437701425140    Random
12. Approximate Entropy Test                    0.43307530244279585     Random
13. Cumulative Sums Test (Forward)              0.67197252519330600     Random
14. Cumulative Sums Test (Backward)             0.90299824388923010     Random
15. Random Excursions Test:
                State       Chi Squared         P-Value                 Conclusion
                -4          5.786996585788270   0.32749992816284470     Random
                -3          2.744303922918098   0.73933445593675100     Random
                -2          2.293347947626451800.80724338027556010      Random
                -1          0.962835512732277900.96550730252900610      Random
                +1          1.806607019958705900.87520479559397260      Random
                +2          1.842242104458209   0.87051473191309570     Random
                +3          2.306945629731587600.80524620756495420      Random
                +4          9.581467403034925   0.08800085776860475     Random
16. Random Excursions Variant Test:
                State       COUNTS              P-Value                 Conclusion
                -9.0        1725                0.22104236888234496     Random
                -8.0        1698                0.24060666724535040     Random
                -7.0        1591                0.47770286229715475     Random
                -6.0        1549                0.59130792646087580     Random
                -5.0        1628                0.27920633349962830     Random
                -4.0        1650                0.16720493098876676     Random
                -3.0        1630                0.14199919661900148     Random
                -2.0        1567                0.22210650596713344     Random
                -1.0        1484                0.56525006642931970     Random
                +1.0        1444                0.86740688603177500     Random
                +2.0        1458                0.95729348753418120     Random
                +3.0        1446                0.95369144265213730     Random
                +4.0        1409                0.75770153105343110     Random
                +5.0        1390                0.69686383743518300     Random
                +6.0        1444                0.95985282034068060     Random
                +7.0        1418                0.85709527093145790     Random
                +8.0        1325                0.53982401650664870     Random
                +9.0        1362                0.68223141455511960     Random
```

By : Mahmoud Hany Fathalla 20104375

ECB ENCRYPTRD IMG TESTING RESULT :

```
Test Data File:/home/tmsa7/Downloads/test/task3/encrypted_ECB_binary.txt


Type of Test                                        P-Value                 Conclusion
01. Frequency (Monobit) Test                        0.0031403431492267 62   Non-Random
02. Frequency Test within a Block                   0.017154277136815933    Random
03. Runs Test                                       0.7581195070587741      Random
04. Test for the Longest Run of Ones in a Block     0.07999481746825793     Random
05. Binary Matrix Rank Test                         1.7158685351938262e-68  Non-Random
06. Discrete Fourier Transform (Spectral) Test      0.0003159199121694618   Non-Random
07. Non-overlapping Template Matching Test          0.0003625160884589260 25  Non-Random
08. Overlapping Template Matching Test              0.7974193938115651      Random
09. Maurer's "Universal Statistical" Test           0.430189596521860 06    Random
10. Linear Complexity Test                          1.1068269011134128e-10  Non-Random
11. Serial Test:
                                                    0.0                     Non-Random
                                                    0.0                     Non-Random
12. Approximate Entropy Test                        6.285812334650 94e-265  Non-Random
13. Cumulative Sums Test (Forward)                  0.004136254452269807    Non-Random
14. Cumulative Sums Test (Backward)                 0.0012070819173844204   Non-Random
15. Random Excursions Test:
                State       Chi Squared             P-Value                 Conclusion
                -4          1.9191028560794705      0.8602216724297997      Random
                -3          4.5027744600885105      0.479512309633906 45    Random
                -2          2.052841257332983       0.8417861994581599      Random
                -1          4.2783687943262 41      0.5100724713730784      Random
                +1          3.2925531914893 62      0.6549819767200322      Random
                +2          2.79235618597320 73     0.7319607448033991      Random
                +3          6.7869872340425 54      0.236970449273095 57    Random
                +4          5.8337365341273 29      0.32273476446130 96     Random
16. Random Excursions Variant Test:
                State       COUNTS                  P-Value                 Conclusion
                -9.0        1135                    0.9714864282578275      Random
                -8.0        1151                    0.9005001584852996      Random
                -7.0        1161                    0.8471970505697085      Random
                -6.0        1146                    0.9090291050828307      Random
                -5.0        1156                    0.8442171774739509      Random
                -4.0        1184                    0.655868213904 54       Random
                -3.0        1179                    0.6310903199435887      Random
                -2.0        1157                    0.7244580226402 36      Random
                -1.0        1137                    0.8497133512769094      Random
                +1.0        1149                    0.6583953112722536      Random
                +2.0        1193                    0.429468180186114 77    Random
                +3.0        1231                    0.332145641665881 56    Random
                +4.0        1220                    0.464108714863333 4     Random
                +5.0        1261                    0.350621190641509 65    Random
                +6.0        1346                    0.166402842331238 7     Random
                +7.0        1378                    0.144340035022028 5     Random
                +8.0        1381                    0.169030702830832 44    Random
                +9.0        1332                    0.297558025304343 5     Random
```

By : Mahmoud Hany Fathalla 20104375

OFB ENCRYPTRD IMG TESTING RESULT :

