TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG

BÁO CÁO BÀI TẬP THỰC HÀNH

LAB 1 OFF CLASS

MÔN HỌC: MẬT MÃ HỌC

Họ và tên: LẠI QUAN THIÊN

Mã số sinh viên: 22521385

Lớp: NT219.O21.ANTT

TP. HỒ CHÍ MINH, THÁNG 06 NĂM 2024

MỤC LỤC

PHẦN I: TỔNG QUAN VÀ MÔ TẢ

1.1. Thông tin cá nhân:

Họ và tên: Lại Quan Thiên

Mã số sinh viên: 22521385

Lớp thực hành: NT219.O21.ANTT.1

Link github: Cryptography-Course/Labs/OffClass/Lab_1 at main · WanThinnn/Cryptography-Course (github.com)

1.2. Thông tin thiết bị:

- Thiết bị: Macbook Air 2019 – RAM 8GB (LPDDR3 2133MHz) – SSD 128GB

- Hệ điều hành:

+ Windows 11 Pro (cài đặt thông qua Bootcamp của Apple)

+ Ubuntu 22.04 Jammy Jellyfish (cài đặt thông qua t2linux.org)

- Bộ xử lý:

+ Intel Core i5 8210Y – 1.60GHz – Turbo Boost 3.60Ghz

+ Intel UHD Graphics 617

+ Apple T2 Security Chip

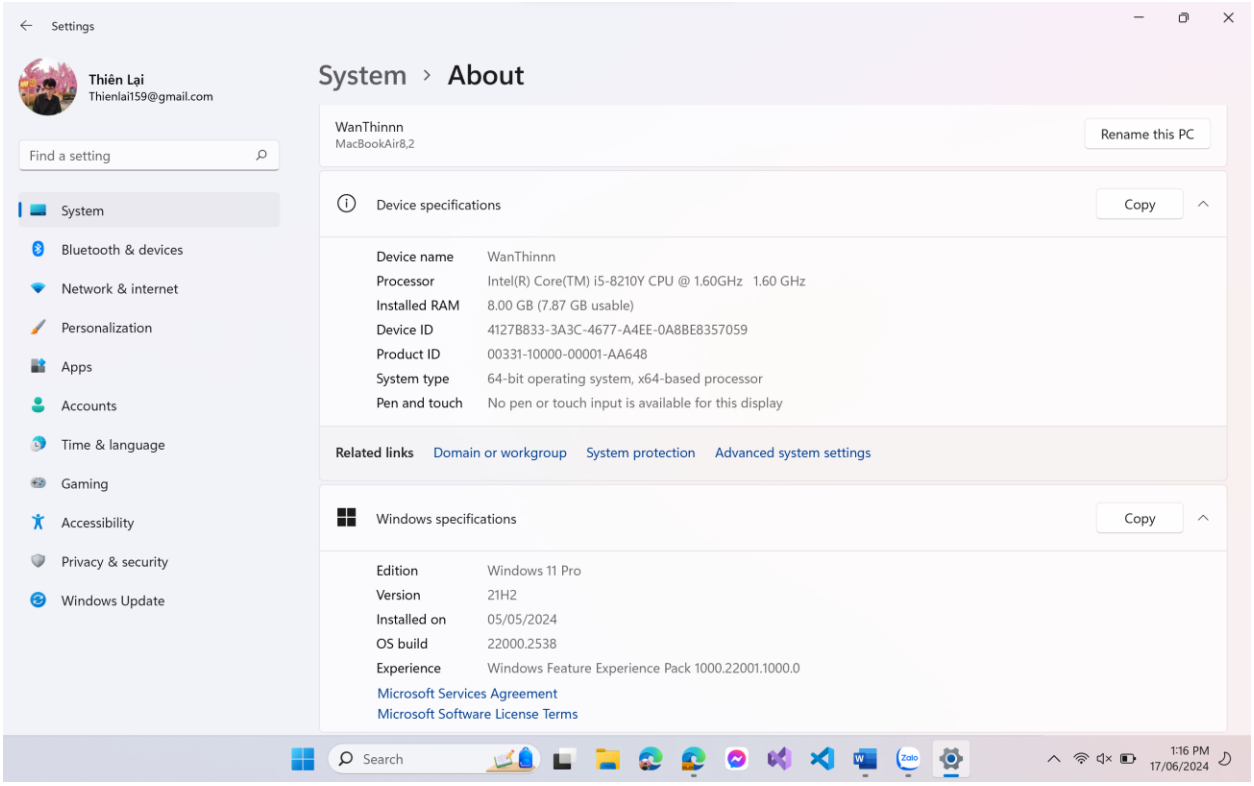
- Thông tin về bộ xử lý:

+ Intel Core i58210Y Processor 4M Cache up to 3.60 GHz Thông số kỹ thuật sản phẩm

+ Hỗ trợ Intel® UHD Graphics 617

+ Apple T2 Security Chip: Security Overview

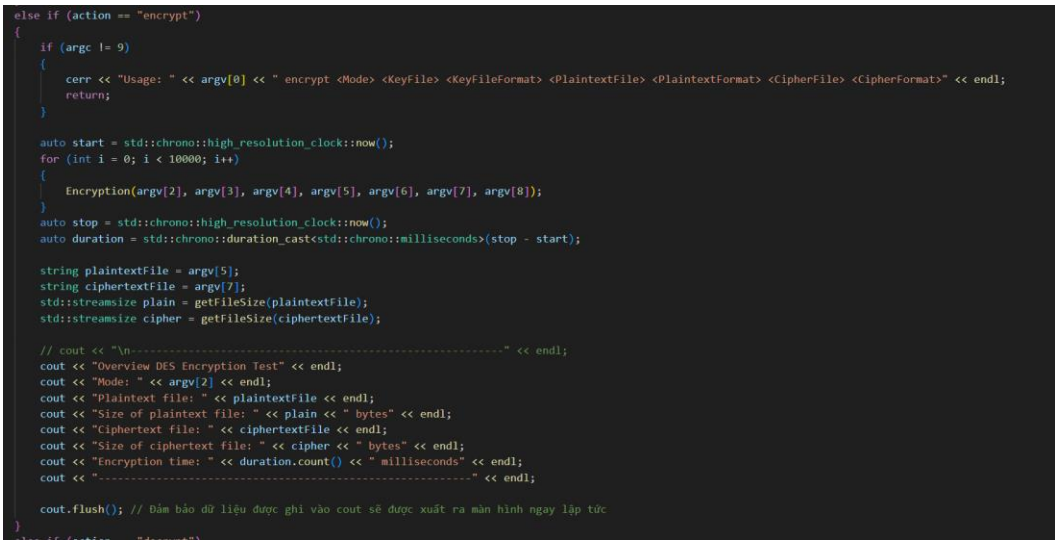
- Các thực nghiệm được thực thi trong quá trình laptop cắm điện, nhiệt độ phòng


Settings
Thiên Lại
Thienlai159@gmail.com
Find a setting
System
Bluetooth & devices
Network & internet
Personalization
Apps
Accounts
Time & language
Gaming
Accessibility
Privacy & security
Windows Update
System › About
WanThinnn
MacBookAir8,2
Rename this PC
Device specifications
Copy
Device name WanThinnn
Processor Intel(R) Core(TM) i5-8210Y CPU @ 1.60GHz 1.60 GHz
Installed RAM 8.00 GB (7.87 GB usable)
Device ID 4127B833-3A3C-4677-A4EE-0A8BE8357059
Product ID 00331-10000-00001-AA648
System type 64-bit operating system, x64-based processor
Pen and touch No pen or touch input is available for this display
Related links Domain or workgroup System protection Advanced system settings
Windows specifications
Copy
Edition Windows 11 Pro
Version 21H2
Installed on 05/05/2024
OS build 22000.2538
Experience Windows Feature Experience Pack 1000.22001.1000.0
Microsoft Services Agreement
Microsoft Software License Terms
Search
1:16 PM
17/06/2024

