Department of Computer Science

San Francisco State University

CSC 852 Security and Data Privacy

Final Project Fall 2022

Analysis of Encryption and Decryption using DES, 3DES, and AES Algorithm

Submitted to Jingyi Wang

Submitted by MST Jasmine Jahan (920115557)

December 11, 2022

Introduction

Data Encryption Standard (DES) and Advanced Encryption Standard (AES) are the symmetric block cipher but both of them including 3DES has some advantages and disadvantages. We know from the definition that DES uses the Feistel network which divides the block into two halves for the process of encryption. Also, DES has a key length of 64 bits which is very small to prevent any kind of brute-force attacks. 3DES is just the 3 times repetitions of the DES algorithm. So in terms of key length and structure, it is similar to the DES algorithm.

On the other hand, AES has three different key lengths 128 bits, 192 bits, and 256 bits which make this algorithm more strong. AES uses Byte substitution layer and Key addition layer for the process of encryption.

To analyze and compare DES, 3DES, and AES algorithms the following steps were taken:

- Measure the time taken for encryption and decryption with eight different input sizes.
- Compare the average time to see which algorithm is fast or slow.
- Identify the component for which an algorithm is weak or strong.
- Identify which algorithm provides better security over other algorithms.

Procedure

In the coding part for this analysis Jupyter Notebook was used to measure time with different input sizes using the DES, 3DES, and AES algorithms from PyCryptodome. This allowed time measurement for each cell.

In the next stage, time(Micro Second) was plotted for eight different input sizes for each algorithm via Google sheets. For the visual representation bar charts and line, charts were used. The average value has been measured from each chart to have the right insight into the time taken by the algorithms. Below is the list of input sizes of each algorithm that was executed in Jupyter Notebook:

- Encryption and Decryption using DES with inputs 16bits / 32bits / 64bits / 256 bits / 512 bits/ 1024bits/ 2048bits
- Encryption and Decryption using DES with inputs 16bits / 32bits / 64bits / 256 bits / 512 bits/ 1024bits/ 2048bits
- Encryption and Decryption using 3DES with input 16bits / 32bits / 64bits / 256 bits / 512 bits/ 1024bits/ 2048bits
- Encryption and Decryption using AES with key length 128 bits and input 16bits / 32bits / 64bits / 256 bits / 512 bits/ 1024bits/ 2048bits
- Encryption and Decryption using AES with key length 192 bits and input 16bits / 32bits / 64bits / 256 bits / 512 bits/ 1024bits/ 2048bits
- Encryption and Decryption using AES with key length 256 bits and input 16bits / 32bits / 64bits / 256 bits / 512 bits/ 1024bits/ 2048bits

Output Screenshots

- Encryption and Decryption using DES Algorithm

```
In [5]: #Encryption and Decryption using DES, Input = 32bits

In [6]: %%time
        key = b'abcdefgh'
        cipher = DES.new(key,DES.MODE_ECB)
        plaintext = b'San Francisco State University Computer Science and Engi'
        msg = cipher.encrypt(plaintext)
        print(msg)
        decipher = DES.new(key,DES.MODE_ECB)
        msg_dec = decipher.decrypt(msg)
        print(msg_dec)

        b'wt\xa6p8[\xac\xe8\x85(EY,\xfd\xa9\xcfE\xf9\xc8\xac\xcd\xc9\xdc8V\x95\xe3T>\xdc\x14\xbf\xfa\xb6\xde\xe6\x92\x05\x1b
        \xdek\x02|\xeete\x0c\xff\x17\xe8\x07E\x93b\xf5\x1f'
        b'San Francisco State University Computer Science and Engi'
        CPU times: user 234 µs, sys: 103 µs, total: 337 µs
        Wall time: 238 µs
```