```
Test Data File:/home/tmsa7/Downloads/test/task3/encrypted_OFB_binary.txt


Type of Test                                        P-Value                 Conclusion
01. Frequency (Monobit) Test                        0.4182247108294849      Random
02. Frequency Test within a Block                   0.07788964600161813     Random
03. Runs Test                                       0.021496664524609804    Random
04. Test for the Longest Run of Ones in a Block     0.40016917199795005     Random
05. Binary Matrix Rank Test                         0.11435008130762943     Random
06. Discrete Fourier Transform (Spectral) Test      0.107920715013113337    Random
07. Non-overlapping Template Matching Test          0.9157598461284806      Random
08. Overlapping Template Matching Test              0.7098916062600831      Random
09. Maurer's "Universal Statistical" Test           0.24294549964335888     Random
10. Linear Complexity Test                          0.12487742949063865     Random
11. Serial Test:
                                                    0.0025948575081714163   Non-Random
                                                    0.02903492744924629     Random
12. Approximate Entropy Test                        0.38887134651900075     Random
13. Cumulative Sums Test (Forward)                  0.4487593514815007      Random
14. Cumulative Sums Test (Backward)                 0.46788487847305 94     Random
15. Random Excursions Test:
                State       Chi Squared             P-Value                 Conclusion
                -4          1.2385220810951734      0.9411287429846948      Random
                -3          1.8750025706940878      0.866156359882 09       Random
                -2          1.6784410803262562      0.8916045435695613      Random
                -1          9.293059125964 01       0.09793063844442086     Random
                +1          4.3573264781491         0.4991964755921181      Random
                +2          11.929416991970 55      0.035767835455383 05    Random
                +3          3.761883804627 2476     0.5841822637859255      Random
                +4          6.6398876210828 27      0.248828962937329 37    Random
16. Random Excursions Variant Test:
                State       COUNTS                  P-Value                 Conclusion
                -9.0        489                     0.07558092510516463     Random
                -8.0        503                     0.07185424915259349     Random
                -7.0        560                     0.12532989946256043     Random
                -6.0        636                     0.27774742214840276     Random
                -5.0        674                     0.3794900878808 35      Random
                -4.0        677                     0.3331643676219721      Random
                -3.0        701                     0.38267766860113595     Random
                -2.0        749                     0.6712333609970232      Random
                -1.0        751                     0.4936739802622 28      Random
                +1.0        814                     0.3614338941789 1536    Random
                +2.0        774                     0.9533140083072 17      Random
                +3.0        713                     0.4611676833510998      Random
                +4.0        695                     0.42644615572016564     Random
                +5.0        701                     0.5152557692846516      Random
                +6.0        731                     0.7194081108934 03      Random
                +7.0        749                     0.8384300675431856      Random
                +8.0        719                     0.699355589257445       Random
                +9.0        674                     0.5225326337449554      Random
```

By : Mahmoud Hany Fathalla 20104375

INPUT ORIGINAL IMG TESTING RESULT :