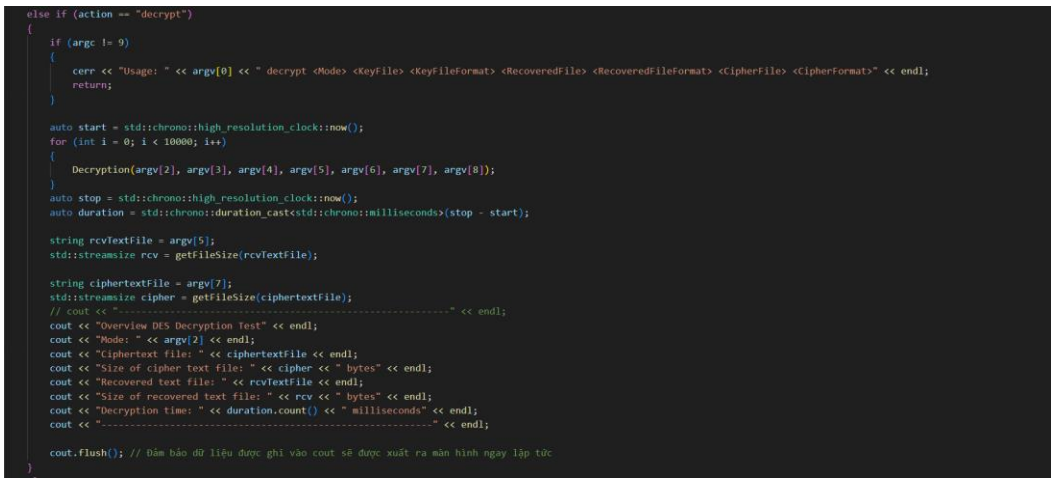
1.3. Tổng quan:

Bài báo cáo bao gồm: Báo cáo code implement thuật toán DES và AES bằng ngôn ngữ C++ và sử dụng thư viện CryptoPP. Sau khi xây dựng code thì tiến hành tạo sáu file test case với kích thước khác nhau, thực hiện đo thời gian 10 000 lần mã hóa/giải mã trên cả hai Hệ điều hành Windows và Linux. Viết bảng thống kê số liệu và vẽ biểu đồ phân tích và so sánh.

Dưới đây là hình ảnh minh hoạ phần code chạy 10 000 lần của DES (bao gồm toàn bộ quá trình load file, check format, thực hiện mã hoá/giải mã và lưu/xuất file):

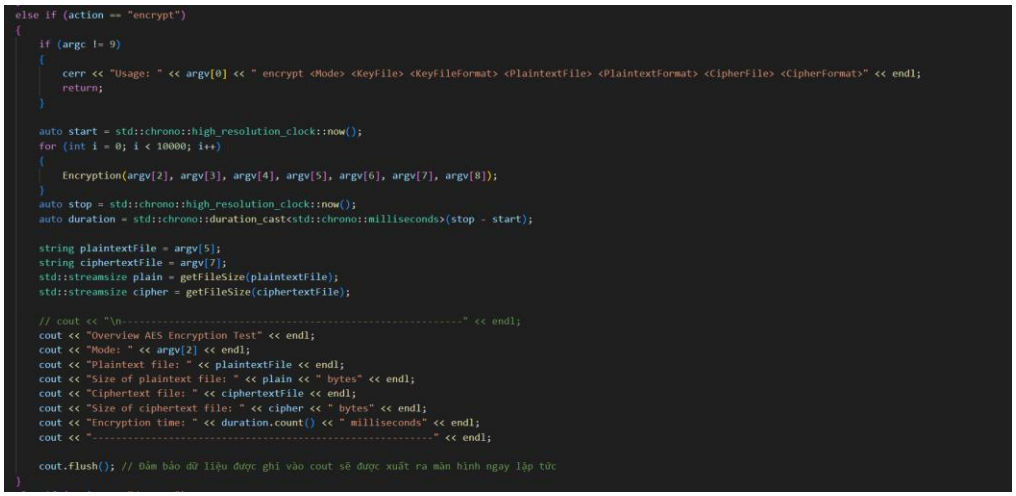


```
else if (action == "encrypt")
{
    if (argc != 9)
    {
        cerr << "Usage: " << argv[0] << " encrypt <Mode> <KeyFile> <KeyFileFormat> <PlaintextFile> <PlaintextFormat> <CipherFile> <CipherFormat>" << endl;
        return;
    }

    auto start = std::chrono::high_resolution_clock::now();
    for (int i = 0; i < 10000; i++)
    {
        Encryption(argv[2], argv[3], argv[4], argv[5], argv[6], argv[7], argv[8]);
    }
    auto stop = std::chrono::high_resolution_clock::now();
    auto duration = std::chrono::duration_cast<std::chrono::milliseconds>(stop - start);

    string plaintextFile = argv[5];
    string ciphertextFile = argv[7];
    std::streamsize plain = getFileSize(plaintextFile);
    std::streamsize cipher = getFileSize(ciphertextFile);

    // cout << "\n------------------------------------------------------------" << endl;
    cout << "Overview DES Encryption Test" << endl;
    cout << "Mode: " << argv[2] << endl;
    cout << "Plaintext file: " << plaintextFile << endl;
    cout << "Size of plaintext file: " << plain << " bytes" << endl;
    cout << "Ciphertext file: " << ciphertextFile << endl;
    cout << "Size of ciphertext file: " << cipher << " bytes" << endl;
    cout << "Encryption time: " << duration.count() << " milliseconds" << endl;
    cout << "------------------------------------------------------------" << endl;

    cout.flush(); // Đảm bảo dữ liệu được ghi vào cout sẽ được xuất ra màn hình ngay lập tức
}
```

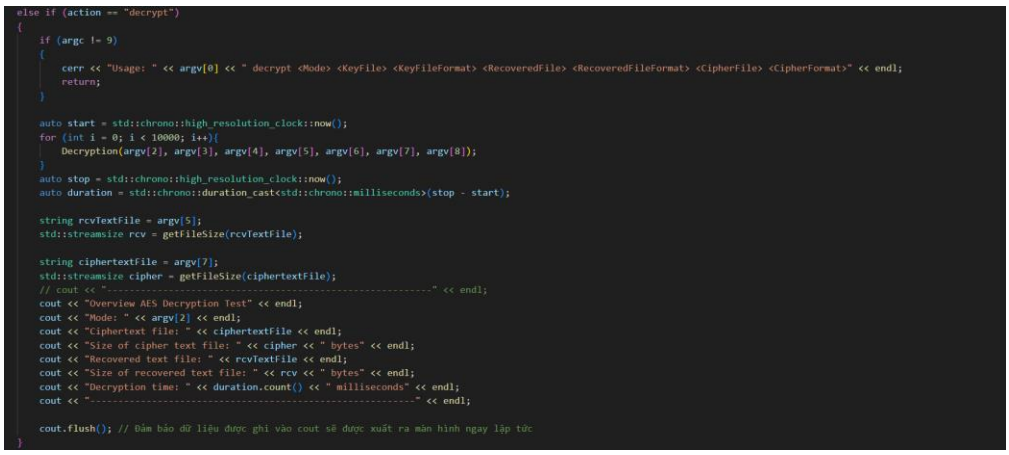


```
else if (action == "decrypt")
{
    if (argc != 9)
    {
        cerr << "Usage: " << argv[0] << " decrypt <Mode> <KeyFile> <KeyFileFormat> <RecoveredFile> <RecoveredFileFormat> <CipherFile> <CipherFormat>" << endl;
        return;
    }

    auto start = std::chrono::high_resolution_clock::now();
    for (int i = 0; i < 10000; i++)
    {
        Decryption(argv[2], argv[3], argv[4], argv[5], argv[6], argv[7], argv[8]);
    }
    auto stop = std::chrono::high_resolution_clock::now();
    auto duration = std::chrono::duration_cast<std::chrono::milliseconds>(stop - start);

    string rcvTextFile = argv[5];
    std::streamsize rcv = getFileSize(rcvTextFile);

    string ciphertextFile = argv[7];
    std::streamsize cipher = getFileSize(ciphertextFile);
    // cout << "------------------------------------------------------------" << endl;
    cout << "Overview DES Decryption Test" << endl;
    cout << "Mode: " << argv[2] << endl;
    cout << "Ciphertext file: " << ciphertextFile << endl;
    cout << "Size of cipher text file: " << cipher << " bytes" << endl;
    cout << "Recovered text file: " << rcvTextFile << endl;
    cout << "Size of recovered text file: " << rcv << " bytes" << endl;
    cout << "Decryption time: " << duration.count() << " milliseconds" << endl;
    cout << "------------------------------------------------------------" << endl;

    cout.flush(); // Đảm bảo dữ liệu được ghi vào cout sẽ được xuất ra màn hình ngay lập tức
}
```

