

Cybersecurity - Homework 2

Vlad Turno (1835365)

October 15, 2025

1) Introduction

In cryptography, a block cipher is an algorithm that operates iterating over blocks of bits having a fixed length. In this homework there will be a performance analysis over three different CBC algorithms (AES, Camellia, and Sm4) all using a symmetric 128-bit key to encrypt and decrypt some files.

2) Performance Analysis

With a simple but quite not short C program it is possible to create some files having a specified dimension (16 bytes, 20KB and 2MB) alongside with a random generated 128-bit key and by using some functions from the OpenSSL library we can encode and decode them using the cpu clock to calculate the operation time in milliseconds.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <openssl/rand.h>
4  #include <openssl/aes.h>
5  #include <openssl/evp.h>
6  #include <openssl/camellia.h>
7  #include <time.h>
8
9  int main(){
10
11     char line[]="\n";
12     printf("%s\n", line);
13
14     /*generate a 128-bit (16-bytes) random key*/
15     unsigned char key[16];
16     RAND_bytes(key, sizeof(key));
17     printf("128-bit key:");
18     for(int i=0; i<sizeof(key); i++)
19         printf("%02x", key[i]);
20     printf("\n");
21     printf("%s\n", line);
22
23     /*generate files to work with*/
24     FILE *file;
25     unsigned char buffer_small[8];
26     unsigned char buffer_medium[10240];
27     unsigned char buffer_large[1024000];
28
29     RAND_bytes(buffer_small, 8);
30     file = fopen("random_small.txt", "w");
31     if(file==NULL){
32         printf("Error opening txt file\n");
33         return 1;
34     }
35     for(int i=0; i<8; i++)
36         fprintf(file, "%02x", buffer_small[i]);
37     printf("Wrote 16 bytes of random data to random_small.txt\n");
38     fclose(file);
39
40     RAND_bytes(buffer_medium, 10240);
41     file = fopen("random_medium.txt", "w");
42     if(file==NULL){
```

```

43     printf("Error opening txt file\n");
44     return 1;
45 }
46 for(int i=0; i<10240; i++)
47     fprintf(file, "%02x", buffer_medium[i]);
48 printf("Wrote 20KB of random data to random_medium.txt\n");
49 fclose(file);
50
51 RAND_bytes(buffer_large, 1024000);
52 file = fopen("random_large.bin", "wb");
53 if(file==NULL){
54     printf("Error opening bin file\n");
55     return 1;
56 }
57 for(int i=0; i<1024000; i++)
58     fprintf(file, "%02x", buffer_large[i]);
59 printf("Wrote 2MB of random data to random_large.bin\n");
60 fclose(file);
61
62 printf("%s\n", line);
63
64 /*encryption using AES CBC mode*/
65 FILE *source_file, *destination_file;
66 unsigned char iv[AES_BLOCK_SIZE];
67 unsigned char input_buffer[1024], output_buffer[1024 + EVP_MAX_BLOCK_LENGTH];
68 int input_length, output_length;
69
70 clock_t start, end;
71 int cpu_usage;
72
73 RAND_bytes(iv, sizeof(iv));
74
75 source_file = fopen("random_small.txt", "rb");
76 destination_file = fopen("aes_small_enc.bin", "wb");
77
78 EVP_CIPHER_CTX *ctx = EVP_CIPHER_CTX_new();
79 EVP_EncryptInit_ex(ctx, EVP_aes_128_cbc(), NULL, key, iv);
80
81 start = clock();
82 while((input_length = fread(input_buffer, 1, sizeof(input_buffer), source_file))
83        >0){
84     EVP_EncryptUpdate(ctx, output_buffer, &output_length, input_buffer,
85                       input_length);
86     fwrite(output_buffer, 1, output_length, destination_file);
87 }
88 EVP_EncryptFinal_ex(ctx, output_buffer, &output_length);
89 fwrite(output_buffer, 1, output_length, destination_file);
90 end = clock();
91
92 EVP_CIPHER_CTX_free(ctx);
93 fclose(source_file);
94 fclose(destination_file);
95
96 cpu_usage = end - start;
97 printf("Encryption of 16 byte file completed in %d ms using AES-128-CBC\n",
98        cpu_usage);
99
100 source_file = fopen("random_medium.txt", "rb");
101 destination_file = fopen("aes_medium_enc.bin", "wb");
102
103 ctx = EVP_CIPHER_CTX_new();
104 EVP_EncryptInit_ex(ctx, EVP_aes_128_cbc(), NULL, key, iv);
105
106 start = clock();
107 while((input_length = fread(input_buffer, 1, sizeof(input_buffer), source_file))
108        >0){
109     EVP_EncryptUpdate(ctx, output_buffer, &output_length, input_buffer,
110                       input_length);
111     fwrite(output_buffer, 1, output_length, destination_file);
112 }
113 EVP_EncryptFinal_ex(ctx, output_buffer, &output_length);
114 fwrite(output_buffer, 1, output_length, destination_file);
115 end = clock();
116
117 EVP_CIPHER_CTX_free(ctx);
118 fclose(source_file);

```