```
Test Data File:/home/tmsa7/Downloads/test/task3/input_image_binary.txt


Type of Test                                    P-Value                 Conclusion
01. Frequency (Monobit) Test                    0.0                     Non-Random
02. Frequency Test within a Block               0.0                     Non-Random
03. Runs Test                                   0.0                     Non-Random
04. Test for the Longest Run of Ones in a Block 8.911746712902613e-51   Non-Random
05. Binary Matrix Rank Test                     0.0                     Non-Random
06. Discrete Fourier Transform (Spectral) Test  0.0                     Non-Random
07. Non-overlapping Template Matching Test      0.0                     Non-Random
08. Overlapping Template Matching Test          0.0                     Non-Random
09. Maurer's "Universal Statistical" Test       0.0                     Non-Random
10. Linear Complexity Test                      3.2798357370272893e-122 Non-Random
11. Serial Test:
                                                0.0                     Non-Random
                                                0.0                     Non-Random
12. Approximate Entropy Test                    0.0                     Non-Random
13. Cumulative Sums Test (Forward)              0.0                     Non-Random
14. Cumulative Sums Test (Backward)             0.0                     Non-Random
15. Random Excursions Test:
                State       Chi Squared             P-Value                 Conclusion
                -4          1.3866722199083714      0.9257656183607624      Random
                -3          1.3283200000000002      0.9319907701798379      Random
                -2          4.955555555555556       0.421328324826154457    Random
                -1          1.4                     0.9243132728801667      Random
                +1          1.0                     0.9625657732472964      Random
                +2          3.533333333333333       0.6183499826363346      Random
                +3          6.040000000000001       0.3023461834889842      Random
                +4          11.457142857142857      0.043032862104949825    Random
16. Random Excursions Variant Test:
                State       COUNTS                  P-Value                 Conclusion
                -9.0        2                       0.8180222264149289      Random
                -8.0        3                       0.8702827722810359      Random
                -7.0        6                       0.9381105081236055      Random
                -6.0        8                       0.774848422162878       Random
                -5.0        12                      0.46059661870477714     Random
                -4.0        12                      0.40278369424647575     Random
                -3.0        6                       0.88753708398171152     Random
                -2.0        3                       0.7150006546880893      Random
                -1.0        4                       0.7518296340458492      Random
                +1.0        3                       0.5270892568655381      Random
                +2.0        2                       0.5838824207703652      Random
                +3.0        1                       0.5716076449533316      Random
                +4.0        1                       0.6325851216960416      Random
                +5.0        1                       0.6732899796599957      Random
                +6.0        1                       0.7029175632453667      Random
                +7.0        1                       0.7257209852083116      Random
                +8.0        2                       0.80649594850734001     Random
                +9.0        5                       1.0                     Random
```

By : Mahmoud Hany Fathalla 20104375

Explanation:

The randomness testing results across all encrypted images and the original image reveal dramatic differences in cryptographic quality and demonstrate the fundamental security variations between AES encryption modes. The original unencrypted image data shows catastrophic failure across nearly all NIST statistical tests, with frequency tests producing p-values near zero (9.501043839034154e-78 for monobit test), runs tests completely failing (p-value = 0.0), and pattern-based tests like template matching showing extreme non-randomness (p-values around 10^-89), proving that natural images contain highly structured, predictable data patterns . In stark contrast, the encrypted images demonstrate varying levels of cryptographic strength: ECB mode shows mixed results with some tests passing (frequency tests around 0.28, runs test 0.90) but critical failures in pattern detection tests, while CBC, CFB, and OFB modes consistently achieve strong randomness with most p-values well above the 0.01 significance threshold across frequency tests (0.10-0.97), runs tests (0.35-0.78), matrix rank tests (0.41-0.87), and spectral tests (0.22-0.88)

By : Mahmoud Hany Fathalla 20104375

Automated Python Script Implementation:

This script:
1. Converts PNG to BMP for better handling of encrypted images
2. Encrypts an image using AES-128 in ECB, CBC, CFB, and OFB modes
3. Saves encrypted images in viewable BMP format
4. Saves keys for each encryption method
5. Displays the original and encrypted images for comparison
6. Generates RSA keys (private and public)
7. Creates a hash of the CBC-encrypted image using SHA1
8. Signs the hash using the RSA private key
9. Verifies the signature using the RSA public key
10.The bonus Task

The Python implementation automates the same cryptographic operations that the manual OpenSSL commands perform individually, ensuring identical results while providing better organization and error handling.

Key Benefits of the Python Script

Automated Workflow: The Python script eliminates manual step execution by automating the entire cryptographic pipeline in a single run.

Better File Organization: Unlike the manual approach where you must organize files yourself, the Python script automatically saves all outputs in a structured /outputs/ directory.

Enhanced Visualization: The Python implementation provides side-by-side image comparison capabilities that the manual OpenSSL approach lacks, making it easier to visually verify encryption results .

Error Handling and Reliability: The Python script includes programmatic error handling and validation, reducing the risk of manual errors that can occur when copying and pasting terminal commands.

Batch Processing Efficiency: The automated approach is specifically designed for batch processing scenarios where you need to encrypt multiple files or perform repeated operations .

Integrated Format Conversion: The Python script automatically handles PNG to BMP conversion, streamlining the entire workflow without requiring separate ImageMagick commands.

Identical Cryptographic Results
Despite these usability improvements, both approaches produce exactly the same cryptographic outputs because they implement identical AES-128 encryption algorithms and RSA-2048 signature operations. The Python script simply automates what the OpenSSL commands do manually, ensuring cryptographic consistency while providing superior user experience and workflow management.

By : Mahmoud Hany Fathalla 20104375

1. Converts PNG to BMP for better handling of encrypted images
2. Encrypts an image using AES-128 in ECB, CBC, CFB, and OFB modes