Dưới đây là hình ảnh minh hoạ phần code chạy 10 000 lần của AES (bao gồm toàn bộ quá trình load file, check format, thực hiện mã hoá/giải mã và lưu/xuất file):



```
else if (action == "encrypt")
{
    if (argc != 9)
    {
        cerr << "Usage: " << argv[0] << " encrypt <Mode> <KeyFile> <KeyFileFormat> <PlaintextFile> <PlaintextFormat> <CipherFile> <CipherFormat>" << endl;
        return;
    }

    auto start = std::chrono::high_resolution_clock::now();
    for (int i = 0; i < 10000; i++)
    {
        Encryption(argv[2], argv[3], argv[4], argv[5], argv[6], argv[7], argv[8]);
    }
    auto stop = std::chrono::high_resolution_clock::now();
    auto duration = std::chrono::duration_cast<std::chrono::milliseconds>(stop - start);

    string plaintextFile = argv[5];
    string ciphertextFile = argv[7];
    std::streamsize plain = getFileSize(plaintextFile);
    std::streamsize cipher = getFileSize(ciphertextFile);

    // cout << "\n------------------------------------------------------------" << endl;
    cout << "Overview AES Encryption Test" << endl;
    cout << "Mode: " << argv[2] << endl;
    cout << "Plaintext file: " << plaintextFile << endl;
    cout << "Size of plaintext file: " << plain << " bytes" << endl;
    cout << "Ciphertext file: " << ciphertextFile << endl;
    cout << "Size of ciphertext file: " << cipher << " bytes" << endl;
    cout << "Encryption time: " << duration.count() << " milliseconds" << endl;
    cout << "------------------------------------------------------------" << endl;

    cout.flush(); // Đảm bảo dữ liệu được ghi vào cout sẽ được xuất ra màn hình ngay lập tức
}
```



```
else if (action == "decrypt")
{
    if (argc != 9)
    {
        cerr << "Usage: " << argv[0] << " decrypt <Mode> <KeyFile> <KeyFileFormat> <RecoveredFile> <RecoveredFileFormat> <CipherFile> <CipherFormat>" << endl;
        return;
    }

    auto start = std::chrono::high_resolution_clock::now();
    for (int i = 0; i < 10000; i++){
        Decryption(argv[2], argv[3], argv[4], argv[5], argv[6], argv[7], argv[8]);
    }
    auto stop = std::chrono::high_resolution_clock::now();
    auto duration = std::chrono::duration_cast<std::chrono::milliseconds>(stop - start);

    string rcvTextFile = argv[5];
    std::streamsize rcv = getFileSize(rcvTextFile);

    string ciphertextFile = argv[7];
    std::streamsize cipher = getFileSize(ciphertextFile);
    // cout << "------------------------------------------------------------" << endl;
    cout << "Overview AES Decryption Test" << endl;
    cout << "Mode: " << argv[2] << endl;
    cout << "Ciphertext file: " << ciphertextFile << endl;
    cout << "Size of cipher text file: " << cipher << " bytes" << endl;
    cout << "Recovered text file: " << rcvTextFile << endl;
    cout << "Size of recovered text file: " << rcv << " bytes" << endl;
    cout << "Decryption time: " << duration.count() << " milliseconds" << endl;
    cout << "------------------------------------------------------------" << endl;

    cout.flush(); // Đảm bảo dữ liệu được ghi vào cout sẽ được xuất ra màn hình ngay lập tức
}
```

PHẦN II: NỘI DUNG THỰC HÀNH

2.1. DES - Data Encryption Standard:

2.1.1. Bảng thống kê số liệu thực nghiệm:

Dưới đây là bảng thống kê chi tiết thời gian trung bình encrypt/decrypt của từng mode (đơn vị thời gian: ms):

2.1.1.1. Kết quả thực nghiệm trên hệ điều hành Windows 11:

- **Bảng thời gian trung bình 10 000 lần Encrypt:**

Encrypt	ECB	CBC	OFB	CFB	CTR
File 1 (1KB)	0.98	1.1351	1.1623	1.1521	1.19
File 2 (10KB)	3.9404	4.5583	4.9043	5.2401	4.8271
File 3 (55KB)	7.7573	8.5082	8.2156	8.2664	8.3443
File 4 (104KB)	8.0378	8.2346	8.4647	8.5566	8.4029
File 5 (1.1MB)	24.1757	24.4698	23.5941	23.6998	27.1209
File 6 (5.2MB)	101.7406	106.9557	106.26265	105.0248	107.253

- **Bảng thời gian trung bình 10 000 lần Decrypt:**

Decrypt	ECB	CBC	OFB	CFB	CTR
File 1 (1KB)	0.9431	1.0953	1.1466	1.1174	1.1616
File 2 (10KB)	5.1986	5.5492	5.8493	5.8364	5.54
File 3 (55KB)	9.6851	10.5759	10.1426	10.3866	10.198
File 4 (104KB)	9.2663	10.1873	9.8296	9.5773	10.1067
File 5 (1.1MB)	28.6098	28.3298	33.2239	28.6242	34.0654
File 6 (5.2MB)	135.79555	143.1733	154.31865	148.6077	161.1258

2.1.1.2. Kết quả thực nghiệm trên hệ điều hành Ubuntu 22.04 (Linux):

- **Bảng thời gian trung bình 10 000 lần Encrypt:**

Encrypt	ECB	CBC	OFB	CFB	CTR
File 1 (1KB)	0.2805	0.2945	0.2873	0.3057	0.2926
File 2 (10KB)	0.4856	0.4987	0.5151	0.4682	0.5017
File 3 (55KB)	1.4377	1.7453	1.7192	1.7373	1.7764
File 4 (104KB)	2.6386	3.3966	3.8182	3.4255	3.9295
File 5 (1.1MB)	22.1663	23.4587	21.213	21.333	21.3538
File 6 (5.2MB)	99.90789	101.12299	103.42994	104.19209	105.42029

