

# 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)) >0){
83     EVP_EncryptUpdate(ctx, output_buffer, &output_length, input_buffer, input_length);
84     fwrite(output_buffer, 1, output_length, destination_file);
85 }
86 EVP_EncryptFinal_ex(ctx, output_buffer, &output_length);
87 fwrite(output_buffer, 1, output_length, destination_file);
88 end = clock();
89
90 EVP_CIPHER_CTX_free(ctx);
91 fclose(source_file);
92 fclose(destination_file);
93
94 cpu_usage = end - start;
95 printf("Encryption of 16 byte file completed in %d ms using AES-128-CBC\n",
96       cpu_usage);
97
98 source_file = fopen("random_medium.txt", "rb");
99 destination_file = fopen("aes_medium_enc.bin", "wb");
100
101 ctx = EVP_CIPHER_CTX_new();
102 EVP_EncryptInit_ex(ctx, EVP_aes_128_cbc(), NULL, key, iv);
103
104 start = clock();
105 while((input_length = fread(input_buffer, 1, sizeof(input_buffer), source_file)) >0){
106     EVP_EncryptUpdate(ctx, output_buffer, &output_length, input_buffer, input_length);
107     fwrite(output_buffer, 1, output_length, destination_file);
108 }
109 EVP_EncryptFinal_ex(ctx, output_buffer, &output_length);
110 fwrite(output_buffer, 1, output_length, destination_file);
111 end = clock();
112
113 EVP_CIPHER_CTX_free(ctx);
114 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)) >0){
128     EVP_EncryptUpdate(ctx, output_buffer, &output_length, input_buffer,
129         input_length);
130     fwrite(output_buffer, 1, output_length, destination_file);
131 }
132 EVP_EncryptFinal_ex(ctx, output_buffer, &output_length);
133 fwrite(output_buffer, 1, output_length, destination_file);
134 end = clock();
135
136 EVP_CIPHER_CTX_free(ctx);
137 fclose(source_file);
138 fclose(destination_file);
139
140 cpu_usage = end - start;
141 printf("Encryption of 2MB file completed in %d ms using AES-128-CBC\n", cpu_usage);
142 printf("%s\n", line);
143
/*decryption using AES CBC mode*/
144 source_file = fopen("aes_small_enc.bin", "rb");
145 destination_file = fopen("aes_small_dec.bin", "wb");
146
147 fread(iv, 1, AES_BLOCK_SIZE, source_file);
148 ctx = EVP_CIPHER_CTX_new();
149 EVP_DecryptInit_ex(ctx, EVP_aes_128_cbc(), NULL, key, iv);
150
151 start = clock();
152 while((input_length = fread(input_buffer, 1, sizeof(input_buffer), source_file)) >0){
153     EVP_DecryptUpdate(ctx, output_buffer, &output_length, input_buffer,
154         input_length);
155     fwrite(output_buffer, 1, output_length, destination_file);
156 }
157 EVP_DecryptFinal_ex(ctx, output_buffer, &output_length);
158 fwrite(output_buffer, 1, output_length, destination_file);
159 end = clock();
160
161 EVP_CIPHER_CTX_free(ctx);
162 fclose(source_file);
163 fclose(destination_file);
164
165 cpu_usage = end - start;
166 printf("Decryption of 16 byte file completed in %d ms using AES-128-CBC\n",
167     cpu_usage);
168
169 source_file = fopen("aes_medium_enc.bin", "rb");
170 destination_file = fopen("aes_medium_dec.bin", "wb");
171
172 fread(iv, 1, AES_BLOCK_SIZE, source_file);
173 ctx = EVP_CIPHER_CTX_new();
174 EVP_DecryptInit_ex(ctx, EVP_aes_128_cbc(), NULL, key, iv);
175
176 start = clock();
177 while((input_length = fread(input_buffer, 1, sizeof(input_buffer), source_file)) >0){
178     EVP_DecryptUpdate(ctx, output_buffer, &output_length, input_buffer,
179         input_length);
180     fwrite(output_buffer, 1, output_length, destination_file);
181 }
182 EVP_DecryptFinal_ex(ctx, output_buffer, &output_length);
183 fwrite(output_buffer, 1, output_length, destination_file);
184 end = clock();

```