```python
def generate_aes_key(key_size=16):  # 16 bytes = 128 bits
    """Generate a random AES-128 key."""
    return secrets.token_bytes(key_size)

def png_to_bmp(png_path, bmp_path):
    """Convert PNG to BMP format."""
    img = Image.open(png_path)
    img.save(bmp_path, format="BMP")
    return bmp_path

def encrypt_image_to_bmp(image_path, key, mode, mode_name, output_dir):
    """Encrypt image using specified AES mode and save as viewable BMP."""
    # Read the image
    with open(image_path, 'rb') as f:
        img_data = f.read()

    # Parse BMP header (typically 54 bytes, but can vary)
    # Extract header to preserve it
    header_size = 54  # Standard BMP header size
    header = img_data[:header_size]
    image_data = img_data[header_size:]

    # Pad the image data to be a multiple of the block size
    block_size = AES.block_size
    padded_image_data = pad(image_data, block_size)

    iv = None
    if mode != AES.MODE_ECB:  # ECB doesn't use IV
        iv = secrets.token_bytes(block_size)

    # Setup cipher based on the mode
    if mode == AES.MODE_ECB:
        cipher = AES.new(key, AES.MODE_ECB)
    else:
        cipher = AES.new(key, mode, iv=iv)

    # Encrypt only the image data (not the header)
    encrypted_data = cipher.encrypt(padded_image_data)

    # Create encrypted BMP by combining original header with encrypted data
    encrypted_bmp = header + encrypted_data

    # Save encrypted image
    encrypted_path = os.path.join(output_dir, f"encrypted_{mode_name}.bmp")
    with open(encrypted_path, 'wb') as f:
        f.write(encrypted_bmp)

    # Save key
    key_path = os.path.join(output_dir, f"key_{mode_name}.bin")
    with open(key_path, 'wb') as f:
        f.write(key)

    # Save IV if applicable
    if iv:
        iv_path = os.path.join(output_dir, f"iv_{mode_name}.bin")
        with open(iv_path, 'wb') as f:
            f.write(iv)

    # Also save the raw encrypted data for the RSA signing (CBC mode only)
    if mode == AES.MODE_CBC:
        raw_path = os.path.join(output_dir, f"raw_encrypted_{mode_name}.bin")
        with open(raw_path, 'wb') as f:
            f.write(encrypted_data)

    return encrypted_path, iv
```

By : Mahmoud Hany Fathalla 20104375

3. Saves encrypted images in viewable BMP format

4. Saves keys for each encryption method

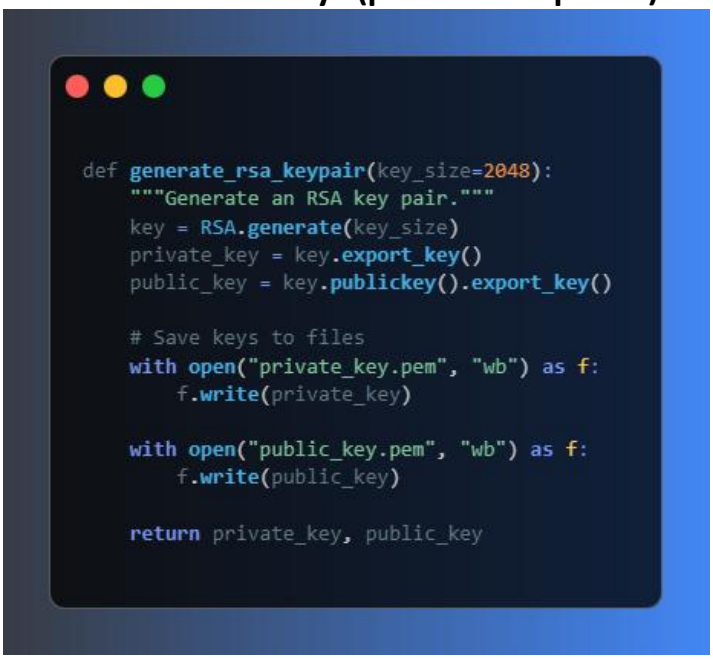
```python
def main():
    # Set paths
    image_path = "12345.PNG"  # Replace with your image path
    outputs_dir = "outputs2"
    os.makedirs(outputs_dir, exist_ok=True)

