Owen Durkin
CS449

Question 1:

A.1 Openssl speed -seconds 1 aes-128-cbc

```
Doing aes-128 cbc for 1s on 16 size blocks: 16457990 aes-128 cbc's in 0.84s
Doing aes-128 cbc for 1s on 64 size blocks: 4301830 aes-128 cbc's in 0.89s
Doing aes-128 cbc for 1s on 256 size blocks: 1096236 aes-128 cbc's in 0.98s
Doing aes-128 cbc for 1s on 1024 size blocks: 274728 aes-128 cbc's in 0.92s
Doing aes-128 cbc for 1s on 8192 size blocks: 34849 aes-128 cbc's in 0.99s
Doing aes-128 cbc for 1s on 16384 size blocks: 17447 aes-128 cbc's in 0.98s

The 'numbers' are in 1000s of bytes per second processed.
type             16 bytes     64 bytes    256 bytes   1024 bytes   8192 bytes  16384 bytes
aes-128 cbc     312369.92k   308997.89k   285199.61k   305120.90k   289830.46k   290499.64k
```

A.2 Openssl speed -seconds 1 aes-192-cbc

```
Doing aes-192 cbc for 1s on 16 size blocks: 14393724 aes-192 cbc's in 0.95s
Doing aes-192 cbc for 1s on 64 size blocks: 3759733 aes-192 cbc's in 0.94s
Doing aes-192 cbc for 1s on 256 size blocks: 923624 aes-192 cbc's in 0.88s
Doing aes-192 cbc for 1s on 1024 size blocks: 232666 aes-192 cbc's in 0.81s
Doing aes-192 cbc for 1s on 8192 size blocks: 29062 aes-192 cbc's in 0.78s
Doing aes-192 cbc for 1s on 16384 size blocks: 14632 aes-192 cbc's in 0.89s

The 'numbers' are in 1000s of bytes per second processed.
type             16 bytes     64 bytes    256 bytes   1024 bytes   8192 bytes  16384 bytes
aes-192 cbc     241657.49k   256527.62k   270225.99k   293411.31k   304444.89k   269360.32k
```

A.3 Openssl speed -seconds 1 aes-256-cbc

```
Doing aes-256 cbc for 1s on 16 size blocks: 12681004 aes-256 cbc's in 0.95s
Doing aes-256 cbc for 1s on 64 size blocks: 3189931 aes-256 cbc's in 0.86s
Doing aes-256 cbc for 1s on 256 size blocks: 793577 aes-256 cbc's in 0.94s
Doing aes-256 cbc for 1s on 1024 size blocks: 205991 aes-256 cbc's in 0.97s
Doing aes-256 cbc for 1s on 8192 size blocks: 25492 aes-256 cbc's in 0.99s
Doing aes-256 cbc for 1s on 16384 size blocks: 12674 aes-256 cbc's in 0.95s

The 'numbers' are in 1000s of bytes per second processed.
type             16 bytes     64 bytes    256 bytes   1024 bytes   8192 bytes  16384 bytes
aes-256 cbc     212902.48k   237666.57k   216583.91k   217907.83k   212010.62k   217891.73k
```

A.4 Openssl speed -seconds 1 rsa

```
Doing 512 bits private rsa's for 1s: 23413 512 bits private RSA's in 0.99s
Doing 512 bits public rsa's for 1s: 228206 512 bits public RSA's in 0.97s
Doing 1024 bits private rsa's for 1s: 12295 1024 bits private RSA's in 0.97s
Doing 1024 bits public rsa's for 1s: 147149 1024 bits public RSA's in 0.91s
Doing 2048 bits private rsa's for 1s: 1910 2048 bits private RSA's in 0.99s
Doing 2048 bits public rsa's for 1s: 56923 2048 bits public RSA's in 0.86s
Doing 3072 bits private rsa's for 1s: 640 3072 bits private RSA's in 0.98s
Doing 3072 bits public rsa's for 1s: 30381 3072 bits public RSA's in 1.00s
Doing 4096 bits private rsa's for 1s: 281 4096 bits private RSA's in 0.91s
Doing 4096 bits public rsa's for 1s: 17869 4096 bits public RSA's in 0.98s
Doing 7680 bits private rsa's for 1s: 32 7680 bits private RSA's in 1.00s
Doing 7680 bits public rsa's for 1s: 5381 7680 bits public RSA's in 0.92s
Doing 15360 bits private rsa's for 1s: 7 15360 bits private RSA's in 1.06s
Doing 15360 bits public rsa's for 1s: 1392 15360 bits public RSA's in 0.94s

                  sign    verify    sign/s verify/s
rsa  512 bits 0.000042s 0.000004s  23769.5 235750.0
rsa 1024 bits 0.000079s 0.000006s  12688.3 162416.1
rsa 2048 bits 0.000516s 0.000015s   1939.1  66266.6
rsa 3072 bits 0.001538s 0.000033s    650.4  30381.0
rsa 4096 bits 0.003228s 0.000055s    309.8  18159.6
rsa 7680 bits 0.031250s 0.000171s     32.0   5836.2
rsa 15360 bits 0.151714s 0.000674s      6.6   1484.0
```