```

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

```

    >0){
250     EVP_EncryptUpdate(ctx, output_buffer, &output_length, input_buffer,
251                     input_length);
252     fwrite(output_buffer, 1, output_length, destination_file);
253 }
254 EVP_EncryptFinal_ex(ctx, output_buffer, &output_length);
255 fwrite(output_buffer, 1, output_length, destination_file);
256 end = clock();
257
258 EVP_CIPHER_CTX_free(ctx);
259 fclose(source_file);
260 fclose(destination_file);
261
262 cpu_usage = end - start;
263 printf("Encryption of 20KB file completed in %d ms using CAMELLIA-128-CBC\n",
264        cpu_usage);
265
266 source_file = fopen("random_large.bin", "rb");
267 destination_file = fopen("camellia_large_enc.bin", "wb");
268
269 fwrite(ivc, 1, CAMELLIA_BLOCK_SIZE, destination_file);
270
271 ctx = EVP_CIPHER_CTX_new();
272 EVP_EncryptInit_ex(ctx, EVP_camellia_128_cbc(), NULL, key, ivc);
273
274 start = clock();
275 while((input_length = fread(input_buffer, 1, sizeof(input_buffer), source_file)) >0){
276     EVP_EncryptUpdate(ctx, output_buffer, &output_length, input_buffer,
277                     input_length);
278     fwrite(output_buffer, 1, output_length, destination_file);
279 }
280 EVP_EncryptFinal_ex(ctx, output_buffer, &output_length);
281 fwrite(output_buffer, 1, output_length, destination_file);
282 end = clock();
283
284 EVP_CIPHER_CTX_free(ctx);
285 fclose(source_file);
286 fclose(destination_file);
287
288 cpu_usage = end - start;
289 printf("Encryption of 2MB file completed in %d ms using CAMELLIA-128-CBC\n",
290        cpu_usage);
291 printf("%s\n", line);
292
293 /*decryption using CAMELLIA CBC mode*/
294 source_file = fopen("camellia_small_enc.bin", "rb");
295 destination_file = fopen("camellia_small_dec.bin", "wb");
296
297 fread(ivc, 1, CAMELLIA_BLOCK_SIZE, source_file);
298 ctx = EVP_CIPHER_CTX_new();
299 EVP_DecryptInit_ex(ctx, EVP_camellia_128_cbc(), NULL, key, iv);
300
301 start = clock();
302 while((input_length = fread(input_buffer, 1, sizeof(input_buffer), source_file)) >0){
303     EVP_DecryptUpdate(ctx, output_buffer, &output_length, input_buffer,
304                     input_length);
305     fwrite(output_buffer, 1, output_length, destination_file);
306 }
307 EVP_DecryptFinal_ex(ctx, output_buffer, &output_length);
308 fwrite(output_buffer, 1, output_length, destination_file);
309 end = clock();
310
311 EVP_CIPHER_CTX_free(ctx);
312 fclose(source_file);
313 fclose(destination_file);
314
315 cpu_usage = end - start;
316 printf("Decryption of 16 byte file completed in %d ms using CAMELLIA-128-CBC\n",
317        cpu_usage);
318
319 source_file = fopen("camellia_medium_enc.bin", "rb");
320 destination_file = fopen("camellia_medium_dec.bin", "wb");
321
322 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)) >0){
322     EVP_DecryptUpdate(ctx, output_buffer, &output_length, input_buffer, input_length);
323     fwrite(output_buffer, 1, output_length, destination_file);
324 }
325 EVP_DecryptFinal_ex(ctx, output_buffer, &output_length);
326 fwrite(output_buffer, 1, output_length, destination_file);
327 end = clock();
328
329 EVP_CIPHER_CTX_free(ctx);
330 fclose(source_file);
331 fclose(destination_file);
332
333 cpu_usage = end - start;
334 printf("Decryption of 20KB file completed in %d ms using CAMELLIA-128-CBC\n",
335         cpu_usage);
336
337 source_file = fopen("camellia_large_enc.bin", "rb");
338 destination_file = fopen("camellia_large_dec.bin", "wb");
339
340 fread(ivc, 1, CAMELLIA_BLOCK_SIZE, source_file);
341 ctx = EVP_CIPHER_CTX_new();
342 EVP_DecryptInit_ex(ctx, EVP_camellia_128_cbc(), NULL, key, iv);
343
344 start = clock();
345 while((input_length = fread(input_buffer, 1, sizeof(input_buffer), source_file)) >0){
346     EVP_DecryptUpdate(ctx, output_buffer, &output_length, input_buffer, input_length);
