

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG

THÁI NGỌC DIỄM TRINH – 22521541

CLASS: NT219.O22.ANTT

OFF-CLASS LABS

LAB 4: ECC-BASED DIGITAL SIGNATURE WITH
CRYPTOPP/OPENSSL

TP. HỒ CHÍ MINH, NĂM 2024

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG

THÁI NGỌC DIỄM TRINH - 22521541

OFF-CLASS LABS
LAB 4: ECC-BASED DIGITAL SIGNATURE WITH
CRYPTOPP/OPENSSL

GIẢNG VIÊN HƯỚNG DẪN
TS. NGUYỄN NGỌC TỰ

TP. HỒ CHÍ MINH, NĂM 2024

MỤC LỤC

Chương 1.	HARDWARE RESOURCE.....	1
Chương 2.	BUILD TASKS.....	2
2.1.	ECDSA.....	2
2.1.1.	Command Line:	2
2.1.2.	GUI.....	4
2.2.	RSAPSS.....	5
Chương 3.	COMPUTATION PERFORMANCE.....	7
3.1.	Sign.....	7
3.2.	Verify.....	8
Chương 4.	COMMENT AND COMPARSION	9
4.1.	ECDSA.....	9
4.2.	RSAPSS.....	9

Chương 1. **HARDWARE RESOURCE**

System Information	
Current Date/Time:	Saturday, June 15, 2024, 12:06:33 PM
Computer Name:	THAITRINH
Operating System:	Windows 11 Home Single Language 64-bit (10.0, Build 22631)
Language:	English (Regional Setting: English)
System Manufacturer:	ASUSTeK COMPUTER INC.
System Model:	ROG Strix G513IE_G513IE
BIOS:	G513IE.329
Processor:	AMD Ryzen 7 4800H with Radeon Graphics (16 CPUs)
Memory:	8192MB RAM
Page file:	11172MB used, 10762MB available
DirectX Version:	DirectX 12

- Ubuntu:

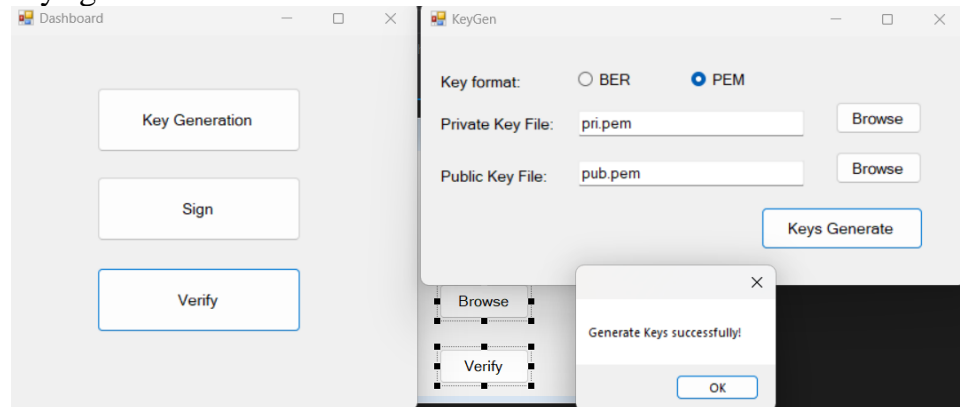
The image shows a C++ IDE with a file named `ECDSA.cpp` open. The code is a C++ program for ECDSA operations. It includes a `main` function that takes command-line arguments. The program can generate keys, sign a message, or verify a signature. The code is as follows:

```
234 int main(int argc, char *argv[])
235 {
236     #endif
237     if (argc < 2)
238     {
239         std::cerr << "Usage: \n"
240         << argv[0] << " genkey <format> <privateKeyFile> <publicKeyFile>\n"
241         << argv[0] << " sign <privateKeyFile> <inputFile> <signFile> \n"
242         << argv[0] << " verify <publicKeyFile> <inputFile> <signFile>\n";
243     }
244     return -1;
245 }
246
247 string mode = argv[1];
248
249 if (mode == "genkey" && argc == 5)
250 {
251     if (GenerateKeys(argv[2], argv[3], argv[4]))
252     {
253         std::cout << "Key generation successfully and saved to " << argv[3] << ", " << argv[4] << std::endl;
254     }
255     else
256     {
257         cout << "Key generation unsuccessfully!" << std::endl;
258         return -1;
259     }
260 }
261
262 else if (mode == "sign" && argc == 5)
263 {
264     if (sign(argv[2], argv[3], argv[4]))
265     {
266         std::cout << "Sign successfully!";
267     }
268     else
269     {
270         cout << "Sign unsuccessfully!";
271         return 1;
272     }
273 }
```

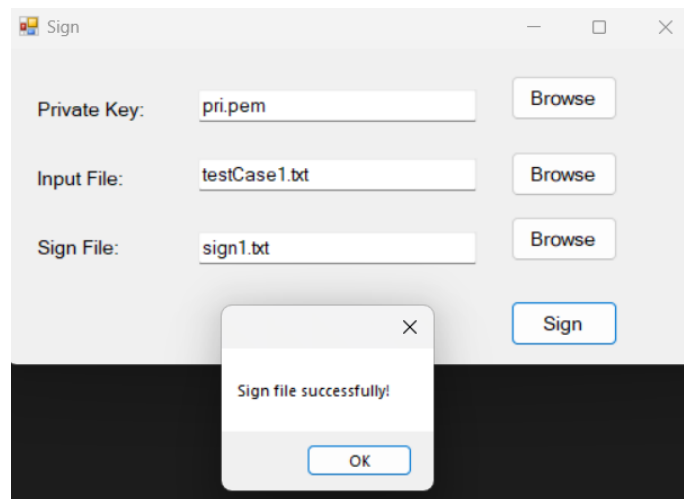
The terminal window shows the execution of the program. The first command is `./ECDSA genkey BER pri.ber pub.ber`, which outputs "Key generation successfully and saved to pri.ber, pub.ber". The second command is `./ECDSA sign pri.ber testCase5.txt sign5.txt`, which outputs "Average time for signing over 1000 rounds: 0.918 ms" and "Sign successfully!". The third command is `./ECDSA verify pub.ber testCase5.txt sign5.txt`, which outputs "Average time for verifying over 1000 rounds: 0.599 ms" and "Verify successfully!". The fourth command is `./ECDSA verify pub.ber testCase5.txt sign5.txt`, which outputs "Verify unsuccessfully!".

2.1.2. GUI

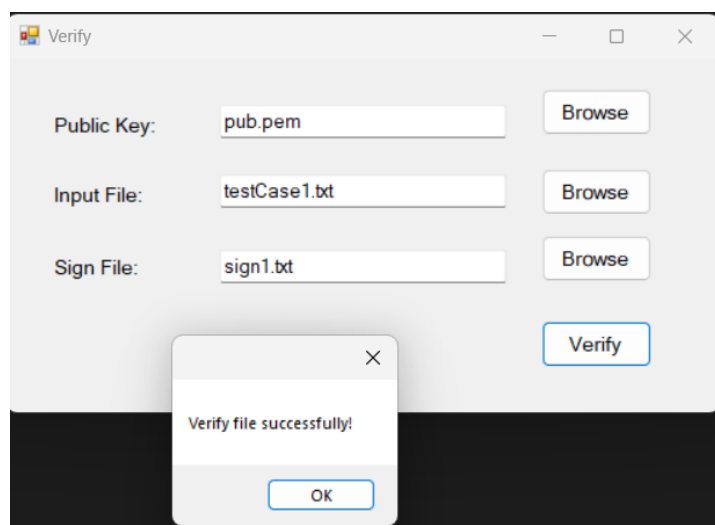
- Keys generation:



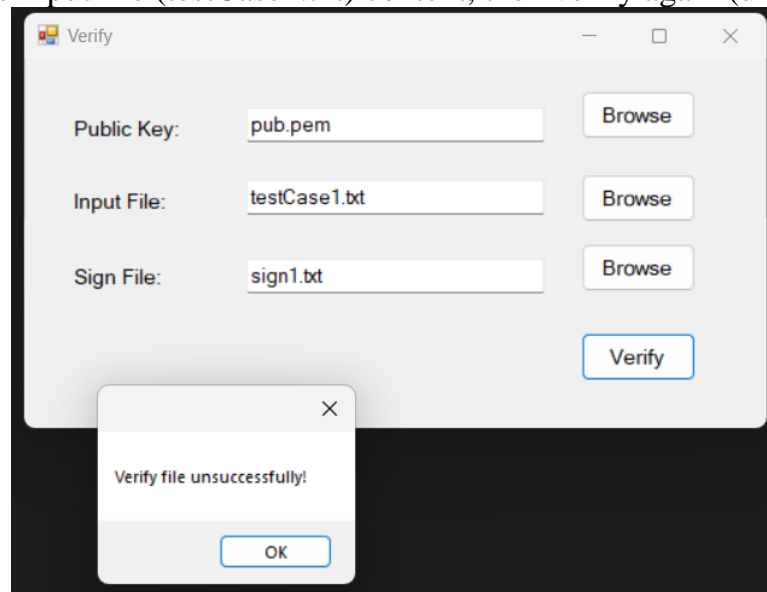
- Sign:



- Verify:



- Change input file (testCase1.txt) content, then verify again (unsuccessfully):



2.2. RSAPSS

- Windows:

```
218 int verify(const char *publicKeyFile, const char *filename, const char *signFile)
219 {
220     return 0;
221 }
222
223 int main(int argc, char *argv[])
224 {
225     if (argc < 2)
226     {
227         std::cerr << "Usage: \n"
228                 << argv[0] << " genkey <format> <privateKeyFile> <publicKeyFile> <keySize>\n"
229                 << argv[0] << " sign <privateKeyFile> <inputFile> <signFile> \n"
230                 << argv[0] << " verify <publicKeyFile> <inputFile> <signFile>\n";
231         return -1;
232     }
233
234     OpenSSL_add_all_algorithms();
235     ERR_load_crypto_strings();
236
237     string mode = argv[1];
238
239     if (mode == "genkey" && argc == 6)
240     {
241         const char *privateKeyFile = argv[2];
242         const char *publicKeyFile = argv[3];
243         const char *format = argv[4];
244         const int keysize = stoi(argv[5]);
245
246         // hool result = GenerateKeys(privateKeyFile, publicKeyFile, format, keysize);
247     }
248 }
```



```

PS C:\Users\thait\Desktop\NT219-Cryptography-UIT\offClassTasks\Task05> ./RSAPSS genkey PEM pri.pem pub.pem 3072
Key generation successful.
PS C:\Users\thait\Desktop\NT219-Cryptography-UIT\offClassTasks\Task05> ./RSAPSS sign pri.pem testCase1.txt sign1.txt
Average time for signing over 1000 rounds: 6.609 ms
Signing successful.
PS C:\Users\thait\Desktop\NT219-Cryptography-UIT\offClassTasks\Task05> ./RSAPSS verify pub.pem testCase1.txt sign1.txt
Average time for verifying over 1000 rounds: 0.183 ms
Signature verification successful.
PS C:\Users\thait\Desktop\NT219-Cryptography-UIT\offClassTasks\Task05> ./RSAPSS verify pub.pem testCase1.txt sign1.txt
Average time for verifying over 1000 rounds: 0.169 ms

```

- Ubuntu:

The screenshot shows a code editor with the file `RSAPSS.cpp` open. The code is a C++ program that implements RSA operations. It includes a `main` function that takes command-line arguments. The program can generate keys, sign files, and verify signatures. It uses the OpenSSL library for cryptographic operations. The terminal output at the bottom shows the program being executed with various flags and arguments, resulting in successful key generation and signing.

```

330 }
331
332 int main(int argc, char *argv[])
333 {
334     if (argc < 2)
335     {
336         std::cerr << "Usage: \n"
337             << argv[0] << " genkey <format> <privateKeyFile> <publicKeyFile> <keySize>\n"
338             << argv[0] << " sign <privateKeyFile> <inputFile> <signFile> \n"
339             << argv[0] << " verify <publicKeyFile> <inputFile> <signFile>\n";
340         return -1;
341     }
342
343     OpenSSL_add_all_algorithms();
344     ERR_load_crypto_strings();
345
346     string mode = argv[1];
347
348     if (mode == "genkey" && argc == 6)
349     {
350         const char *privateKeyFile = argv[2];
351         const char *publicKeyFile = argv[3];
352         const char *format = argv[4];
353         const int keysize = stoi(argv[5]);
354
355         // bool result = GenerateKeys(privateKeyFile, publicKeyFile, format, keysize);
356         bool result = GenerateKeys(argv[2], argv[3], argv[4], stoi(argv[5]));
357         if (GenerateKeys(argv[2], argv[3], argv[4], stoi(argv[5])))
358         {
359             cout << "Key generation successful." << endl;
360         }
361         else
362         {
363             cerr << "Key generation failed." << endl;
364             return -1;
365         }
366     }
367
368     PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
369
370 Executing task: /usr/bin/g++ -g2 -O3 -DDEBUG /home/thaitrinh/Desktop/labTask/ECDSA/RSAPSS.cpp -o /home/thaitrinh/Desktop/labTask/ECDSA/RSAPSS -pthread -I/home/thaitrinh/Desktop/labTask/ECDSA/include -L/home/thaitrinh/Desktop/labTask/ECDSA/lib -l:libcrypto.a -l:libssl.a -l:libbctop.a -Wall -w
371
372 Terminal will be reused by tasks, press any key to close it.

```

```

thaitrinh@thaitrinh:~/Desktop/labTask/ECDSA$ ./RSAPSS genkey BER pri.ber pub.ber 3072
Key generation successful.
thaitrinh@thaitrinh:~/Desktop/labTask/ECDSA$ ./RSAPSS sign pri.ber testCase1.txt sign1.txt
Average time for signing over 1000 rounds: 2.696 ms
Sign file successfully!
thaitrinh@thaitrinh:~/Desktop/labTask/ECDSA$ ./RSAPSS verify pub.ber testCase1.txt sign1.txt
Average time for verifying over 1000 rounds: 0.075 ms
Verify file successfully!
thaitrinh@thaitrinh:~/Desktop/labTask/ECDSA$ ./RSAPSS verify pub.ber testCase1.txt sign1.txt
Average time for verifying over 1000 rounds: 0.077 ms
Verify file unsuccessfully! 8
thaitrinh@thaitrinh:~/Desktop/labTask/ECDSA$

```

Chương 3. COMPUTATION PERFORMANCE

Number of iterations: 1000

Time counter: mili seconds

3.1. Sign

	ECDSA		RSAPSS	
Test case	Windows	Linux	Windows	Linux
1 (1KB)	6.803	1.109	6.655	2.37
2 (20KB)	7.458	0.906	6.654	2.376
3 (50KB)	7.63	1.173	6.766	2.46
4 (100KB)	7.944	0.924	6.876	2.755
5 (200KB)	8.195	0.976	6.934	2.937
6 (1MB)	7.832	1.007	7.524	10.257
7 (5MB)	7.844	1.079	17.881	26.167

3.2. Verify

	ECDSA		RSAPSS	
Test case	Windows	Linux	Windows	Linux
1 (1KB)	0.635	0.615	0.197	0.076
2 (20KB)	0.635	0.611	0.242	0.11
3 (50KB)	0.63	0.608	0.306	0.185
4 (100KB)	0.643	0.612	0.418	0.323
5 (200KB)	0.658	0.616	0.636	0.594
6 (1MB)	0.682	0.614	3.166	3.942
7 (5MB)	0.658	0.606	13.982	17.371

Chương 4. COMMENT AND COMPARSION

4.1. ECDSA

For sign algorith, Ubuntu has much superior performance compared to Windows, nearly 6 times faster.

For verify algorith, the difference between Linux and Windows is not large, but Linux is still slightly faster than Windows in all test cases.

With files of different sizes, the time to sign and verify is not significantly different.

4.2. RSAPSS

For both sign and verify algorithh, Ubuntu has a shorter average run time than Windows.

The larger the file, the slower the running time.

4.3. Conclusion

For files smaller than 200KB in size, running time on RSAPSS can be faster or equal to that on ECDSA.

But for larger files, RSAPSS runs much slower, if the file size is larger, it is slower, while ECDSA has no difference in running time with files of different sizes.