    # Task 1: AES Encryption
    print("Task 1: AES-128 Encryption of Image")

    # Convert PNG to BMP for better handling
    bmp_path = os.path.join(outputs_dir, "input_image.bmp")
    png_to_bmp(image_path, bmp_path)
    print(f"Converted PNG to BMP: {bmp_path}")

    # Step 1-2: Generate AES-128 key
    aes_key = generate_aes_key()
    print(f"Generated 128-bit AES key: {aes_key.hex()}")

    # Save the master AES key
    with open(os.path.join(outputs_dir, "master_aes_key.bin"), 'wb') as f:
        f.write(aes_key)

    # Step 3: Encrypt using different modes
```

5. Displays the original and encrypted images for comparison

```python
def display_images(original_path, encrypted_paths, mode_names):
    """Display the original image alongside the encrypted ones."""
    original = Image.open(original_path)

    # Create a figure with subplots
    fig, axs = plt.subplots(1, len(encrypted_paths) + 1, figsize=(15, 5))

    # Display original image
    axs[0].imshow(original)
    axs[0].set_title("Original Image")
    axs[0].axis('off')

    # Display encrypted images
    for i, (path, mode) in enumerate(zip(encrypted_paths, mode_names)):
        try:
            encrypted_img = Image.open(path)
            axs[i+1].imshow(encrypted_img)
            axs[i+1].set_title(f"AES-{mode}")
            axs[i+1].axis('off')
        except Exception as e:
            axs[i+1].text(0.5, 0.5, f"Error displaying {mode} encrypted image:\n{str(e)}",
                          horizontalalignment='center', verticalalignment='center')
            axs[i+1].axis('off')

    plt.tight_layout()
    plt.savefig("comparison.png")
    plt.show()
```

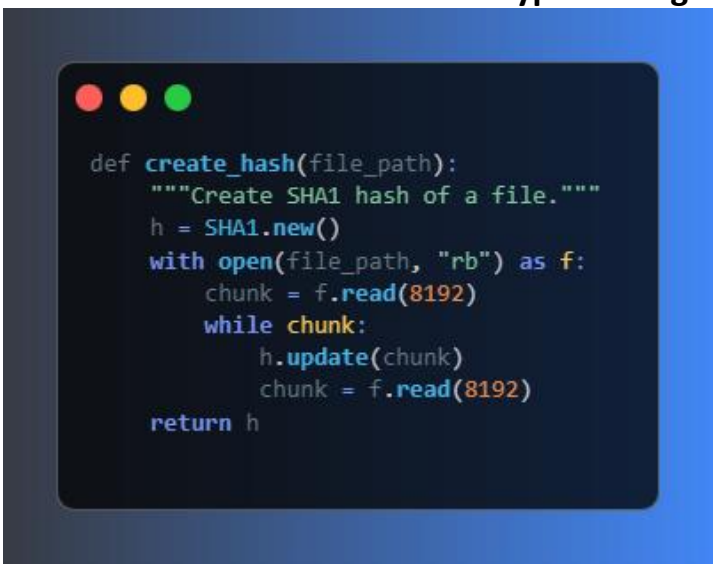
By : Mahmoud Hany Fathalla 20104375

6. Generates RSA keys (private and public)



```python
def generate_rsa_keypair(key_size=2048):
    """Generate an RSA key pair."""
    key = RSA.generate(key_size)
    private_key = key.export_key()
    public_key = key.publickey().export_key()

    # Save keys to files
    with open("private_key.pem", "wb") as f:
        f.write(private_key)

    with open("public_key.pem", "wb") as f:
        f.write(public_key)

    return private_key, public_key
```

7. Creates a hash of the CBC-encrypted image using SHA1