347     fwrite(output_buffer, 1, output_length, destination_file);
348 }
349 EVP_DecryptFinal_ex(ctx, output_buffer, &output_length);
350 fwrite(output_buffer, 1, output_length, destination_file);
351 end = clock();
352
353 EVP_CIPHER_CTX_free(ctx);
354 fclose(source_file);
355 fclose(destination_file);
356
357 cpu_usage = end - start;
358 printf("Decryption of 2MB file completed in %d ms using CAMELLIA-128-CBC\n",
359         cpu_usage);
360 printf("%s\n", line);
361
362 /*encryption using SM4 CBC mode*/
363 source_file = fopen("random_small.txt", "rb");
364 destination_file = fopen("sm4_small_enc.bin", "wb");
365
366 ctx = EVP_CIPHER_CTX_new();
367 EVP_EncryptInit_ex(ctx, EVP_sm4_cbc(), NULL, key, iv);
368
369 start = clock();
370 while((input_length = fread(input_buffer, 1, sizeof(input_buffer), source_file)) >0){
371     EVP_EncryptUpdate(ctx, output_buffer, &output_length, input_buffer, input_length);
372     fwrite(output_buffer, 1, output_length, destination_file);
373 }
374 EVP_EncryptFinal_ex(ctx, output_buffer, &output_length);
375 fwrite(output_buffer, 1, output_length, destination_file);
376 end = clock();
377
378 EVP_CIPHER_CTX_free(ctx);
379 fclose(source_file);
380 fclose(destination_file);
381
382 cpu_usage = end - start;
383 printf("Encryption of 16 byte file completed in %d ms using SM4-128-CBC\n",
384         cpu_usage);
385
386 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)) >0){
391     EVP_EncryptUpdate(ctx, output_buffer, &output_length, input_buffer, input_length);
392     fwrite(output_buffer, 1, output_length, destination_file);
393 }
394 EVP_EncryptFinal_ex(ctx, output_buffer, &output_length);
395 fwrite(output_buffer, 1, output_length, destination_file);
396 end = clock();
397
398 EVP_CIPHER_CTX_free(ctx);
399 fclose(source_file);
400 fclose(destination_file);
401
402 cpu_usage = end - start;
403 printf("Encryption of 20KB file completed in %d ms using SM4-128-CBC\n",
        cpu_usage);
404
405 source_file = fopen("random_large.bin", "rb");
406 destination_file = fopen("sm4_large_enc.bin", "wb");
407
408 ctx = EVP_CIPHER_CTX_new();
409 EVP_EncryptInit_ex(ctx, EVP_sm4_cbc(), NULL, key, iv);
410
411 start = clock();
412 while((input_length = fread(input_buffer, 1, sizeof(input_buffer), source_file)) >0){
413     EVP_EncryptUpdate(ctx, output_buffer, &output_length, input_buffer, input_length);
414     fwrite(output_buffer, 1, output_length, destination_file);
415 }
416 EVP_EncryptFinal_ex(ctx, output_buffer, &output_length);
417 fwrite(output_buffer, 1, output_length, destination_file);
418 end = clock();
419
420 EVP_CIPHER_CTX_free(ctx);
421 fclose(source_file);
422 fclose(destination_file);
423
424 cpu_usage = end - start;
425 printf("Encryption of 2MB file completed in %d ms using SM4-128-CBC\n", cpu_usage);
426 printf("%s\n", line);
427
/*decryption using SM4 CBC mode*/
428 source_file = fopen("sm4_small_enc.bin", "rb");
429 destination_file = fopen("sm4_small_dec.bin", "wb");
430
431 fread(iv, 1, AES_BLOCK_SIZE, source_file);
432 ctx = EVP_CIPHER_CTX_new();
433 EVP_DecryptInit_ex(ctx, EVP_sm4_cbc(), NULL, key, iv);
434
435 start = clock();
436 while((input_length = fread(input_buffer, 1, sizeof(input_buffer), source_file)) >0){
437     EVP_DecryptUpdate(ctx, output_buffer, &output_length, input_buffer, input_length);
438     fwrite(output_buffer, 1, output_length, destination_file);
439 }
440 EVP_DecryptFinal_ex(ctx, output_buffer, &output_length);
441 fwrite(output_buffer, 1, output_length, destination_file);
442 end = clock();
443
444 EVP_CIPHER_CTX_free(ctx);
445 fclose(source_file);
446 fclose(destination_file);
447
448 cpu_usage = end - start;
449 printf("Decryption of 16 byte file completed in %d ms using SM4-128-CBC\n",
        cpu_usage);