In [7]:
```
#Encryption and Decryption using DES, Input = 64bits
```

In [8]:
```
%%time
key = b'abcdefgh'
cipher = DES.new(key,DES.MODE_ECB)
plaintext = b'San Francisco State University Computer Science and EngiSan Francisco State University Computer Science a
msg = cipher.encrypt(plaintext)
print(msg)
decipher = DES.new(key,DES.MODE_ECB)
msg_dec = decipher.decrypt(msg)
print(msg_dec)
```

```
b'wt\xa6p8[\xac\xe8\x85(EY,\xfd\xa9\xcfE\xf9\xc8\xac\xcd\xc9\xdc8V\x95\xe3T>\xdc\x14\xbf\xfa\xb6\xde\xe6\x92\x05\x1b
\xdek\x02|\xeete\x0c\xff\x17\xe8\x07E\x93b\xf5\x1fwt\xa6p8[\xac\xe8\x85(EY,\xfd\xa9\xcfE\xf9\xc8\xac\xcd\xc9\xdc8V\x9
5\xe3T>\xdc\x14\xbf\xfa\xb6\xde\xe6\x92\x05\x1b\xdek\x02|\xeete\x0c\xff\x17\xe8\x07E\x93b\xf5\x1f'
b'San Francisco State University Computer Science and EngiSan Francisco State University Computer Science and Engi'
CPU times: user 306 µs, sys: 258 µs, total: 564 µs
Wall time: 335 µs
```

In [9]:
```
#Encryption and Decryption using DES, Input = 128bits
```

In [10]:
```
%%time
key = b'abcdefgh'
cipher = DES.new(key,DES.MODE_ECB)
plaintext = b'San Francisco State University Computer Science and EngiSan Francisco State University Computer Science a
msg = cipher.encrypt(plaintext)
print(msg)
decipher = DES.new(key,DES.MODE_ECB)
msg_dec = decipher.decrypt(msg)
print(msg_dec)
```

```
b'wt\xa6p8[\xac\xe8\x85(EY,\xfd\xa9\xcfE\xf9\xc8\xac\xcd\xc9\xdc8V\x95\xe3T>\xdc\x14\xbf\xfa\xb6\xde\xe6\x92\x05\x1b
\xdek\x02|\xeete\x0c\xff\x17\xe8\x07E\x93b\xf5\x1fwt\xa6p8[\xac\xe8\x85(EY,\xfd\xa9\xcfE\xf9\xc8\xac\xcd\xc9\xdc8V\x9
5\xe3T>\xdc\x14\xbf\xfa\xb6\xde\xe6\x92\x05\x1b\xdek\x02|\xeete\x0c\xff\x17\xe8\x07E\x93b\xf5\x1fwt\xa6p8[\xac\xe8\x8
5(EY,\xfd\xa9\xcfE\xf9\xc8\xac\xcd\xc9\xdc8V\x95\xe3T>\xdc\x14\xbf\xfa\xb6\xde\xe6\x92\x05\x1b\xdek\x02|\xeete\x0c\xf
f\x17\xe8\x07E\x93b\xf5\x1fwt\xa6p8[\xac\xe8\x85(EY,\xfd\xa9\xcfE\xf9\xc8\xac\xcd\xc9\xdc8V\x95\xe3T>\xdc\x14\xbf\xfa
\xb6\xde\xe6\x92\x05\x1b\xdek\x02|\xeete\x0c\xff\x17\xe8\x07E\x93b\xf5\x1f'
b'San Francisco State University Computer Science and EngiSan Francisco State University Computer Science and EngiSan
Francisco State University Computer Science and EngiSan Francisco State University Computer Science and Engi'
CPU times: user 368 µs, sys: 332 µs, total: 700 µs
Wall time: 362 µs
```

- Encryption and Decryption using 3DES Algorithm

In [7]:
```
#Encryption and Decryption using 3DES, Input = 32bits
```

In [8]:
```
%%time
key = b'abcdefgh'
plaintext = b'San Francisco State University Computer Science and Engi'
while True:
    try:
        key = DES3.adjust_key_parity(get_random_bytes(24))
        break
    except ValueError:
        pass
cipher = DES3.new(key, DES3.MODE_ECB)
msg = cipher.encrypt(plaintext)
print(msg)
decipher = DES3.new(key,DES3.MODE_ECB)
msg_dec = decipher.decrypt(msg)
print(msg_dec)
```

```
b'\x84u \x97\xcc\xcc\xb1\xcf\xfb\x86\xd2\x8c\x8fQP\xd0{\x99&MPrB\x0e\x02f\x8c\xd1q\xe3b\xc6pH\x91\xd7"\xfc+\xca?8?\x8
el6\x8f\xc8\x16M|+\xd2aV\x89'
b'San Francisco State University Computer Science and Engi'
CPU times: user 502 µs, sys: 248 µs, total: 750 µs
Wall time: 502 µs
```

In [9]:
```
#Encryption and Decryption using 3DES, Input = 64bits
```

In [10]:
```
%%time
key = b'abcdefgh'
plaintext = b'San Francisco State University Computer Science and EngiSan Francisco State University Computer Science a
while True:
    try:
        key = DES3.adjust_key_parity(get_random_bytes(24))
        break
    except ValueError:
        pass
cipher = DES3.new(key, DES3.MODE_ECB)
msg = cipher.encrypt(plaintext)
print(msg)
decipher = DES3.new(key,DES3.MODE_ECB)
msg_dec = decipher.decrypt(msg)
print(msg_dec)
```

```
b'MD@\x01\x1a\xf0,\xacX\x93\xc5\x7f\xe7\xafHz\xcf9\x0e|\x0c\x97m\xd8\xdb\x1e\x85\xd7\xfaJ\xd1y\xe9\x07\xf5S*\xd2s\xad
\xc8\xd6\xab\x1df\xa9\r\x87\xc3\xd5\x05\xdb\xd4\x99\\\xd0MD@\x01\x1a\xf0,\xacX\x93\xc5\x7f\xe7\xafHz\xcf9\x0e|\x0c\x9
7m\xd8\xdb\x1e\x85\xd7\xfaJ\xd1y\xe9\x07\xf5S*\xd2s\xad\xc8\xd6\xab\x1df\xa9\r\x87\xc3\xd5\x05\xdb\xd4\x99\\\xd0'
b'San Francisco State University Computer Science and EngiSan Francisco State University Computer Science and Engi'
CPU times: user 468 µs, sys: 194 µs, total: 662 µs
Wall time: 413 µs
```

In [11]:
```
#Encryption and Decryption using 3DES, Input = 128bits
```

In [12]:
```
%%time
key = b'abcdefgh'
plaintext = b'San Francisco State University Computer Science and EngiSan Francisco State University Computer Science a
while True:
    try:
        key = DES3.adjust_key_parity(get_random_bytes(24))
        break
    except ValueError:
        pass
cipher = DES3.new(key, DES3.MODE_ECB)
msg = cipher.encrypt(plaintext)
print(msg)
decipher = DES3.new(key,DES3.MODE_ECB)
msg_dec = decipher.decrypt(msg)
print(msg_dec)
```

```
b'\x94\xdc\x938BN\x8d\xf9Ax\xf37\xc8\x9fx\xd9\xf3\x90\x0e\x8b\x95\xb9\xc7\x11\x83\x90c:V\xfa\xc4\xbe\x1cF*\xd5\x17\xc
7\x91oP\x88i\t\x19\x08~.Qo\xbfes)i\xc9\x94\xdc\x938BN\x8d\xf9Ax\xf37\xc8\x9fx\xd9\xf3\x90\x0e\x8b\x95\xb9\xc7\x11\x83
\x90c:V\xfa\xc4\xbe\x1cF*\xd5\x17\xc7\x91oP\x88i\t\x19\x08~.Qo\xbfes)i\xc9\x94\xdc\x938BN\x8d\xf9Ax\xf37\xc8\x9fx\xd9
\xf3\x90\x0e\x8b\x95\xb9\xc7\x11\x83\x90c:V\xfa\xc4\xbe\x1cF*\xd5\x17\xc7\x91oP\x88i\t\x19\x08~.Qo\xbfes)i\xc9\x94\xd
c\x938BN\x8d\xf9Ax\xf37\xc8\x9fx\xd9\xf3\x90\x0e\x8b\x95\xb9\xc7\x11\x83\x90c:V\xfa\xc4\xbe\x1cF*\xd5\x17\xc7\x91oP\x
88i\t\x19\x08~.Qo\xbfes)i\xc9'
b'San Francisco State University Computer Science and EngiSan Francisco State University Computer Science and EngiSan
Francisco State University Computer Science and EngiSan Francisco State University Computer Science and Engi'
CPU times: user 628 µs, sys: 508 µs, total: 1.14 ms
Wall time: 692 µs
```

- Encryption and Decryption using AES Algorithm

In [5]:
```
#Encryption and Decryption using AES, Key length 128bits & Input = 32bits
```

In [6]:
```
%%time
key = b'Sixteen byte key'
plaintext = b'Coding InterviewCoding Interview'
cipher = AES.new(key,AES.MODE_ECB)
msg = cipher.encrypt(plaintext)
print(msg)
decipher = AES.new(key,AES.MODE_ECB)
msg_dec = decipher.decrypt(msg)
print(msg_dec)
```

```
b'\xb2\xfb6\xa9q\xd3\xf5\xae\xdeO\xa0QlH\xeb\x8d\xb2\xfb6\xa9q\xd3\xf5\xae\xdeO\xa0QlH\xeb\x8d'
b'Coding InterviewCoding Interview'
CPU times: user 113 µs, sys: 28 µs, total: 141 µs
Wall time: 118 µs
```

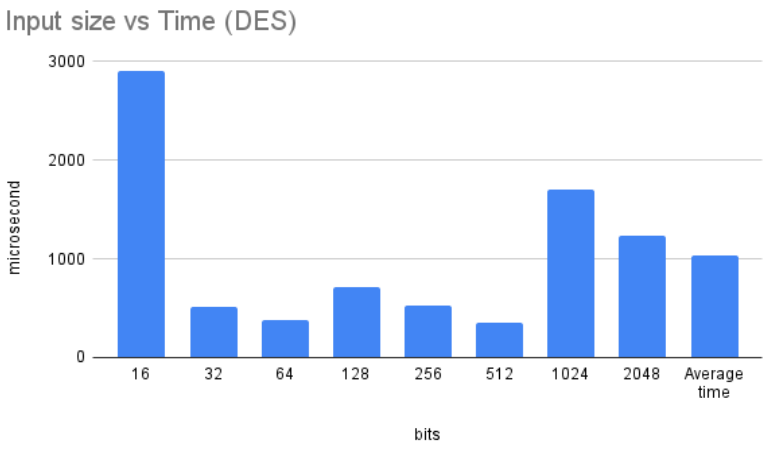
```
In [21]: #Encryption and Decryption using AES, Key length 192bits & Input = 64bits
```

```
In [22]: %%time
key = b'San Francisco State Univ'
plaintext = b'Coding InterviewCoding InterviewCoding InterviewCoding Interview'
cipher = AES.new(key,AES.MODE_ECB)
msg = cipher.encrypt(plaintext)
print(msg)
decipher = AES.new(key,AES.MODE_ECB)
msg_dec = decipher.decrypt(msg)
print(msg_dec)

b'%XkT\x8e\xf2\xbb\x1bI\xf5!\xeb\x9c\xa0G\x10%XkT\x8e\xf2\xbb\x1bI\xf5!\xeb\x9c\xa0G\x10%XkT\x8e\xf2\xbb\x1bI\xf5!\xe
b\x9c\xa0G\x10%XkT\x8e\xf2\xbb\x1bI\xf5!\xeb\x9c\xa0G\x10'
b'Coding InterviewCoding InterviewCoding InterviewCoding Interview'
CPU times: user 350 µs, sys: 195 µs, total: 545 µs
Wall time: 451 µs
```

```
In [39]: #Encryption and Decryption using AES, Key length 256bits & Input = 256bits
```

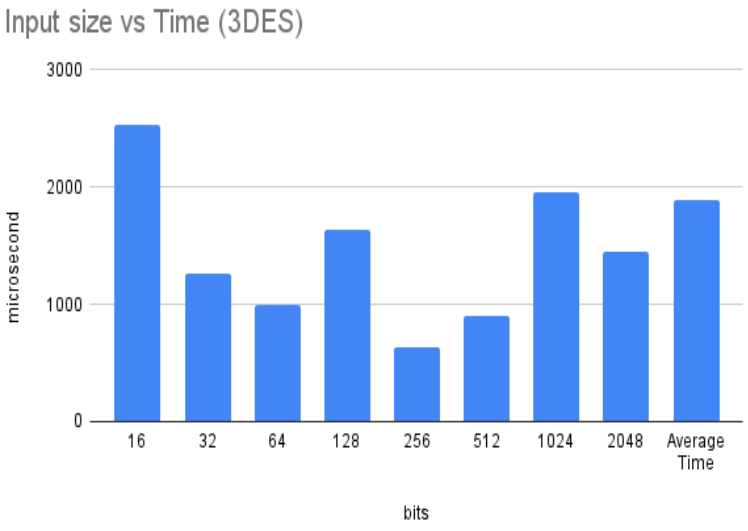
```
In [40]: %%time
from Crypto.Cipher import AES
key = b'Leetcode Prework PracticeSession'
plaintext = b'Coding InterviewCoding InterviewCoding InterviewCoding InterviewCoding InterviewCoding InterviewCoding Ir
cipher = AES.new(key,AES.MODE_ECB)
msg = cipher.encrypt(plaintext)
print(msg)
decipher = AES.new(key,AES.MODE_ECB)
msg_dec = decipher.decrypt(msg)
print(msg_dec)

b'\xca\xc6\xde\xdb&+\xac\xa8\xab\xf9\xe0\xe0|\xae\xbf\xd3\xca\xc6\xde\xdb&+\xac\xa8\xab\xf9\xe0\xe0|\xae\xbf\xd3\xca
\xc6\xde\xdb&+\xac\xa8\xab\xf9\xe0\xe0|\xae\xbf\xd3\xca\xc6\xde\xdb&+\xac\xa8\xab\xf9\xe0\xe0|\xae\xbf\xd3\xca\xc6\xd
e\xdb&+\xac\xa8\xab\xf9\xe0\xe0|\xae\xbf\xd3\xca\xc6\xde\xdb&+\xac\xa8\xab\xf9\xe0\xe0|\xae\xbf\xd3\xca\xc6\xde\xdb&+
\xac\xa8\xab\xf9\xe0\xe0|\xae\xbf\xd3\xca\xc6\xde\xdb&+\xac\xa8\xab\xf9\xe0\xe0|\xae\xbf\xd3\xca\xc6\xde\xdb&+\xac\xa
8\xab\xf9\xe0\xe0|\xae\xbf\xd3\xca\xc6\xde\xdb&+\xac\xa8\xab\xf9\xe0\xe0|\xae\xbf\xd3\xca\xc6\xde\xdb&+\xac\xa8\xab\x
f9\xe0\xe0|\xae\xbf\xd3\xca\xc6\xde\xdb&+\xac\xa8\xab\xf9\xe0\xe0|\xae\xbf\xd3\xca\xc6\xde\xdb&+\xac\xa8\xab\xf9\xe0
\xe0|\xae\xbf\xd3\xca\xc6\xde\xdb&+\xac\xa8\xab\xf9\xe0\xe0|\xae\xbf\xd3\xca\xc6\xde\xdb&+\xac\xa8\xab\xf9\xe0\xe0|\x
ae\xbf\xd3\xca\xc6\xde\xdb&+\xac\xa8\xab\xf9\xe0\xe0|\xae\xbf\xd3'
b'Coding InterviewCoding InterviewCoding InterviewCoding InterviewCoding InterviewCoding InterviewCoding InterviewCod
ing InterviewCoding InterviewCoding InterviewCoding InterviewCoding InterviewCoding InterviewCoding InterviewCoding I
nterviewCoding Interview'
CPU times: user 109 µs, sys: 27 µs, total: 136 µs
Wall time: 112 µs
```

Time Measurement



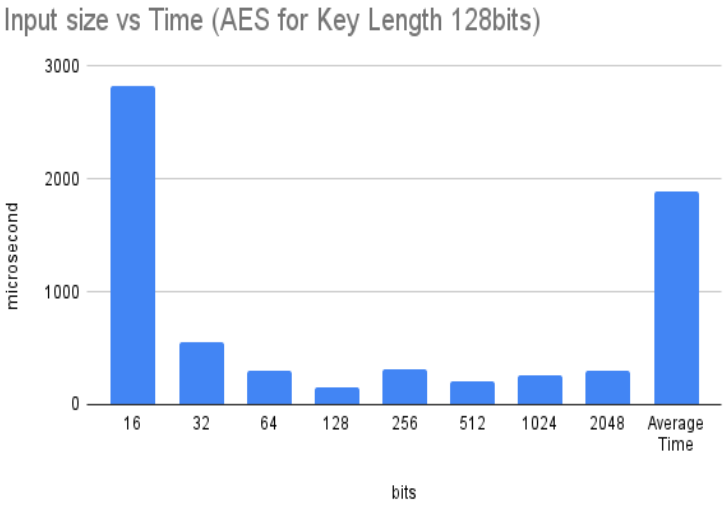
Bits	Microsecond
16	2900
32	514
64	374
128	719
256	522
512	358
1024	1700
2048	1240
Average time	1040.875

Figure 1 Encryption & Decryption using DES



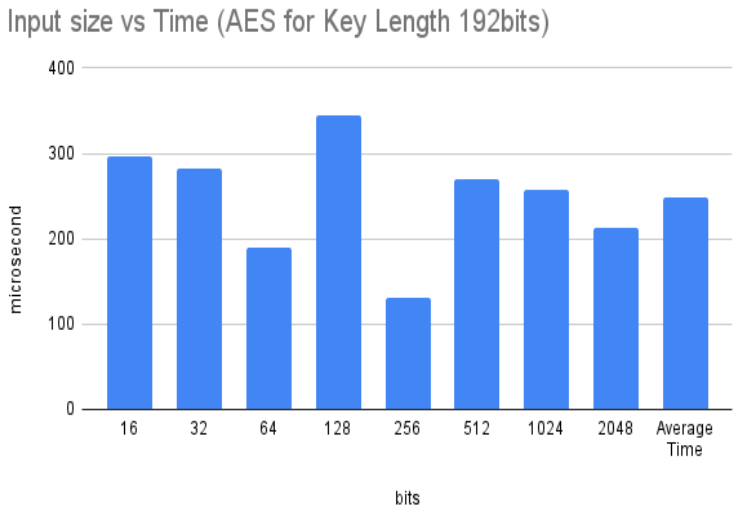
Bits	Microsecond
16	2530
32	1260
64	1000
128	1630
256	630
512	896
1024	1960
2048	1450
Average Time	1895

Figure 2 Encryption and Decryption using 3DES



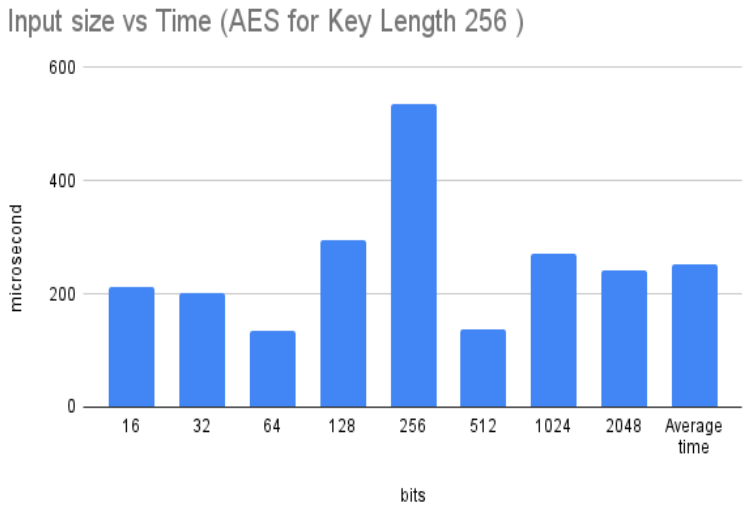
Bits	Microsecond
16	2830
32	552
64	294
128	157
256	309
512	202
1024	260
2048	296
Average Time	1895

Figure 3 Encryption and Decryption using AES with key length 128 bits



Bits	Microsecond
16	296
32	283
64	189
128	345
256	131
512	269
1024	257
2048	212
Average Time	247.75

Figure 4 Encryption and Decryption using AES with key length 192 bits



Bits	Microsecond
16	212
32	202
64	134
128	295
256	535
512	137
1024	271
2048	241
Average time	253.375

Figure 5 Encryption and Decryption using AES with key length 256 bits

Comparison Charts

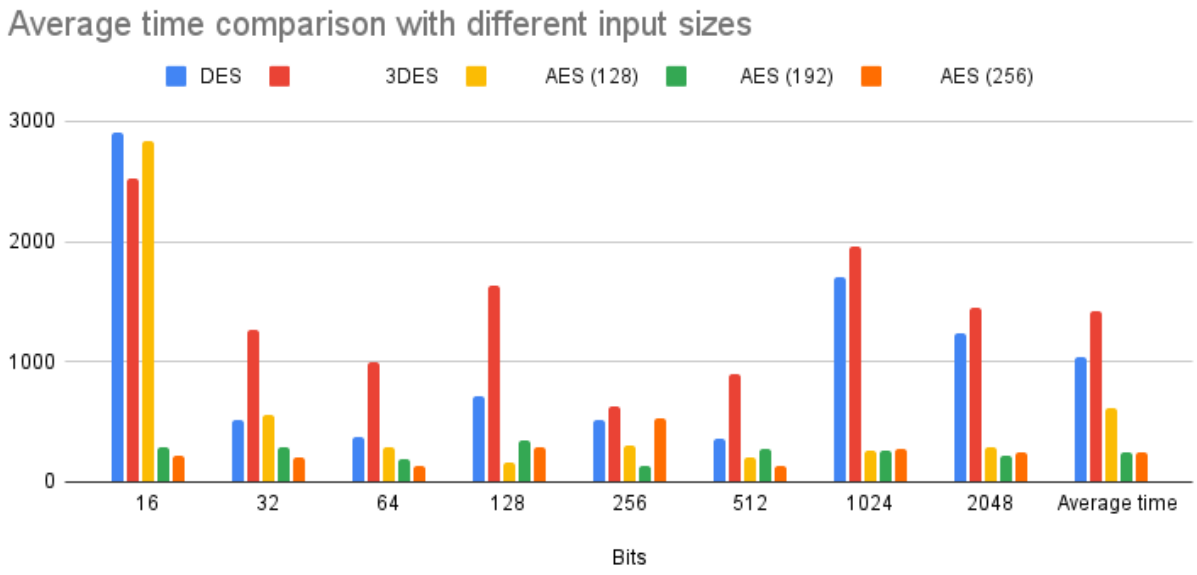


Figure 6 Visual representation of DES, 3DES and AES with different input sizes

Bits	DES	3DES	AES (128)	AES (192)	AES (256)
16	2900	2530	2830	296	212
32	514	1260	552	283	202
64	374	1000	294	189	134
128	719	1630	157	345	295
256	522	630	309	131	535
512	358	896	202	269	137
1024	1700	1960	260	257	271
2048	1240	1450	296	212	241
Average time	**1040.875**	**1419.5**	**612.5**	**247.75**	**253.375**

Table 1 Comparative execution times (in micro seconds) with different input sizes

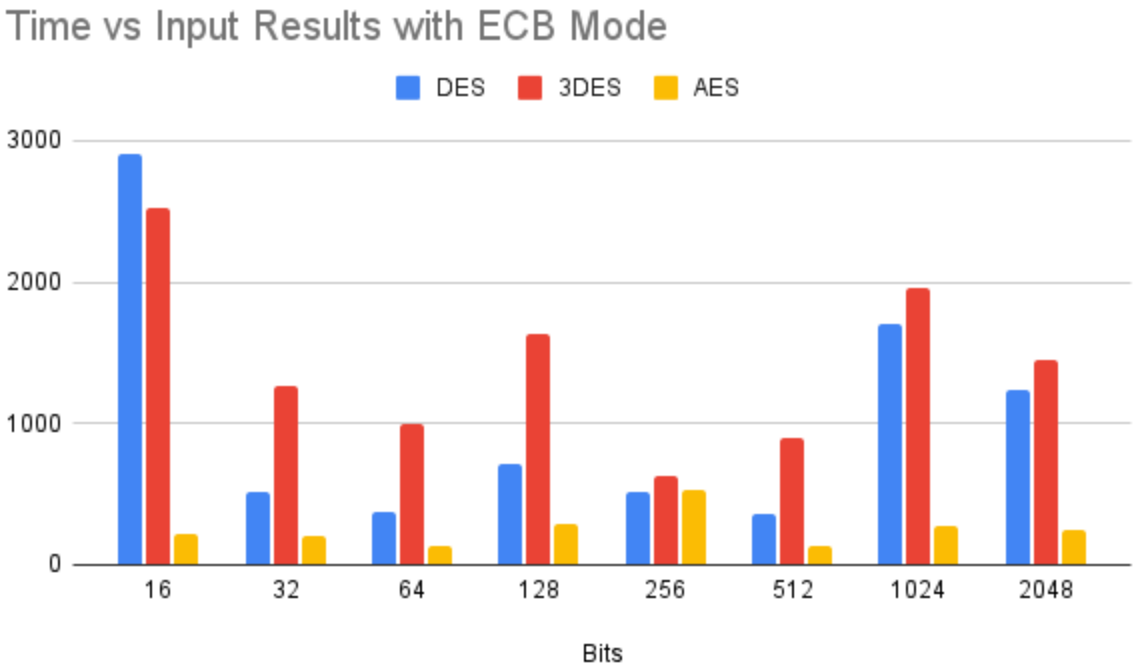


Figure 7 Visual representation of DES, 3DES and AES with ECB Mode

Bits	DES	3DES	AES
16	2900	2530	212
32	514	1260	202
64	374	1000	134
128	719	1630	295
256	522	630	535
512	358	896	137
1024	1700	1960	271
2048	1240	1450	241

Table 2 Comparative execution times (in micro seconds) with ECB Mode

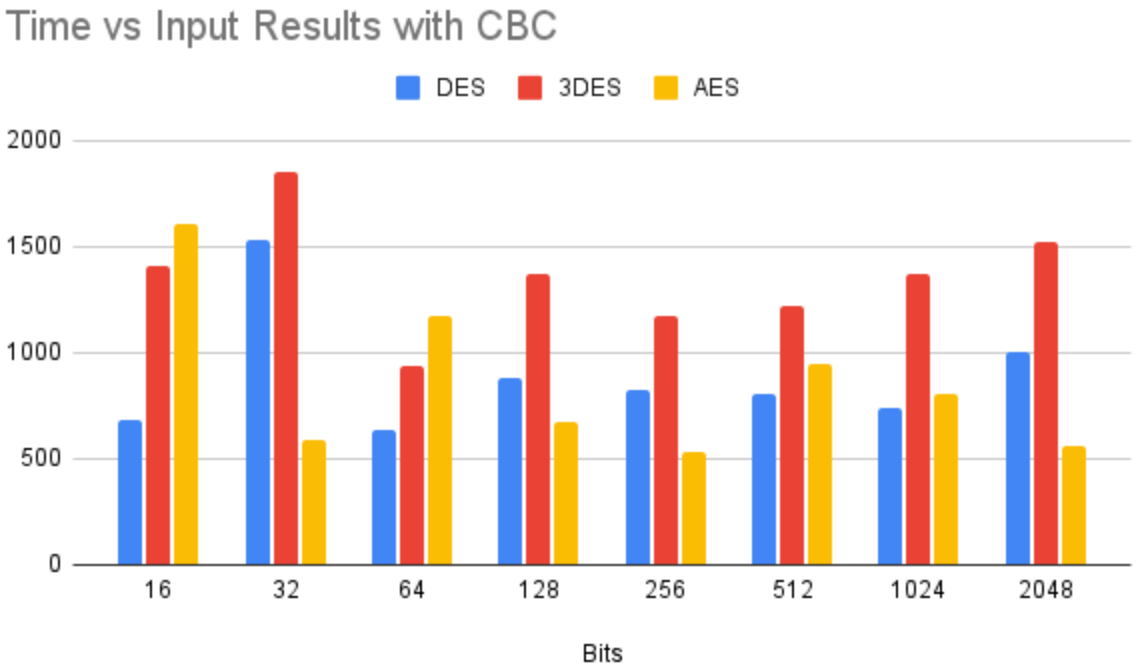


Figure 7 Visual representation of DES, 3DES and AES with CBC Mode

Bits	DES	3DES	AES
16	679	1410	1610
32	1530	1850	593
64	640	935	1170
128	885	1370	669
256	824	1170	536
512	805	1220	947
1024	735	1370	801
2048	1000	1520	563

Table 3 Comparative execution times (in microseconds) with CBC Mode

	DES	3DES	AES
Average Time	1040 Micro Sec	1419 Micro Sec	371 Micro Sec
Speed (Slow/Fast)	DES is comparatively slower	3DES is 3x of regular DES. It is also slower.	AES is faster.
Key Length	56 bits	56/112/168 bits	128, 192, 256 bits
Security (Week/Strong)	Smaller key which makes DES less secure	3DES is more secure than DES. But less secure than AES	AES has large and multiple keys which makes it more secure.
Component	Feistel network	Feistel network	Byte Substitution & Key Addition Layer

Table 4 Comparison of DES, 3DES, and AES based on speed and security

Conclusion

In this work, by analyzing and comparing DES, 3DES, and AES algorithm we have seen AES is faster compared to DES and 3DES. DES uses a Feistel network structure for encryption and decryption procedure. AES algorithm uses the Byte substitution layer and Key Addition layer. DES can be easily broken with an exhaustive key-search attack since DES has only 56 bits limited key length which is very small. So plain DES is not secure for most applications anymore.

In the case of 3DES, the encryption key is still limited to 56/112/168 bits as replicated by the DES algorithm. We have seen repeating the same process three times take a longer time. 3DES relied on the Feistel network as well. AES algorithm has a longer and stronger key length compared to DES and 3DES which makes AES secure. The average time for AES using different input sizes is less than DES and 3DES which makes AES algorithm faster. So, 3DES and DES algorithm takes longer for encryption and decryption than the AES algorithm. We can conclude that AES is a fast and secure algorithm.