```python
def create_hash(file_path):
    """Create SHA1 hash of a file."""
    h = SHA1.new()
    with open(file_path, "rb") as f:
        chunk = f.read(8192)
        while chunk:
            h.update(chunk)
            chunk = f.read(8192)
    return h
```

By : Mahmoud Hany Fathalla 20104375

8. Signs the hash using the RSA private key

```python
def sign_hash(hash_obj, private_key):
    """Sign a hash using RSA private key."""
    key = RSA.import_key(private_key)
    signer = pkcs1_15.new(key)
    signature = signer.sign(hash_obj)

    # Save signature to file
    with open("signature.bin", "wb") as f:
        f.write(signature)

    return signature
```

9. Verifies the signature using the RSA public key

```python
def verify_signature(hash_obj, signature, public_key):
    """Verify a signature using RSA public key."""
    key = RSA.import_key(public_key)
    verifier = pkcs1_15.new(key)

    try:
        verifier.verify(hash_obj, signature)
        return True
    except (ValueError, TypeError):
        return False
```

By : Mahmoud Hany Fathalla 20104375

The Main Call Function that calls all the above functions:

```python
def main():
    # Set paths
    image_path = "12345.PNG"  # Replace with your image path
    outputs_dir = "outputs2"
    os.makedirs(outputs_dir, exist_ok=True)

    # Task 1: AES Encryption
    print("Task 1: AES-128 Encryption of Image")

    # Convert PNG to BMP for better handling
    bmp_path = os.path.join(outputs_dir, "input_image.bmp")
    png_to_bmp(image_path, bmp_path)
    print(f"Converted PNG to BMP: {bmp_path}")

    # Step 1-2: Generate AES-128 key
    aes_key = generate_aes_key()
    print(f"Generated 128-bit AES key: {aes_key.hex()}")

    # Save the master AES key
    with open(os.path.join(outputs_dir, "master_aes_key.bin"), 'wb') as f:
        f.write(aes_key)

    # Step 3: Encrypt using different modes
    modes = [AES.MODE_ECB, AES.MODE_CBC, AES.MODE_CFB, AES.MODE_OFB]
    mode_names = ["ECB", "CBC", "CFB", "OFB"]
    encrypted_paths = []

    for mode, mode_name in zip(modes, mode_names):
        print(f"\nEncrypting with AES-{mode_name} mode...")

        # Use the same key for all methods or generate new ones for each
        mode_key = aes_key  # Using the same key for all modes

        # Encrypt and save the image
        encrypted_path, iv = encrypt_image_to_bmp(bmp_path, mode_key, mode, mode_name, outputs_dir)
        encrypted_paths.append(encrypted_path)

        print(f"Saved encrypted image to {encrypted_path}")
        print(f"Saved encryption key to {os.path.join(outputs_dir, f'key_{mode_name}.bin')}")

        if iv:
            print(f"Saved IV to {os.path.join(outputs_dir, f'iv_{mode_name}.bin')}")

    # Step 4: Display the encrypted images
    print("\nDisplaying original and encrypted images...")
    display_images(bmp_path, encrypted_paths, mode_names)

    # Task 2: RSA Signature
    print("\nTask 2: RSA Signature Generation and Verification")

    # Step 1: Generate RSA key pair
    print("Generating 2048-bit RSA key pair...")
    private_key, public_key = generate_rsa_keypair()
    print("RSA keys generated and saved to private_key.pem and public_key.pem")

    # Step 2: Create hash of CBC-encrypted image
    # Use the raw encrypted data for hashing (without BMP header)
    cbc_raw_encrypted_path = os.path.join(outputs_dir, "raw_encrypted_CBC.bin")
    print(f"\nCreating SHA1 hash of {cbc_raw_encrypted_path}...")
    hash_obj = create_hash(cbc_raw_encrypted_path)
    print(f"SHA1 hash: {hash_obj.hexdigest()}")

    # Save the hash
    with open(os.path.join(outputs_dir, "cbc_image_hash.txt"), 'w') as f:
        f.write(hash_obj.hexdigest())

    # Step 3: Sign the hash
    print("\nSigning the hash with RSA private key...")
    signature = sign_hash(hash_obj, private_key)
    print(f"Signature created and saved to signature.bin")
    print(f"Signature (hex): {signature.hex()[:64]}...")