```

```

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

```

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

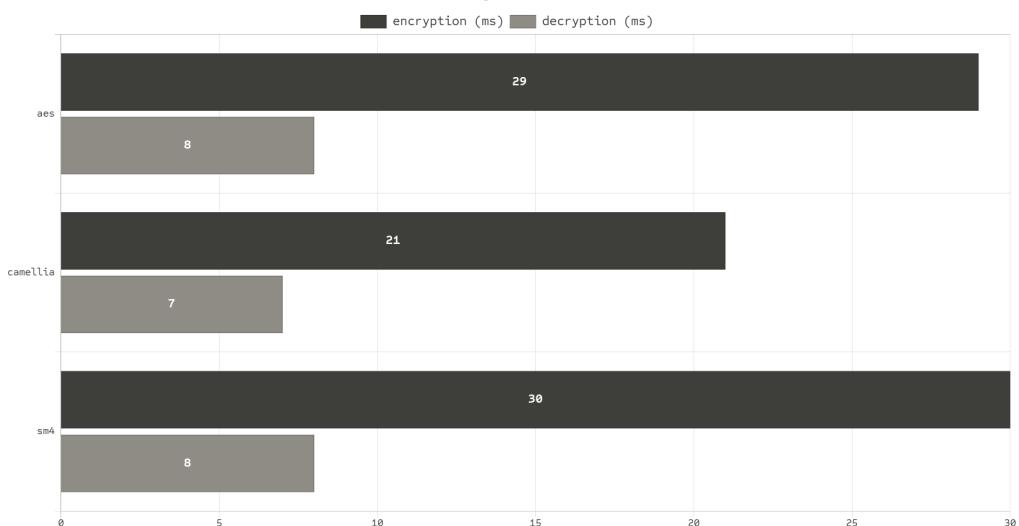
Encryption of 16 byte file completed in 21 ms using CAMELLIA-128-CBC
Encryption of 20KB file completed in 236 ms using CAMELLIA-128-CBC
Encryption of 2MB file completed in 22569 ms using CAMELLIA-128-CBC

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

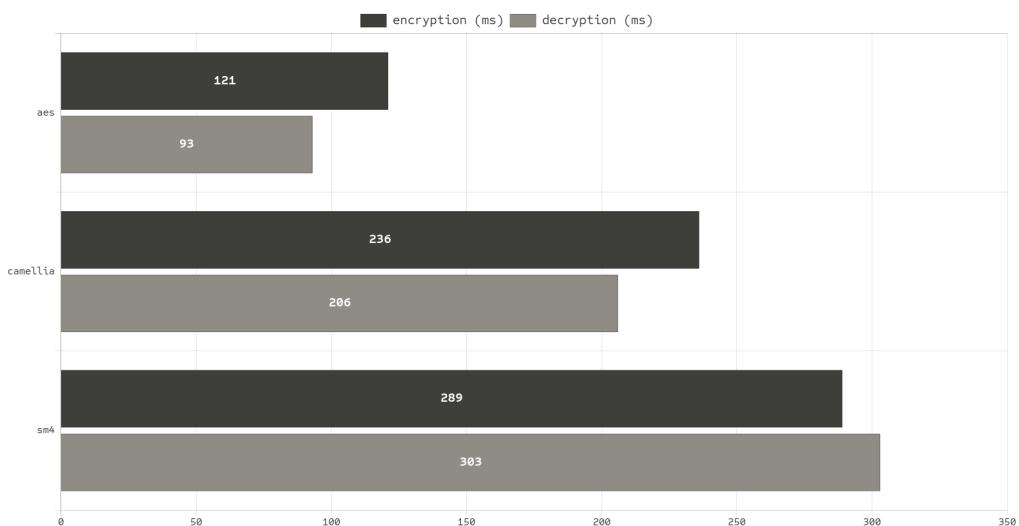
Encryption of 16 byte file completed in 30 ms using SM4-128-CBC
Encryption of 20KB file completed in 289 ms using SM4-128-CBC
Encryption of 2MB file completed in 30255 ms using SM4-128-CBC

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
```

### 16 byte file



### 20 kb file



### 2 mb file