B.

For aes-xxx-cbc it can be noticed that as the key size increases, (128, 192, 256), we see a decrease in the number of operations able to be performed per second across all tested size blocks.
Interestingly, there is a transverse relation when it comes to the speed in which the data gets processed depending on the key size. For example, 256 processes a 16384-byte block in 217,891.73 KB/s while 128 processes the same size block in 290,499.64 KB/s.

A variety of factors can play a role in driving these trends such as algorithm overhead which is more noticeable on smaller data chunks because of setup cost being a larger portion of overall processing time. Opposing that however is parallelization which smaller blocks utilize better. Smaller data blocks may also fit inside the cache without having to fetch from main memory. One specific note is that at a certain point you can see certain block sizes get processed quicker because they are better optimized. There can also be differences in hardware that affect these values.
Essentially there are multiple factors at play, but it boils down to hardware, algorithm design, and software optimization. The optimal size will depend on these factors.

C.

RSA is altering the bit size as it runs tests unlike the AES which I set the key size of for each run. Inherently this command was running encryption times. If I wanted decryption times I would have to add the -decrypt flag which gives:

```
Doing 512 bits private rsa's for 1s: 23396 512 bits private RSA's in 1.00s
Doing 512 bits public rsa's for 1s: 231765 512 bits public RSA's in 0.97s
Doing 1024 bits private rsa's for 1s: 12144 1024 bits private RSA's in 1.00s
Doing 1024 bits public rsa's for 1s: 139063 1024 bits public RSA's in 0.92s
Doing 2048 bits private rsa's for 1s: 1878 2048 bits private RSA's in 0.94s
Doing 2048 bits public rsa's for 1s: 54323 2048 bits public RSA's in 0.91s
Doing 3072 bits private rsa's for 1s: 632 3072 bits private RSA's in 0.91s
Doing 3072 bits public rsa's for 1s: 29433 3072 bits public RSA's in 0.99s
Doing 4096 bits private rsa's for 1s: 284 4096 bits private RSA's in 0.92s
Doing 4096 bits public rsa's for 1s: 17376 4096 bits public RSA's in 0.89s
Doing 7680 bits private rsa's for 1s: 31 7680 bits private RSA's in 0.97s
Doing 7680 bits public rsa's for 1s: 5355 7680 bits public RSA's in 0.91s
Doing 15360 bits private rsa's for 1s: 7 15360 bits private RSA's in 1.11s
Doing 15360 bits public rsa's for 1s: 1388 15360 bits public RSA's in 0.94s
```

```
                   sign    verify    sign/s verify/s
rsa  512 bits 0.000043s 0.000004s  23396.0 239426.7
rsa 1024 bits 0.000082s 0.000007s  12144.0 150827.5
rsa 2048 bits 0.000499s 0.000017s   2002.1  59959.2
rsa 3072 bits 0.001434s 0.000033s    697.6  29881.2
rsa 4096 bits 0.003243s 0.000051s    308.4  19501.7
rsa 7680 bits 0.031258s 0.000169s     32.0   5910.6
rsa 15360 bits 0.158571s 0.000675s      6.3   1481.3
```

The encryption and decryption times are almost identical across the board, but any differences can most likely be attributed to the number of operations as there is slight variance. The factors that play a role in the trends we see are similar to that of AES, but some differences include mathematical and computational complexity. Mathematical complexity changes with key size and operations become slower with larger key sizes. Computational complexity also changes with key size and modular multiplications and exponentiations increase with larger key sizes. Aside from that it is mainly a matter of hardware choices due to variables like process architecture, which means some processors are made with cryptography in mind.