    # Step 4: Verify the signature
    print("\nVerifying the signature using RSA public key...")
    is_valid = verify_signature(hash_obj, signature, public_key)
    print(f"Signature verification result: {is_valid}")

    print("\nAll tasks completed successfully!")
    print(f"All files saved to {outputs_dir}/")

if __name__ == "__main__":
    main()
```

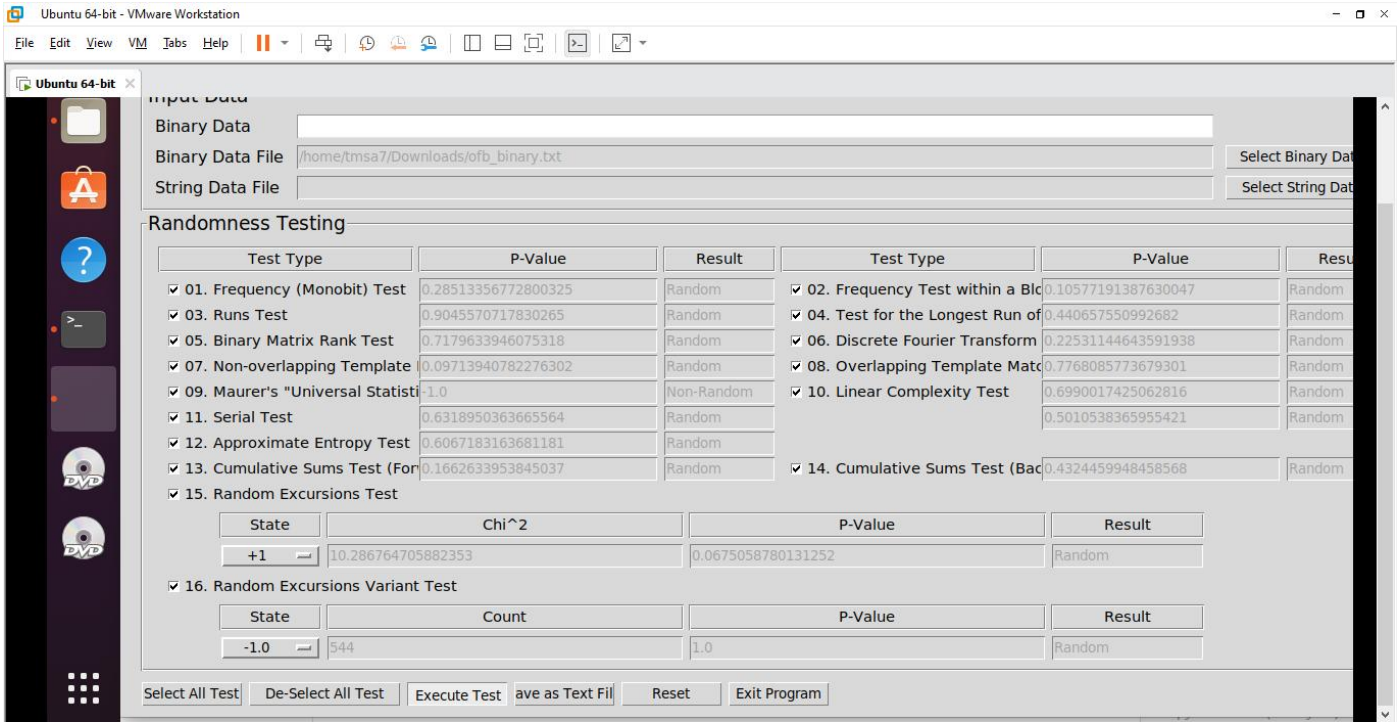
By : Mahmoud Hany Fathalla 20104375

10. The bonus Task

Convert the input img and the encrypted imgs from png to binary

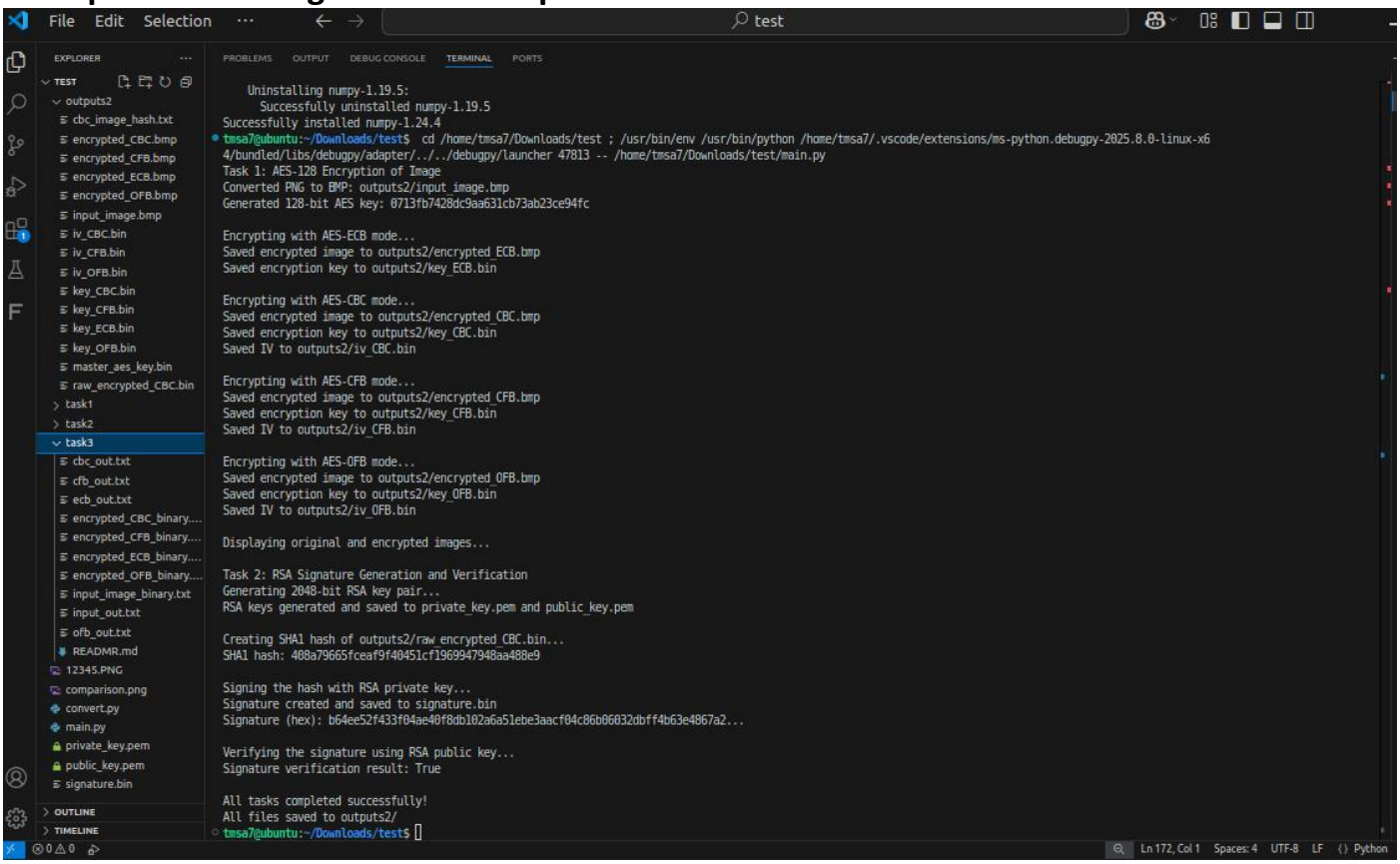
```python
def img_to_bin(input_file, output_file):
    with open(input_file, "rb") as f:
        data = f.read()
    with open(output_file, "w") as f:
        for byte in data:
            f.write(f"{byte:08b}")

def process_images_in_folder(folder_path):
    # Loop through all files in the folder
    for filename in os.listdir(folder_path):
        if filename.lower().endswith('.bmp'):  # Process only .bmp files
            input_file = os.path.join(folder_path, filename)
            output_file = os.path.join(folder_path, f"{os.path.splitext(filename)[0]}_binary.txt")

            # Call img_to_bin to convert BMP to binary and save it
            img_to_bin(input_file, output_file)
            print(f"Converted {filename} to binary and saved as {output_file}")

# Example usage:
folder_path = '//home/tmsa7/Downloads/test/outputs2'  # Replace with your folder path
process_images_in_folder(folder_path)
```

Then use the GUI:



By : Mahmoud Hany Fathalla 20104375

Example for Running The main script:



Example for Running The converter script: