LAB REPORT: RSA-OAEP Cipher using CryptoPP

Instructor	**Đỗ Thị Phương Uyên**	
Student 1:	Tào Minh Đức	23520315
Student 2:	Mai Nguyễn Phúc Minh	23520930

LAB 3 REPORT'S TABLE OF CONTENTS

1. Hardware resource

Device:	Lenovo Gaming Legion 5 15IAH7H
Chip:	Intel Core i5 12500H - Cores: 12 - P-core: 4 - E-core: 8 - Logical processor: 16
Ram & Memory:	DDR5-4800 – 16GB (RAM) 512 GB SSD x2
Operating Systems:	Window 11 Ubuntu

2. Input testcase

- Making a executed program to automatically generate a random input with 6 different testcase:
    - 100 bytes input
    - 200 bytes input
    - 300 bytes input
    - 1 KB input
    - 10 KB input
    - 1 MB input
- **Note:** These testcase are generated randomly based on the program **makingtextcase.exe**

3. RSA-OAEP (Windows System)

- **Key using throughout all files:** *public.pem* (for public key) and *private.pem* (for private key)
- **Key size:** 3072
- **File encrypted:** *cipher.bin*
- **File plaintext after decrypted:** *output.txt* || *decrypted.txt*
- **Usage format:**

  + D:\CRYPTO\LAB\rsaoaep.exe gen <keysize> <format> <privateKeyFile> <publicKeyFile> (*e.g:* .\rsaoecp.exe gen 3072 PEM private.pem public.pem)

+ D:\CRYPTO\LAB\rsaoaep.exe enc <format> <publicKeyFile> <plainFile> <cipherFile> (*e.g:* .\rsaoecp.exe enc PEM public.pem random_1K.txt cipher.bin)

+D:\CRYPTO\LAB\rsaoaep.exe dec <format> <privateKeyFile> <plainFile> <cipherFile> *(e.g:* .\rsaoecp.exe dec PEM private.pem output.txt cipher.bin)

- **Abbreviations:** TT (Total Time), AT (Average Time)
- **Time counter:** Mili second (ms)
- **Execution Time (average of 10000 executions):**

	100B	**200B**	**300B**	**1KB**	**10KB**	**1MB**
Encrytion	TT: 3005 AT: 0.3005	TT: 4013 AT: 0.4013	TT: 4381 AT: 0.4381	TT: 14329 AT: 1.4329	TT: 38988 AT: 3.5988	TT: 2223489 AT: 222.3489
Decryption	TT: 944064 AT: 94.4064	TT: 1.25673e+06 AT: 125.673	TT: 1.32358e+06 AT: 132.358	TT: 1130723 AT: 113.0723	TT: 2768786 AT: 276.8796	TT: 169267852 AT: 16926.7852

- **Note:** The execution time on big files (10KB and 1MB) would be bigger due to the complexity of computing.

4. **RSA-OAEP (Linux System)**

- **Key using throughout all files:** *public.pem* (for public key) and *private.pem* (for private key)
- **Key size:** 3072
- **File encrypted:** *cipher.bin*
- **File plaintext after decrypted:** *output.txt* || *decrypted.txt*
- **Usage format:**

+ ./rsaoaep.exe gen <keysize> <format> <privateKeyFile> <publicKeyFile> (*e.g:* .\rsaoecp.exe gen 3072 PEM private.pem public.pem)

+ ./rsaoaep.exe enc <format> <publicKeyFile> <plainFile> <cipherFile> (*e.g:* .\rsaoecp.exe enc PEM public.pem random_1K.txt cipher.bin)

+ ./rsaoaep.exe dec <format> <privateKeyFile> <plainFile> <cipherFile> *(e.g:* .\rsaoecp.exe dec PEM private.pem output.txt cipher.bin)

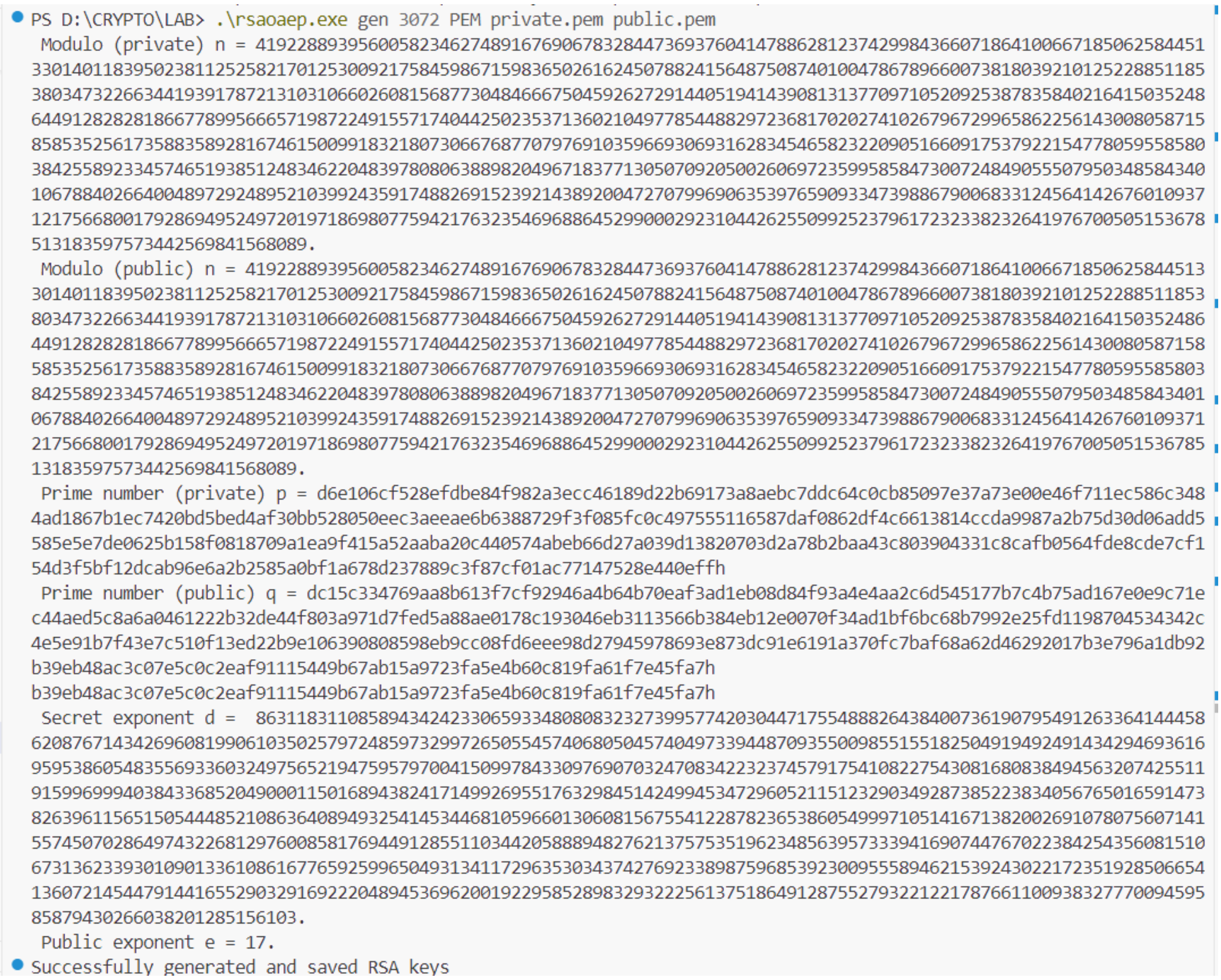
- **Abbreviations:** TT (Total Time), AT (Average Time)
- **Time counter:** Mili second (ms)
- **Execution Time (average of 10000 executions):**

	100B	**200B**	**300B**	**1KB**	**10KB**	**1MB**
Encrytion	TT: 2573 AT: 0.2573	TT: 3022 AT: 0.3022	TT: 3094 AT: 0.3094	TT: 1914 AT: 0.1914	TT: 14940 AT: 1.494	TT: 1.49644e+06 AT: 149.644
Decryption	TT: 519651 AT: 51,9651	TT: 519974 AT: 51.9974	TT: 520512 AT: 52.0512	TT: -- AT: --	TT: -- AT: --	TT: -- AT: --

- **Note:** The execution time on big files (1KB, 10KB and 1MB) would be bigger due to the complexity of computing.

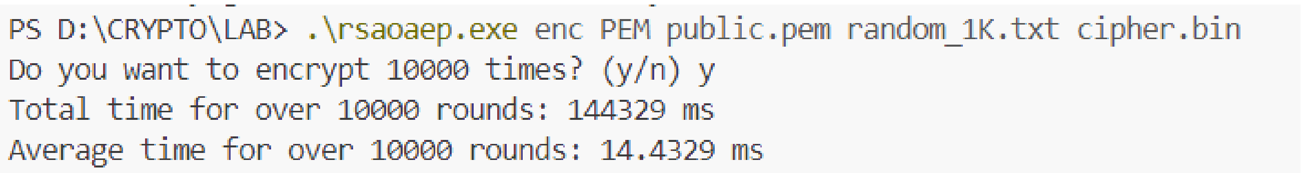
5. **Sample images**

- **Generate public/private key:**



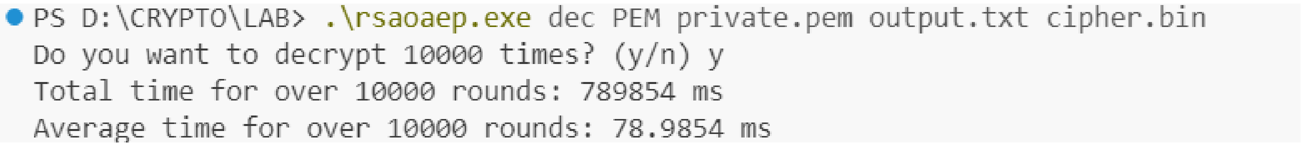
```
PS D:\CRYPTO\LAB> .\rsaoaep.exe gen 3072 PEM private.pem public.pem
 Modulo (private) n = 41922889395600582346274891676906783284473693760414788628123742998436607186410066718506258445 1
3301401183950238112525821701253009217584598671598365026162450788241564875087401004786789660073818039210125228851185
3803473226634419391787213103106602608156877304846667504592627291440519414390813137709710520925387835840216415035248
6449128282818667789956665719872249155717404425023537136021049778544882972368170202741026796729965862256143008058715
8585352561735883589281674615009918321807306676877079769103596693069316283454658232209051660917537922154778059558580
3842558923345746519385124834622048397808063889820496718377130507092050026069723599585847300724849055507950348584340
1067884026640048972924895210399243591748826915239214389200472707996906353976590933473988679006833124564142676010937
1217566800179286949524972019718698077594217632354696886452990002923104426255099252379617232338232641976700505153678
5131835975734425698415680 89.
 Modulo (public) n = 419228893956005823462748916769067832844736937604147886281237429984366071864100667185062584451 3
3014011839502381125258217012530092175845986715983650261624507882415648750874010047867896600738180392101252288511853
8034732266344193917872131031066026081568773048466675045926272914405194143908131377097105209253878358402164150352486
4491282828186677899566657198722491557174044250235371360210497785448829723681702027410267967299658622561430080587158
5853525617358835892816746150099183218073066768770797691035966930693162834546582322090516609175379221547780595585803
8425589233457465193851248346220483978080638898204967183771305070920500260697235995858473007248490555079503485843401
0678840266400489729248952103992435917488269152392143892004727079969063539765909334739886790068331245641426760109371
2175668001792869495249720197186980775942176323546968864529900029231044262550992523796172323382326419767005051536785
1318359757344256984156808 9.
 Prime number (private) p = d6e106cf528efdbe84f982a3ecc46189d22b69173a8aebc7ddc64c0cb85097e37a73e00e46f711ec586c348
4ad1867b1ec7420bd5bed4af30bb528050eec3aeeae6b6388729f3f085fc0c497555116587daf0862df4c6613814ccda9987a2b75d30d06add5
585e5e7de0625b158f0818709a1ea9f415a52aaba20c440574abeb66d27a039d13820703d2a78b2baa43c803904331c8cafb0564fde8cde7cf1
54d3f5bf12dcab96e6a2b2585a0bf1a678d237889c3f87cf01ac77147528e440effh
 Prime number (public) q = dc15c334769aa8b613f7cf92946a4b64b70eaf3ad1eb08d84f93a4e4aa2c6d545177b7c4b75ad167e0e9c71e
c44aed5c8a6a0461222b32de44f803a971d7fed5a88ae0178c193046eb3113566b384eb12e0070f34ad1bf6bc68b7992e25fd1198704534342c
4e5e91b7f43e7c510f13ed22b9e106390808598eb9cc08fd6eee98d27945978693e873dc91e6191a370fc7baf68a62d46292017b3e796a1db92
b39eb48ac3c07e5c0c2eaf91115449b67ab15a9723fa5e4b60c819fa61f7e45fa7h
b39eb48ac3c07e5c0c2eaf91115449b67ab15a9723fa5e4b60c819fa61f7e45fa7h
 Secret exponent d =  863118311085894342423306593348080832327399577420304471755488826438400736190795491263364144458
6208767143426960819906103502579724859732997265055457406805045740497339448709355009855155182504919492491434294693616
9595386054835569336032497565219475957970041509978433097690703247083422323745791754108227543081680838494563207425511
9159969994038433685204900011501689438241714992695517632984514249945347296052115123290349287385223834056765016591473
8263961156515054448521086364089493254145344681059660130608156755412287823653860549997105141671382002691078075607141
5574507028649743226812976008581769449128551103442058889482762137575351962348563957333941690744767022384254356081510
6731362339301090133610861677659259965049313411729635303437427692338987596853923009555894621539243022172351928506654
1360721454479144165529032916922204894536962001922958528983293222561375186491287552793221221787661100938327770094595
858794302660382012851561 03.
 Public exponent e = 17.
Successfully generated and saved RSA keys
```

- **Encrypted text:**



```
PS D:\CRYPTO\LAB> .\rsaoaep.exe enc PEM public.pem random_1K.txt cipher.bin
Do you want to encrypt 10000 times? (y/n) y
Total time for over 10000 rounds: 144329 ms
Average time for over 10000 rounds: 14.4329 ms
```

- **Decrypted text:**



```
PS D:\CRYPTO\LAB> .\rsaoaep.exe dec PEM private.pem output.txt cipher.bin
Do you want to decrypt 10000 times? (y/n) y
Total time for over 10000 rounds: 789854 ms
Average time for over 10000 rounds: 78.9854 ms
```

6. Conclusion

- The execution time of RSA-OAEP appears to be lower than AES in Labs 1 and 2. This is likely due to RSA-OAEP relying heavily on expensive mathematical operations involving large numbers (modular exponentiation).
- Execution times on a Linux system (over 10,000 iterations) were consistently faster than those on a Windows system. As file sizes increase, the encryption and decryption times using RSA also increase noticeably.

➔ RSA, as an asymmetric encryption algorithm (using a public-private key pair), is not efficient for encrypting large files due to its high computational overhead. For larger files (e.g., around 1MB), AES is a more suitable choice due to its simpler algorithm and significantly lower computation time compared to RSA.

➔ RSA is best used to securely exchange symmetric keys, which can then be used by AES for efficient bulk data encryption. This hybrid approach combines the strengths of both algorithms to ensure both data security and performance.