```

114     fclose(destination_file);
115
116     cpu_usage = end - start;
117     printf("Encryption of 20KB file completed in %d ms using AES-128-CBC\n",
118           cpu_usage);
119
120     source_file = fopen("random_large.bin", "rb");
121     destination_file = fopen("aes_large_enc.bin", "wb");
122
123     ctx = EVP_CIPHER_CTX_new();
124     EVP_EncryptInit_ex(ctx, EVP_aes_128_cbc(), NULL, key, iv);
125
126     start = clock();
127     while((input_length = fread(input_buffer, 1, sizeof(input_buffer), source_file))
128           > 0){
129         EVP_EncryptUpdate(ctx, output_buffer, &output_length, input_buffer,
130                           input_length);
131         fwrite(output_buffer, 1, output_length, destination_file);
132     }
133     EVP_EncryptFinal_ex(ctx, output_buffer, &output_length);
134     fwrite(output_buffer, 1, output_length, destination_file);
135     end = clock();
136
137     EVP_CIPHER_CTX_free(ctx);
138     fclose(source_file);
139     fclose(destination_file);
140
141     cpu_usage = end - start;
142     printf("Encryption of 2MB file completed in %d ms using AES-128-CBC\n", cpu_usage);
143     printf("%s\n", line);
144
145     /*decryption using AES CBC mode*/
146     source_file = fopen("aes_small_enc.bin", "rb");
147     destination_file = fopen("aes_small_dec.bin", "wb");
148
149     fread(iv, 1, AES_BLOCK_SIZE, source_file);
150     ctx = EVP_CIPHER_CTX_new();
151     EVP_DecryptInit_ex(ctx, EVP_aes_128_cbc(), NULL, key, iv);
152
153     start = clock();
154     while((input_length = fread(input_buffer, 1, sizeof(input_buffer), source_file))
155           > 0){
156         EVP_DecryptUpdate(ctx, output_buffer, &output_length, input_buffer,
157                           input_length);
158         fwrite(output_buffer, 1, output_length, destination_file);
159     }
160     EVP_DecryptFinal_ex(ctx, output_buffer, &output_length);
161     fwrite(output_buffer, 1, output_length, destination_file);
162     end = clock();
163
164     EVP_CIPHER_CTX_free(ctx);
165     fclose(source_file);
166     fclose(destination_file);
167
168     cpu_usage = end - start;
169     printf("Decryption of 16 byte file completed in %d ms using AES-128-CBC\n",
170           cpu_usage);
171
172     source_file = fopen("aes_medium_enc.bin", "rb");
173     destination_file = fopen("aes_medium_dec.bin", "wb");
174
175     fread(iv, 1, AES_BLOCK_SIZE, source_file);
176     ctx = EVP_CIPHER_CTX_new();
177     EVP_DecryptInit_ex(ctx, EVP_aes_128_cbc(), NULL, key, iv);
178
179     start = clock();
180     while((input_length = fread(input_buffer, 1, sizeof(input_buffer), source_file))
181           > 0){
182         EVP_DecryptUpdate(ctx, output_buffer, &output_length, input_buffer,
183                           input_length);
184         fwrite(output_buffer, 1, output_length, destination_file);
185     }
186     EVP_DecryptFinal_ex(ctx, output_buffer, &output_length);
187     fwrite(output_buffer, 1, output_length, destination_file);
188     end = clock();

```

```

181 EVP_CIPHER_CTX_free(ctx);
182 fclose(source_file);
183 fclose(destination_file);
184
185
186 cpu_usage = end - start;
187 printf("Decryption of 20KB file completed in %d ms using AES-128-CBC\n",
188        cpu_usage);
189
190 source_file = fopen("aes_large_enc.bin", "rb");
191 destination_file = fopen("aes_large_dec.bin", "wb");
192
193 fread(iv, 1, AES_BLOCK_SIZE, source_file);
194 ctx = EVP_CIPHER_CTX_new();
195 EVP_DecryptInit_ex(ctx, EVP_aes_128_cbc(), NULL, key, iv);
196
197 start = clock();
198 while((input_length = fread(input_buffer, 1, sizeof(input_buffer), source_file))
199       > 0){
200     EVP_DecryptUpdate(ctx, output_buffer, &output_length, input_buffer,
201                      input_length);
202     fwrite(output_buffer, 1, output_length, destination_file);
203 }
204 EVP_DecryptFinal_ex(ctx, output_buffer, &output_length);
205 fwrite(output_buffer, 1, output_length, destination_file);
206 end = clock();
207
208 EVP_CIPHER_CTX_free(ctx);
209 fclose(source_file);
210 fclose(destination_file);
211
212 cpu_usage = end - start;
213 printf("Decryption of 2MB file completed in %d ms using AES-128-CBC\n", cpu_usage
214        );
215 printf("%s\n", line);
216
217 /*encryption using CAMELLIA CBC mode*/
218 unsigned char ivc[CAMELLIA_BLOCK_SIZE];
219
220 source_file = fopen("random_small.txt", "rb");
221 destination_file = fopen("camellia_small_enc.bin", "wb");
222
223 fwrite(ivc, 1, CAMELLIA_BLOCK_SIZE, destination_file);
224
225 ctx = EVP_CIPHER_CTX_new();
226 EVP_EncryptInit_ex(ctx, EVP_camellia_128_cbc(), NULL, key, ivc);
227
228 start = clock();
229 while((input_length = fread(input_buffer, 1, sizeof(input_buffer), source_file))
230       > 0){
231     EVP_EncryptUpdate(ctx, output_buffer, &output_length, input_buffer,
232                      input_length);
233     fwrite(output_buffer, 1, output_length, destination_file);
234 }
235 EVP_EncryptFinal_ex(ctx, output_buffer, &output_length);
236 fwrite(output_buffer, 1, output_length, destination_file);
237 end = clock();
238
239 EVP_CIPHER_CTX_free(ctx);
240 fclose(source_file);
241 fclose(destination_file);
242
243 cpu_usage = end - start;
244 printf("Encryption of 16 byte file completed in %d ms using CAMELLIA-128-CBC\n",
245        cpu_usage);
246
247 source_file = fopen("random_medium.txt", "rb");
248 destination_file = fopen("camellia_medium_enc.bin", "wb");
249
250 fwrite(ivc, 1, CAMELLIA_BLOCK_SIZE, destination_file);
251
252 ctx = EVP_CIPHER_CTX_new();
253 EVP_EncryptInit_ex(ctx, EVP_camellia_128_cbc(), NULL, key, ivc);
254
255 start = clock();
256 while((input_length = fread(input_buffer, 1, sizeof(input_buffer), source_file))

```

```

>0){
250     EVP_EncryptUpdate(ctx, output_buffer, &output_length, input_buffer,
        input_length);
251     fwrite(output_buffer, 1, output_length, destination_file);
252 }
253 EVP_EncryptFinal_ex(ctx, output_buffer, &output_length);
254 fwrite(output_buffer, 1, output_length, destination_file);
255 end = clock();
256
257 EVP_CIPHER_CTX_free(ctx);
258 fclose(source_file);
259 fclose(destination_file);
260
261 cpu_usage = end - start;
262 printf("Encryption of 20KB file completed in %d ms using CAMELLIA-128-CBC\n",
        cpu_usage);
263
264 source_file = fopen("random_large.bin", "rb");
265 destination_file = fopen("camellia_large_enc.bin", "wb");
266
267 fwrite(ivc, 1, CAMELLIA_BLOCK_SIZE, destination_file);
268
269 ctx = EVP_CIPHER_CTX_new();
270 EVP_EncryptInit_ex(ctx, EVP_camellia_128_cbc(), NULL, key, ivc);
271
272 start = clock();
273 while((input_length = fread(input_buffer, 1, sizeof(input_buffer), source_file))
        >0){
274     EVP_EncryptUpdate(ctx, output_buffer, &output_length, input_buffer,
        input_length);
275     fwrite(output_buffer, 1, output_length, destination_file);
276 }
277 EVP_EncryptFinal_ex(ctx, output_buffer, &output_length);
278 fwrite(output_buffer, 1, output_length, destination_file);
279 end = clock();
280
281 EVP_CIPHER_CTX_free(ctx);
282 fclose(source_file);
283 fclose(destination_file);
284
285 cpu_usage = end - start;
286 printf("Encryption of 2MB file completed in %d ms using CAMELLIA-128-CBC\n",
        cpu_usage);
287 printf("%s\n", line);
288
289 /*decryption using CAMELLIA CBC mode*/
290 source_file = fopen("camellia_small_enc.bin", "rb");
291 destination_file = fopen("camellia_small_dec.bin", "wb");
292
293 fread(ivc, 1, CAMELLIA_BLOCK_SIZE, source_file);
294 ctx = EVP_CIPHER_CTX_new();
295 EVP_DecryptInit_ex(ctx, EVP_camellia_128_cbc(), NULL, key, iv);
296
297 start = clock();
298 while((input_length = fread(input_buffer, 1, sizeof(input_buffer), source_file))
        >0){
299     EVP_DecryptUpdate(ctx, output_buffer, &output_length, input_buffer,
        input_length);
300     fwrite(output_buffer, 1, output_length, destination_file);
301 }
302 EVP_DecryptFinal_ex(ctx, output_buffer, &output_length);
303 fwrite(output_buffer, 1, output_length, destination_file);
304 end = clock();
305
306 EVP_CIPHER_CTX_free(ctx);
307 fclose(source_file);
308 fclose(destination_file);
309
310 cpu_usage = end - start;
311 printf("Decryption of 16 byte file completed in %d ms using CAMELLIA-128-CBC\n",
        cpu_usage);
312
313 source_file = fopen("camellia_medium_enc.bin", "rb");
314 destination_file = fopen("camellia_medium_dec.bin", "wb");
315
316 fread(ivc, 1, CAMELLIA_BLOCK_SIZE, source_file);

```

```

317     ctx = EVP_CIPHER_CTX_new();
318     EVP_DecryptInit_ex(ctx, EVP_camellia_128_cbc(), NULL, key, iv);
319
320     start = clock();
321     while((input_length = fread(input_buffer, 1, sizeof(input_buffer), source_file))
322           > 0){
323         EVP_DecryptUpdate(ctx, output_buffer, &output_length, input_buffer,
324                           input_length);
325         fwrite(output_buffer, 1, output_length, destination_file);
326     }
327     EVP_DecryptFinal_ex(ctx, output_buffer, &output_length);
328     fwrite(output_buffer, 1, output_length, destination_file);
329     end = clock();
330
331     EVP_CIPHER_CTX_free(ctx);
332     fclose(source_file);
333     fclose(destination_file);
334
335     cpu_usage = end - start;
336     printf("Decryption of 20KB file completed in %d ms using CAMELLIA-128-CBC\n",
337           cpu_usage);
338
339     source_file = fopen("camellia_large_enc.bin", "rb");
340     destination_file = fopen("camellia_large_dec.bin", "wb");
341
342     fread(ivc, 1, CAMELLIA_BLOCK_SIZE, source_file);
343     ctx = EVP_CIPHER_CTX_new();
344     EVP_DecryptInit_ex(ctx, EVP_camellia_128_cbc(), NULL, key, iv);
345
346     start = clock();
347     while((input_length = fread(input_buffer, 1, sizeof(input_buffer), source_file))
348           > 0){
349         EVP_DecryptUpdate(ctx, output_buffer, &output_length, input_buffer,
350                           input_length);
351         fwrite(output_buffer, 1, output_length, destination_file);
352     }
353     EVP_DecryptFinal_ex(ctx, output_buffer, &output_length);
354     fwrite(output_buffer, 1, output_length, destination_file);
355     end = clock();
356
357     EVP_CIPHER_CTX_free(ctx);
358     fclose(source_file);
359     fclose(destination_file);
360
361     cpu_usage = end - start;
362     printf("Decryption of 2MB file completed in %d ms using CAMELLIA-128-CBC\n",
363           cpu_usage);
364     printf("%s\n", line);
365
366     /*encryption using SM4 CBC mode*/
367     source_file = fopen("random_small.txt", "rb");
368     destination_file = fopen("sm4_small_enc.bin", "wb");
369
370     ctx = EVP_CIPHER_CTX_new();
371     EVP_EncryptInit_ex(ctx, EVP_sm4_cbc(), NULL, key, iv);
372
373     start = clock();
374     while((input_length = fread(input_buffer, 1, sizeof(input_buffer), source_file))
375           > 0){
376         EVP_EncryptUpdate(ctx, output_buffer, &output_length, input_buffer,
377                           input_length);
378         fwrite(output_buffer, 1, output_length, destination_file);
379     }
380     EVP_EncryptFinal_ex(ctx, output_buffer, &output_length);
381     fwrite(output_buffer, 1, output_length, destination_file);
382     end = clock();
383
384     EVP_CIPHER_CTX_free(ctx);
385     fclose(source_file);
386     fclose(destination_file);
387
388     cpu_usage = end - start;
389     printf("Encryption of 16 byte file completed in %d ms using SM4-128-CBC\n",
390           cpu_usage);
391
392     source_file = fopen("random_medium.txt", "rb");

```

```

384 destination_file = fopen("sm4_medium_enc.bin", "wb");
385
386 ctx = EVP_CIPHER_CTX_new();
387 EVP_EncryptInit_ex(ctx, EVP_sm4_cbc(), NULL, key, iv);
388
389 start = clock();
390 while((input_length = fread(input_buffer, 1, sizeof(input_buffer), source_file))
391 > 0){
392     EVP_EncryptUpdate(ctx, output_buffer, &output_length, input_buffer,
393         input_length);
394     fwrite(output_buffer, 1, output_length, destination_file);
395 }
396 EVP_EncryptFinal_ex(ctx, output_buffer, &output_length);
397 fwrite(output_buffer, 1, output_length, destination_file);
398 end = clock();
399
400 EVP_CIPHER_CTX_free(ctx);
401 fclose(source_file);
402 fclose(destination_file);
403
404 cpu_usage = end - start;
405 printf("Encryption of 20KB file completed in %d ms using SM4-128-CBC\n",
406     cpu_usage);
407
408 source_file = fopen("random_large.bin", "rb");
409 destination_file = fopen("sm4_large_enc.bin", "wb");
410
411 ctx = EVP_CIPHER_CTX_new();
412 EVP_EncryptInit_ex(ctx, EVP_sm4_cbc(), NULL, key, iv);
413
414 start = clock();
415 while((input_length = fread(input_buffer, 1, sizeof(input_buffer), source_file))
416 > 0){
417     EVP_EncryptUpdate(ctx, output_buffer, &output_length, input_buffer,
418         input_length);
419     fwrite(output_buffer, 1, output_length, destination_file);
420 }
421 EVP_EncryptFinal_ex(ctx, output_buffer, &output_length);
422 fwrite(output_buffer, 1, output_length, destination_file);
423 end = clock();
424
425 EVP_CIPHER_CTX_free(ctx);
426 fclose(source_file);
427 fclose(destination_file);
428
429 cpu_usage = end - start;
430 printf("Encryption of 2MB file completed in %d ms using SM4-128-CBC\n", cpu_usage
431 );
432 printf("%s\n", line);
433
434 /*decryption using SM4 CBC mode*/
435 source_file = fopen("sm4_small_enc.bin", "rb");
436 destination_file = fopen("sm4_small_dec.bin", "wb");
437
438 fread(iv, 1, AES_BLOCK_SIZE, source_file);
439 ctx = EVP_CIPHER_CTX_new();
440 EVP_DecryptInit_ex(ctx, EVP_sm4_cbc(), NULL, key, iv);
441
442 start = clock();
443 while((input_length = fread(input_buffer, 1, sizeof(input_buffer), source_file))
444 > 0){
445     EVP_DecryptUpdate(ctx, output_buffer, &output_length, input_buffer,
446         input_length);
447     fwrite(output_buffer, 1, output_length, destination_file);
448 }
449 EVP_DecryptFinal_ex(ctx, output_buffer, &output_length);
450 fwrite(output_buffer, 1, output_length, destination_file);
451 end = clock();
452
453 EVP_CIPHER_CTX_free(ctx);
454 fclose(source_file);
455 fclose(destination_file);
456
457 cpu_usage = end - start;
458 printf("Decryption of 16 byte file completed in %d ms using SM4-128-CBC\n",
459     cpu_usage);

```

```

451 source_file = fopen("sm4_medium_enc.bin", "rb");
452 destination_file = fopen("sm4_medium_dec.bin", "wb");
453
454
455 fread(iv, 1, AES_BLOCK_SIZE, source_file);
456 ctx = EVP_CIPHER_CTX_new();
457 EVP_DecryptInit_ex(ctx, EVP_sm4_cbc(), NULL, key, iv);
458
459 start = clock();
460 while((input_length = fread(input_buffer, 1, sizeof(input_buffer), source_file))
461        > 0){
462     EVP_DecryptUpdate(ctx, output_buffer, &output_length, input_buffer,
463                       input_length);
464     fwrite(output_buffer, 1, output_length, destination_file);
465 }
466 EVP_DecryptFinal_ex(ctx, output_buffer, &output_length);
467 fwrite(output_buffer, 1, output_length, destination_file);
468 end = clock();
469
470 EVP_CIPHER_CTX_free(ctx);
471 fclose(source_file);
472 fclose(destination_file);
473
474 cpu_usage = end - start;
475 printf("Decryption of 20KB file completed in %d ms using SM4-128-CBC\n",
476        cpu_usage);
477
478 source_file = fopen("sm4_large_enc.bin", "rb");
479 destination_file = fopen("sm4_large_dec.bin", "wb");
480
481
482 fread(iv, 1, AES_BLOCK_SIZE, source_file);
483 ctx = EVP_CIPHER_CTX_new();
484 EVP_DecryptInit_ex(ctx, EVP_sm4_cbc(), NULL, key, iv);
485
486 start = clock();
487 while((input_length = fread(input_buffer, 1, sizeof(input_buffer), source_file))
488        > 0){
489     EVP_DecryptUpdate(ctx, output_buffer, &output_length, input_buffer,
490                       input_length);
491     fwrite(output_buffer, 1, output_length, destination_file);
492 }
493 EVP_DecryptFinal_ex(ctx, output_buffer, &output_length);
494 fwrite(output_buffer, 1, output_length, destination_file);
495 end = clock();
496
497 EVP_CIPHER_CTX_free(ctx);
498 fclose(source_file);
499 fclose(destination_file);
500
501 cpu_usage = end - start;
502 printf("Decryption of 2MB file completed in %d ms using SM4-128-CBC\n", cpu_usage
503        );
504 printf("%s\n", line);
505
506 return 0;
507 }

```

Listing 1: comparator.c

3) Observations

Looking at the program output, it is easy to note how all the algorithms perform similarly over small files (16 bytes to 20kb), but how things change drastically when they work with a slightly bigger file. Now we take the results from our C program and plot them onto a chart to better visualize the results.

```
128-bit key:57bbbb223211c3c9f3866a8813654581

Wrote 16 bytes of random data to random_small.txt
Wrote 20KB of random data to random_medium.txt
Wrote 2MB of random data to random_large.bin

Encryption of 16 byte file completed in 29 ms using AES-128-CBC
Encryption of 20KB file completed in 121 ms using AES-128-CBC
Encryption of 2MB file completed in 10412 ms using AES-128-CBC

Decryption of 16 byte file completed in 8 ms using AES-128-CBC
Decryption of 20KB file completed in 93 ms using AES-128-CBC
Decryption of 2MB file completed in 8912 ms using AES-128-CBC

Encryption of 16 byte file completed in 21 ms using CAMELLIA-128-CBC
Encryption of 20KB file completed in 236 ms using CAMELLIA-128-CBC
Encryption of 2MB file completed in 22569 ms using CAMELLIA-128-CBC

Decryption of 16 byte file completed in 7 ms using CAMELLIA-128-CBC
Decryption of 20KB file completed in 206 ms using CAMELLIA-128-CBC
Decryption of 2MB file completed in 23611 ms using CAMELLIA-128-CBC

Encryption of 16 byte file completed in 30 ms using SM4-128-CBC
Encryption of 20KB file completed in 289 ms using SM4-128-CBC
Encryption of 2MB file completed in 30255 ms using SM4-128-CBC

Decryption of 16 byte file completed in 8 ms using SM4-128-CBC
Decryption of 20KB file completed in 303 ms using SM4-128-CBC
Decryption of 2MB file completed in 30510 ms using SM4-128-CBC
```