- **Bảng thời gian trung bình 10 000 lần Decrypt:**

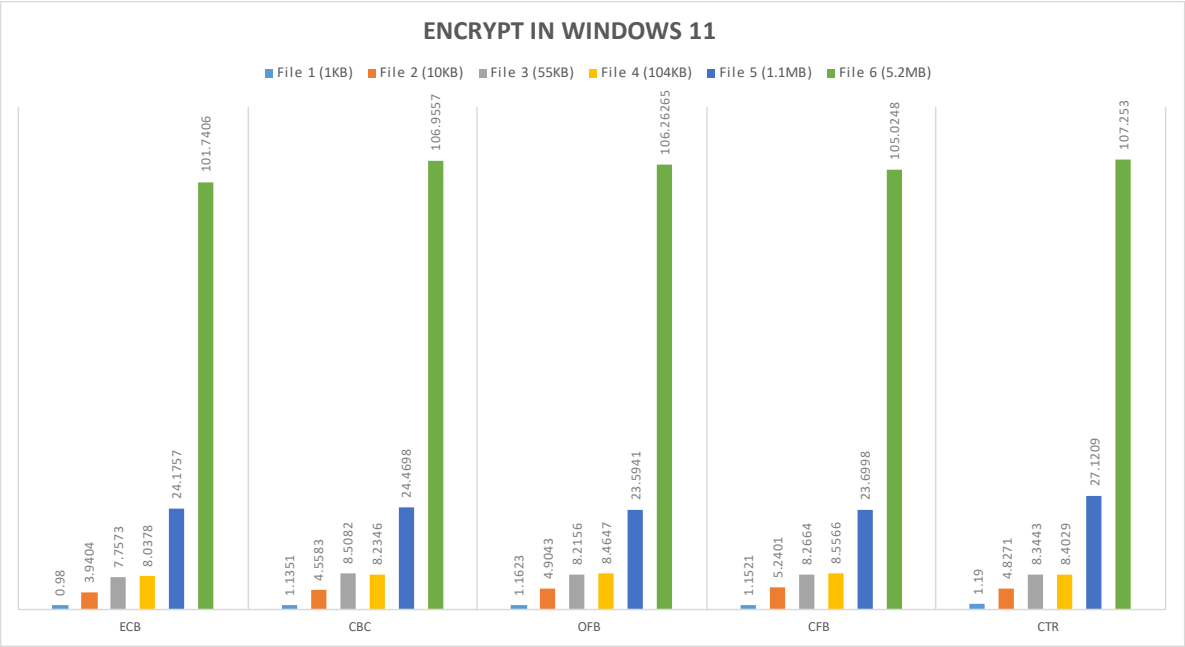
Decrypt	ECB	CBC	OFB	CFB	CTR
File 1 (1KB)	0.2614	0.2739	0.2738	0.2935	0.298
File 2 (10KB)	0.508	0.5075	0.5128	0.5171	0.5166
File 3 (55KB)	1.689	1.6975	1.7579	1.737	1.7861
File 4 (104KB)	3.4042	3.2515	3.3306	3.3085	3.8242
File 5 (1.1MB)	19.3635	19.3624	20.2751	19.9875	20.9006
File 6 (5.2MB)	110.7425	112.296	116.66625	115.7128	118.6979

2.1.2. Biểu đồ cột so sánh các mode:

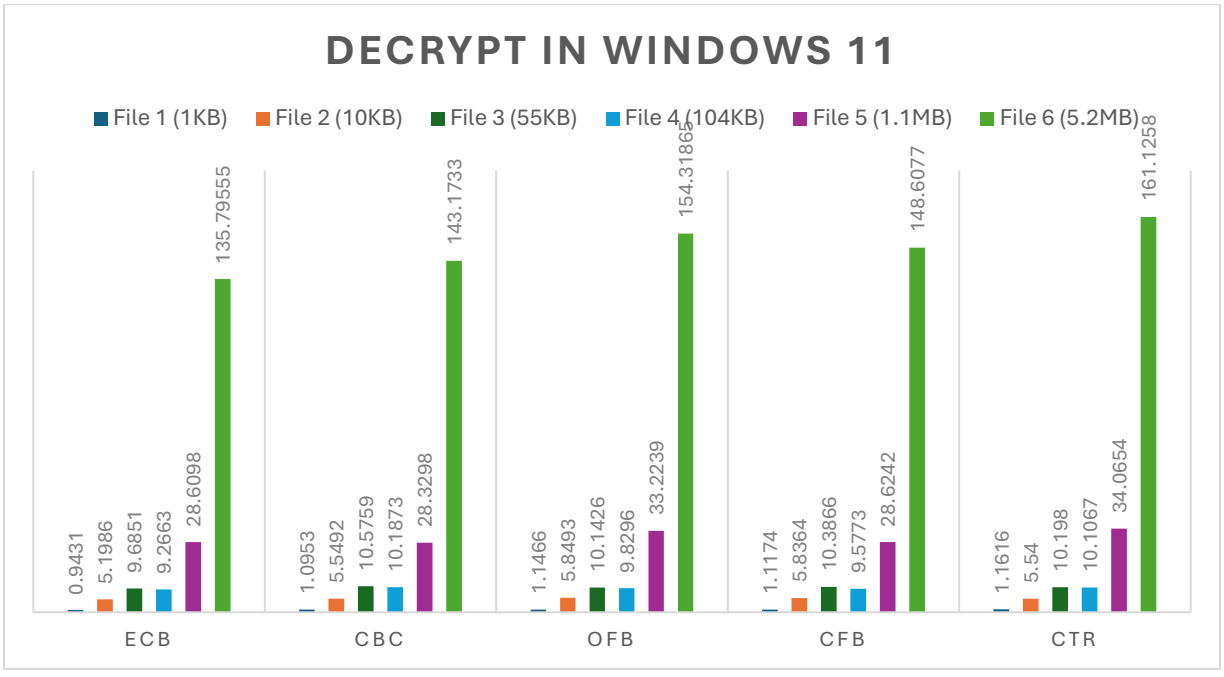
Dựa theo số liệu từ các bảng ở trên, vẽ biểu đồ để dễ dàng so sánh một cách trực quang:

2.1.2.1. Hệ điều hành Windows 11:

- **Biểu đồ thời gian trung bình 10 000 lần Encrypt:**

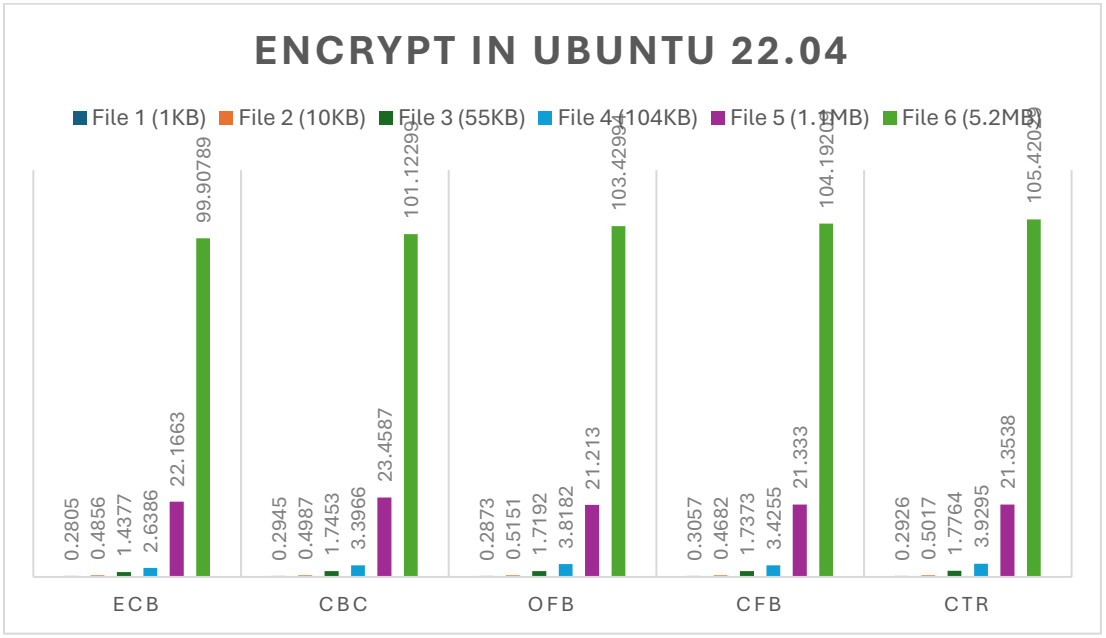


- **Biểu đồ thời gian trung bình 10 000 lần Decrypt:**

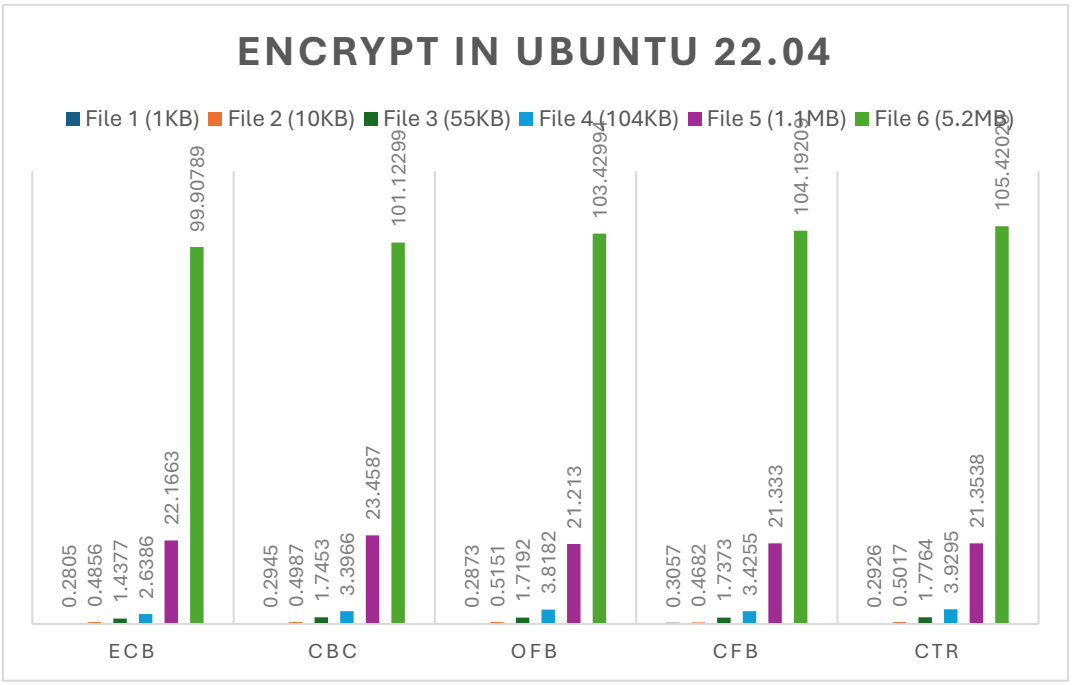


2.1.2.2. Hệ điều hành Ubuntu 22.04 (Linux):

- **Biểu đồ thời gian trung bình 10 000 lần Encrypt:**

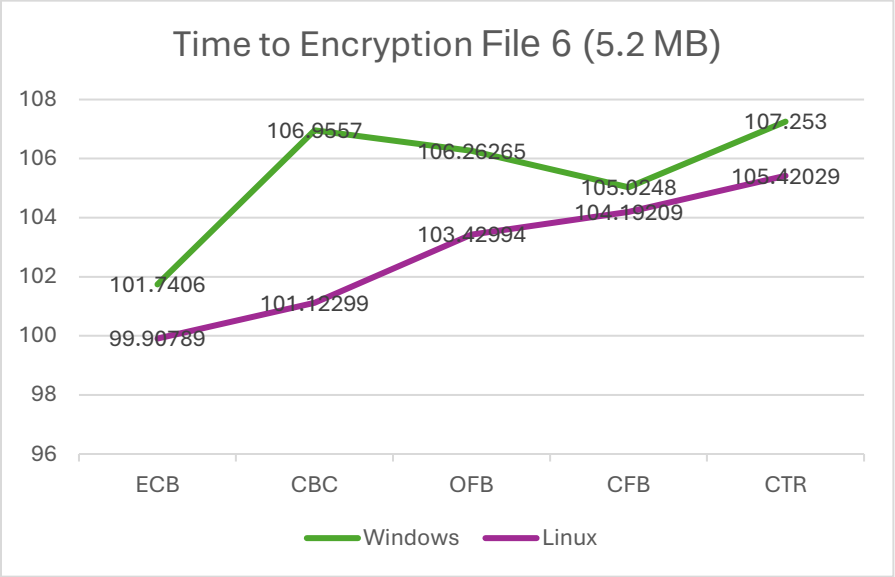


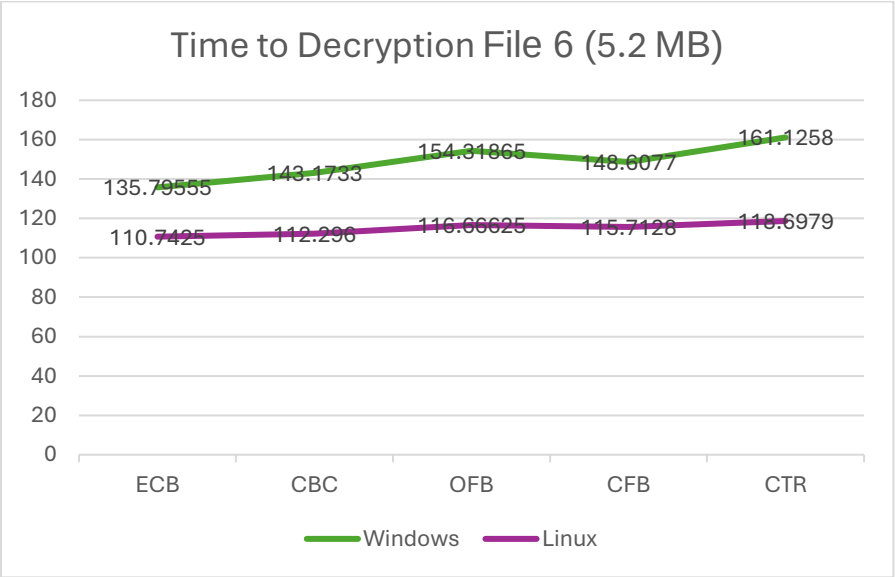
- **Biểu đồ thời gian trung bình 10 000 lần Decrypt:**



2.1.3. Biểu đồ đường so sánh hai hệ điều hành:

Dưới đây là biểu đồ so sánh thời gian mã hoá và giải mã file 5.2 MB trên 2 hệ điều hành:





2.1.4. Phân tích và so sánh:

2.1.4.1. Windows:

- Hiệu suất: Thời gian mã hóa và giải mã cho tất cả các chế độ đều tăng theo kích thước tệp.

- So sánh các chế độ:

+ Các chế độ **ECB và CBC** là những chế độ nhanh hơn đối với các tệp nhỏ hơn, nhưng thời gian của chúng tăng lên đáng kể đối với các tệp lớn hơn.

+ Các chế độ **CTR và CFB, OFB** có thời gian trung bình sấp xỉ nhau.

- Ảnh hưởng của kích thước tệp: Ảnh hưởng của kích thước tệp đến thời gian mã hóa và giải mã là rõ ràng, với các tệp lớn hơn cho thấy sự gia tăng đáng kể về thời gian trong tất cả các chế độ.

=> Những quan sát này cho thấy rằng mặc dù DES có hiệu suất nhất quán trên các chế độ khác nhau, nhưng việc lựa chọn chế độ và kích thước tệp là những yếu tố quan trọng trong việc xác định thời gian mã hóa và giải mã.

2.1.4.2. Linux:

- Hiệu suất: Thời gian mã hóa và giải mã cho tất cả các chế độ đều tăng theo kích thước tệp.

- So sánh các chế độ:

+ Tương tự như Windows 11, các chế độ **ECB và CBC** là những chế độ nhanh hơn đối với các tệp nhỏ hơn, nhưng thời gian của chúng tăng lên đáng kể đối với các tệp lớn hơn.

+ Các chế độ **CTR và CFB, OFB** có thời gian trung bình sấp xỉ nhau.

- Ảnh hưởng của kích thước tệp: Ảnh hưởng của kích thước tệp đến thời gian mã hóa và giải mã là rõ ràng, với các tệp lớn hơn cho thấy sự gia tăng đáng kể về thời gian trong tất cả các chế độ.

2.1.4.3. Kết luận:

- Các biểu đồ cho thấy thời gian mã hóa của một file trên các chế độ mã hóa khác nhau trên Windows và Linux thì Linux mã hóa nhanh hơn Windows đáng kể. Điều này có thể do Linux quản lý tài nguyên và tối ưu hệ thống tốt hơn.

- Kết quả này cho thấy Linux có hiệu suất mã hóa và giải mã tốt hơn so với Windows. Việc lựa chọn hệ điều hành và chế độ mã hóa phù hợp là quan trọng để đạt hiệu suất tối ưu trong các ứng dụng yêu cầu mã hóa nhanh chóng.

2.2. AES - Advanced Encryption Standard:

2.2.1. Bảng thống kê số liệu thực nghiệm:

Dưới đây là bảng thống kê chi tiết thời gian encrypt/decrypt của từng mode (đơn vị thời gian: ms):

2.2.1.1. Hệ điều hành Windows 11:

- **Bảng thời gian trung bình 10 000 lần Encrypt:**

Encrypt	ECB	CBC	OFB	CFB	CTR	XTS	CCM	GCM
File 1 (1KB)	1.1514	1.3187	1.3449	1.3084	1.3866	1.3368	1.3127	1.3292
File 2 (10KB)	1.7563	2.1875	2.0333	2.1445	2.0722	2.1856	2.1816	2.2053
File 3 (55KB)	5.5953	5.8153	5.5635	5.7321	5.7912	6.6598	6.222	6.2793
File 4 (104KB)	7.6234	7.7244	7.5693	7.7069	7.5443	7.6476	7.6566	7.5026
File 5 (1.1MB)	11.4867	10.7627	10.1829	10.4475	9.7825	9.773	9.8753	9.7124
File 6 (5.2MB)	30.0083	35.2831	36.0848	35.2688	30.3148	34.5423	40.816	33.4737

- **Bảng thời gian trung bình 10 000 lần Decrypt:**

Decrypt	ECB	CBC	OFB	CFB	CTR	XTS	CCM	GCM
File 1 (1KB)	1.1051	1.2841	1.4279	1.3688	1.3875	1.4071	1.4223	1.4411
File 2 (10KB)	2.2222	2.1544	2.1665	2.2451	2.212	2.2755	2.4698	2.4227
File 3 (55KB)	5.4448	5.9187	8.3955	7.1648	7.7555	8.4929	8.2476	8.5755
File 4 (104KB)	8.0296	8.7632	8.5017	8.671	8.4657	8.5169	8.6451	8.5791
File 5 (1.1MB)	14.482	13.5903	13.2502	13.3955	12.9302	12.9603	12.9796	13.9026
File 6 (5.2MB)	43.859	43.1083	49.5683	44.8405	46.6597	52.5176	57.7184	48.2429

2.2.1.2. Hệ điều hành Ubuntu 22.04 (Linux):

- **Bảng thời gian trung bình 10 000 lần Encrypt:**

Encrypt	ECB	CBC	OFB	CFB	CTR	XTS	CCM	GCM
File 1 (1KB)	0.6893	0.7716	0.7683	0.8458	0.6599	0.6184	0.9708	0.7388
File 2 (10KB)	0.2353	0.2415	0.264	0.2269	0.2711	0.2949	0.3133	0.2815
File 3 (55KB)	0.3976	0.5019	0.5358	0.481	0.506	0.6064	0.6019	0.5172
File 4 (104KB)	0.7265	0.8674	1.1848	1.0232	0.9275	1.3348	1.0665	1.0308
File 5 (1.1MB)	5.8846	9.6107	8.0074	7.4834	6.1594	7.4527	7.5456	6.1225
File 6 (5.2MB)	25.4632	29.4238	29.8805	30.3666	24.0249	30.9784	31.5544	24.7856

- **Bảng thời gian trung bình 10 000 lần Decrypt:**

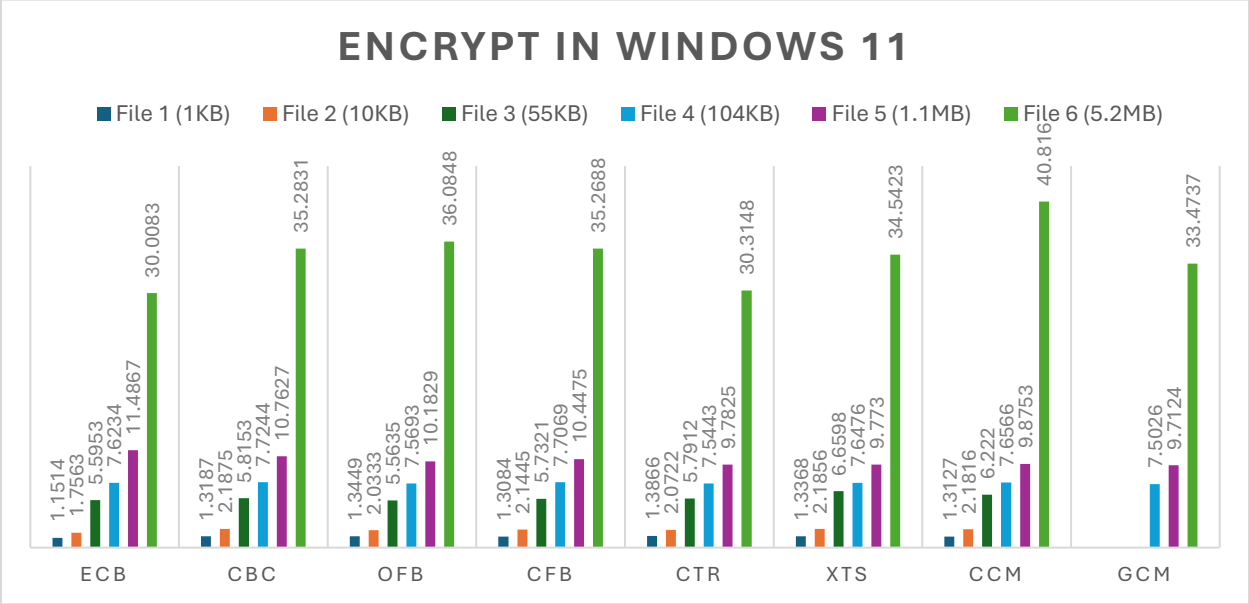
Decrypt	ECB	CBC	OFB	CFB	CTR	XTS	CCM	GCM
File 1 (1KB)	0.1868	0.1946	0.1908	0.202	0.2063	0.1931	0.2042	0.2141
File 2 (10KB)	0.2402	0.2605	0.2526	0.243	0.2514	0.2632	0.3342	0.283
File 3 (55KB)	0.3838	0.4274	0.5818	0.4568	0.5378	0.6799	0.643	0.5239
File 4 (104KB)	0.5476	0.5832	0.8272	0.6635	0.7058	0.8142	1.0147	0.7422
File 5 (1.1MB)	5.3888	5.6533	6.258	5.668	5.6687	6.2976	6.6554	5.8877
File 6 (5.2MB)	25.7986	26.3224	27.0644	26.179	25.9685	26.5759	28.8709	26.108

2.2.2. Biểu đồ cột so sánh các mode:

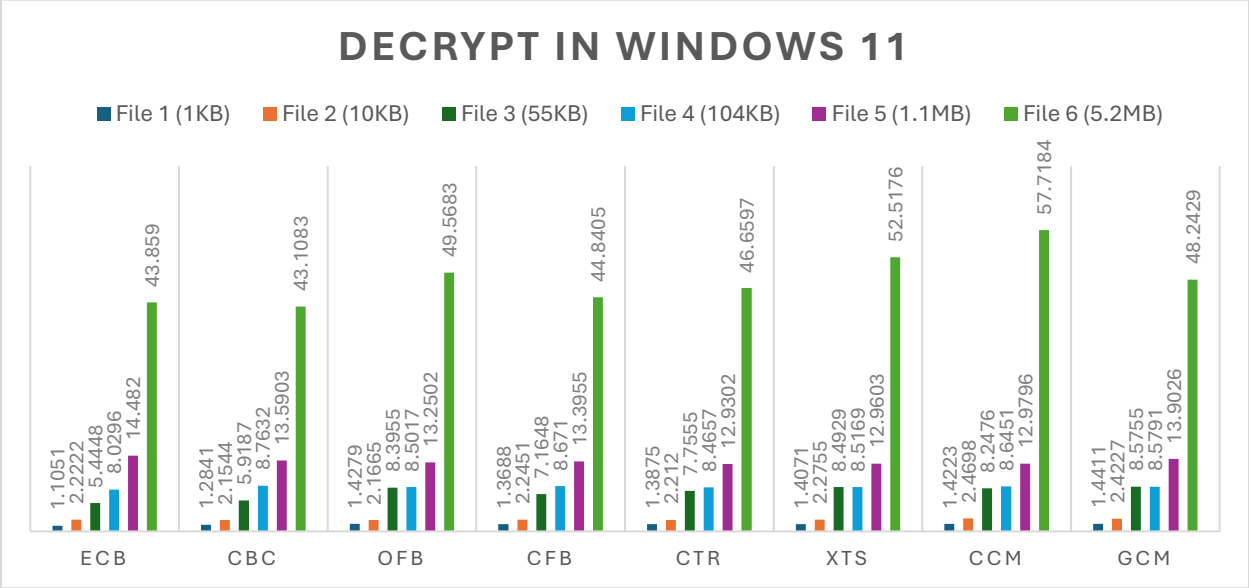
Dựa theo số liệu từ các bảng ở trên, vẽ biểu đồ để dễ dàng so sánh một cách trực quang:

2.2.2.1. Hệ điều hành Windows 11:

- **Biểu đồ thời gian trung bình 10 000 lần Encrypt:**

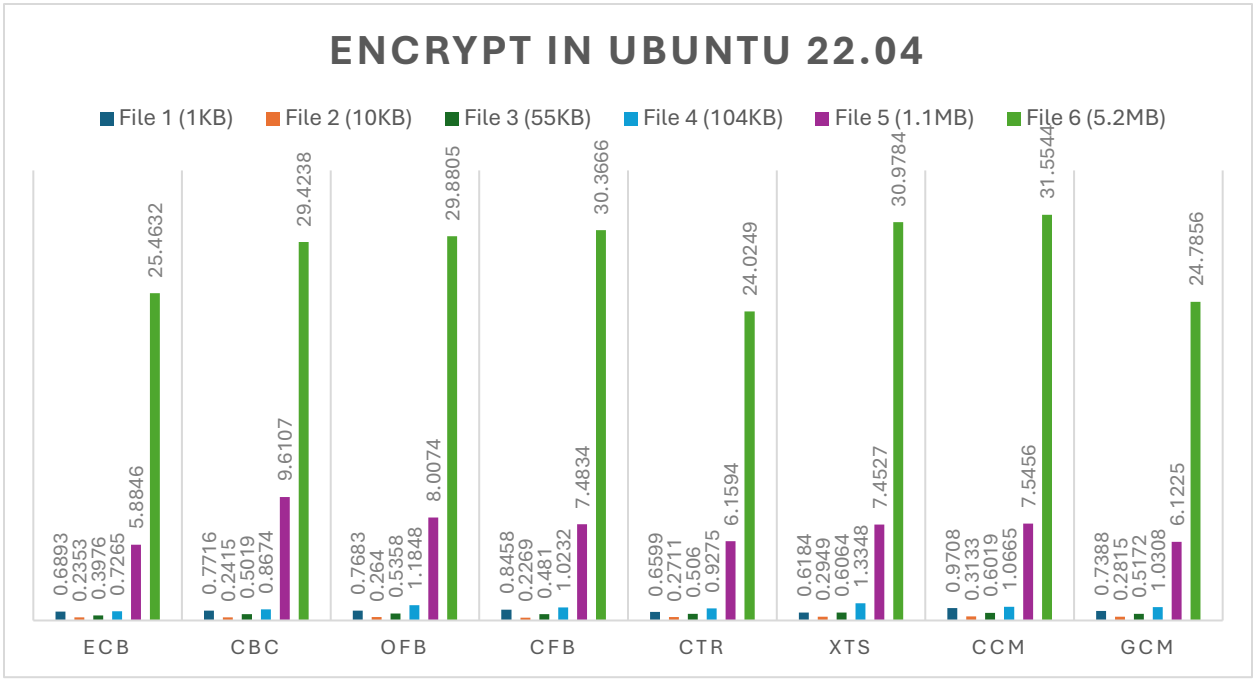


- **Biểu đồ thời gian trung bình 10 000 lần Decrypt:**

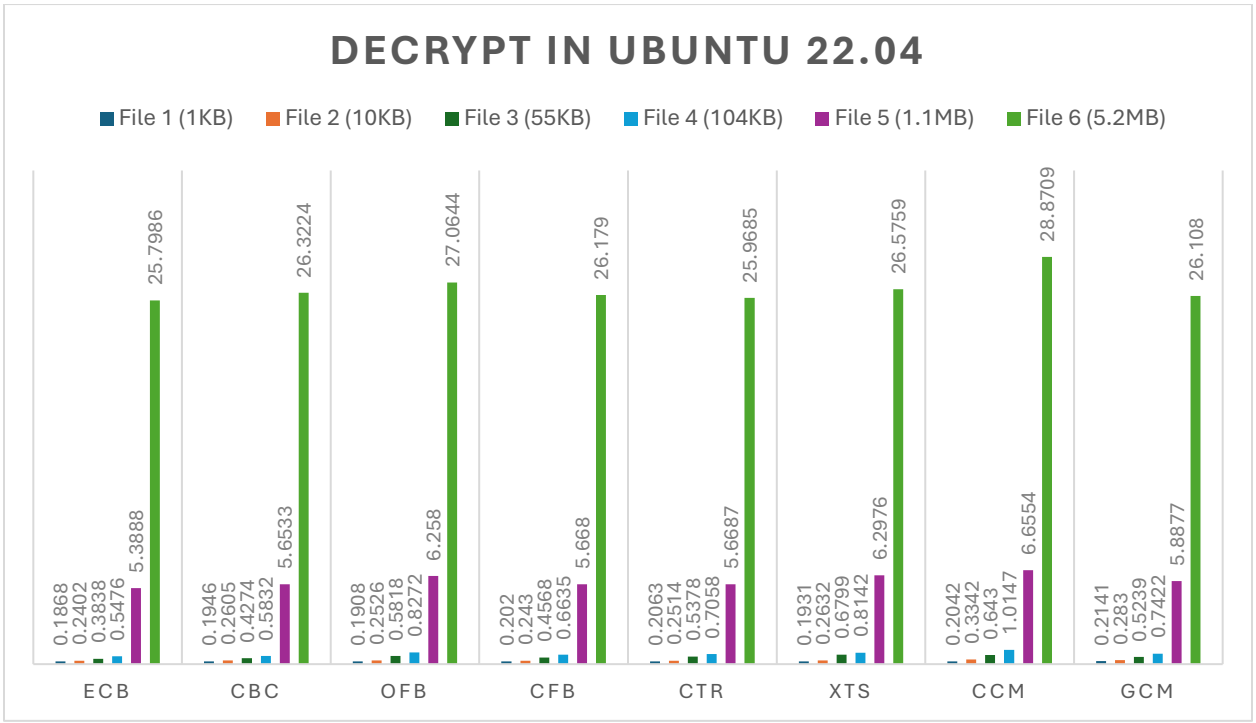


2.2.2.1. Hệ điều hành Ubuntu 22.04 (Linux):

- **Biểu đồ thời gian trung bình 10 000 lần Encrypt:**

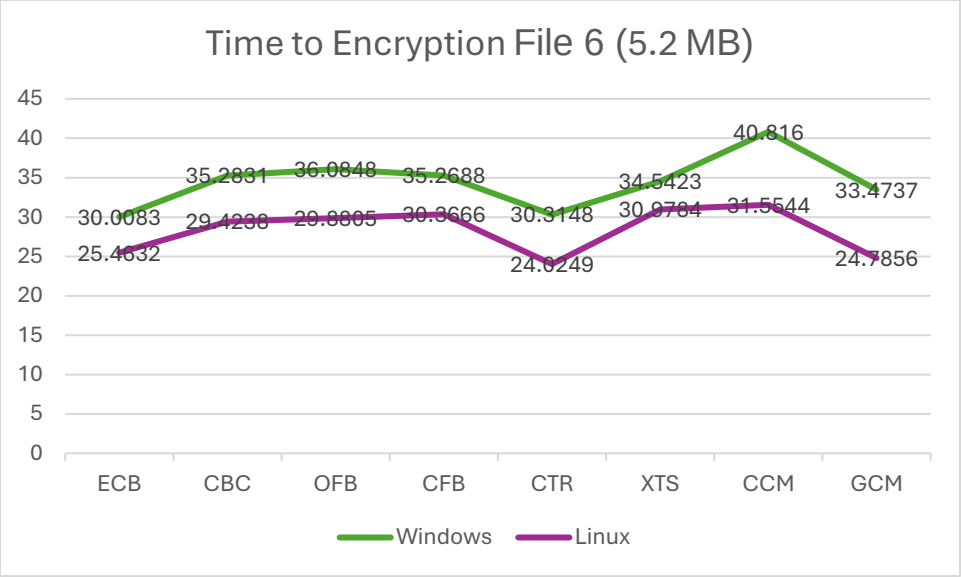


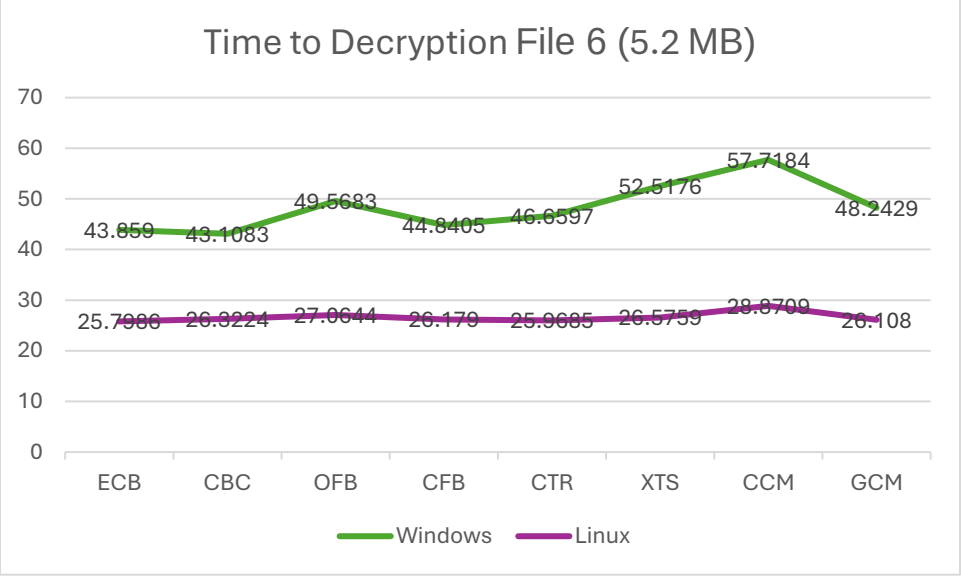
- **Biểu đồ thời gian trung bình 10 000 lần Decrypt:**



2.2.3. Biểu đồ đường so sánh hai hệ điều hành:

Dưới đây là biểu đồ so sánh thời gian mã hoá và giải mã file 5.2 MB trên 2 hệ điều hành:





2.2.4. Phân tích và so sánh:

Theo thống kê, thời gian mã hóa và giải mã giữa các mode mã hóa không chênh lệch lớn, thường nằm trong khoảng từ 0.1 đến 1.0 ms. Do đó, để dễ dàng nhận thấy sự khác biệt, ta chỉ cần chọn một trong hai quá trình này để so sánh giữa Linux và Windows, và lựa chọn file lớn nhất trong các test case.

Xét về thời gian giữa các mode, mode ECB gần như là nhanh nhất. Nguyên nhân là vì ECB mã hóa từng block riêng lẻ, không có các thao tác phức tạp như XOR với IV như các mode khác. Do ECB xử lý các block độc lập với nhau, các khối có thể được xử lý đồng thời, do đó hiệu suất cao hơn. Trong khi đó, CCM mode có thời gian xử lý lâu nhất do tính phức tạp của nó. CCM kết hợp giữa CTR và CBC-MAC để đảm bảo độ xác thực, dẫn đến thời gian xử lý dài hơn. Các mode khác có thời gian xử lý khá tương đồng với nhau. Đối với các file nhỏ hơn 1MB, sự chênh lệch thời gian là không đáng kể.

Các biểu đồ cho thấy thời gian mã hóa của một file trên các chế độ mã hóa khác nhau trên Windows và Linux thì Linux mã hóa nhanh hơn Windows đáng kể. Điều này có thể do Linux quản lý tài nguyên và tối ưu hệ thống tốt hơn => Kết quả này cho thấy Linux có hiệu suất mã hóa tốt hơn so với Windows. Việc lựa chọn hệ điều hành và chế độ mã hóa phù hợp là quan trọng để đạt hiệu suất tối ưu trong các ứng dụng yêu cầu mã hóa nhanh chón

Trong từng mode, kích thước input càng lớn thì thời gian mã hóa và giải mã càng lâu.

2.3. Tổng kết:

Sau bài lab này, em đã biết cách sử dụng thư viện CryptoPP, code implement mã hóa và giải mã DES/AES bằng thư viện này, build task, dual boot và có cái nhìn tổng quát về thời gian thực thi của từng loại, nhận thấy sự khác biệt giữa thực thi trên Linux và trên Windows. Thông qua đó sẽ đưa ra được lựa chọn tốt cho các dự án sau này.