Homework 2

Track A

LAB 0 – DETERMINE $gcd\ (12075, 4655)$

Rewrite as Extended Euclidean

$$gcd\ (a, b) = ax + by$$ *extended euclidean*

$$gcd\ (12075, 4655) = 12075x + 4655y$$ *substitution*

Use Division Algorithm to Solve Euclidean Equations

$$a = bq + r$$
$$12075 = 4655q + r$$
$$12075 = 4655(2) + 2765$$

solve q and r to find a

$$a = bq + r$$
$$4655 = 2765q + r$$
$$4655 = 2765(1) + 1890$$

Repeat until $r = 0$, using b as a and r as b

$$a = bq + r$$
$$2765 = 1890q + r$$
$$2765 = 1890(1) + 875$$

Repeat until $r = 0$, using b as a and r as b

$$a = bq + r$$
$$1890 = 875q + r$$
$$2765 = 875(2) + 140$$

Repeat until $r = 0$, using b as a and r as b

$$a = bq + r$$
$$875 = 140q + r$$
$$875 = 140(6) + 35$$

Repeat until $r = 0$, using b as a and r as b

$$a = bq + r$$
$$140 = 35q + r$$
$$140 = 35(4) + 0$$

Repeat until $r = 0$, using b as a and r as b

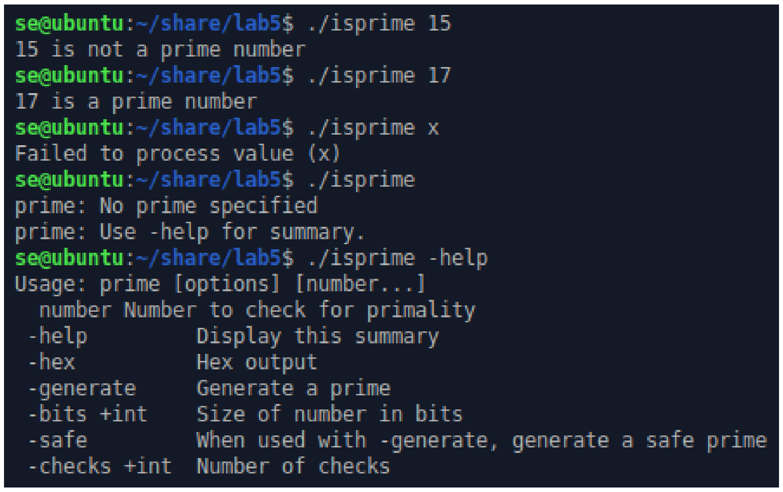
$$gcd\ (12075, 4655) = \mathbf{35}$$

When $r = 0$, $b =$ gcd

LAB 5 – (OPENSSL) GENERATE RANDOM NUMBER AND TEST PRIMALITY

1. (10pts) Based on the script primeTest, write another shell script using OpenSSL command line. The script is called isprime. It requires an integer as an input parameter. If the input integer is a prime, the script will show "*** is a prime". If not, the script will show "*** is not a prime".

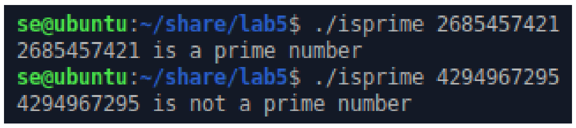
```
#!/bin/bash

input=$1

# Use OpenSSL to check if prime
result=$(openssl prime $input)

# Verify OpenSSL Ouput and display message
if [[ "$result" == *is\ prime ]]; then
    echo "$input is a prime number"

elif [[ "$result" == *not\ prime ]]; then
    echo "$input is not a prime number"
fi
```



```
se@ubuntu:~/share/lab5$ ./isprime 15
15 is not a prime number
se@ubuntu:~/share/lab5$ ./isprime 17
17 is a prime number
se@ubuntu:~/share/lab5$ ./isprime x
Failed to process value (x)
se@ubuntu:~/share/lab5$ ./isprime
prime: No prime specified
prime: Use -help for summary.
se@ubuntu:~/share/lab5$ ./isprime -help
Usage: prime [options] [number...]
  number Number to check for primality
 -help         Display this summary
 -hex          Hex output
 -generate     Generate a prime
 -bits +int    Size of number in bits
 -safe         When used with -generate, generate a safe prime
 -checks +int  Number of checks
```

2. (5pts) Use the program to decide if the following numbers are prime numbers:

   a. 2685457421 – Yes, this is a prime number
   b. 4294967295 – No, this is not a prime number



```
se@ubuntu:~/share/lab5$ ./isprime 2685457421
2685457421 is a prime number
se@ubuntu:~/share/lab5$ ./isprime 4294967295
4294967295 is not a prime number
```

3. (5pts) If an integer passes the Miller-Rabin primality test, does that guarantee the number is a prime number? Why?

No, an integer passing the Miller-Rabin primality test does not guarantee that it is prime. Since the Miller-Rabin primality test is a probabilistic primality test, it can only indicate that the integer is a probable prime. In other words, since the test relies on randomness in its computations, there are scenarios where a composite number might pass the test as if it were prime.

LAB 6 – (OpenSSL) Create RSA Public/Private Key (512bits) without Password Protection

1. (8pts) In Step 1, what is the public key (e, n) used? Why?

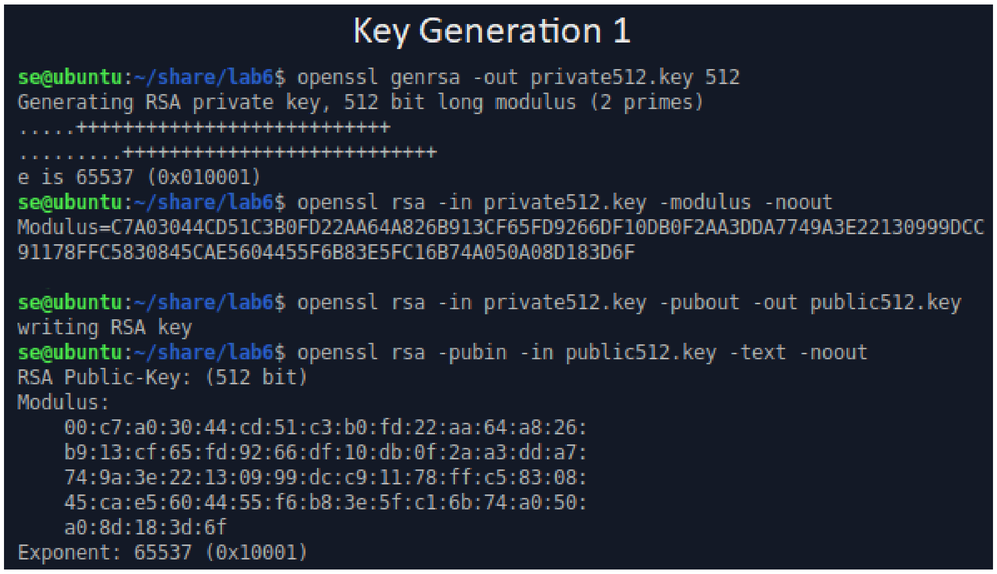
The exponent, 65537 (0x10001), is shown during RSA private key generation in Step 1. This number is popular for RSA key generation because it is a prime number, thus making it relatively prime to the numbers $(p - 1)$ and $(q - 1)$, where p and q are the prime factors of the modulus (n). Also, it is small enough to not slow down the encryption or decryption processes, while still being large enough that when combined with a large modulus (i.e. 2048-bit or 4096-bit), it creates a high level of complexity. This complexity makes it practically impossible for someone to break the encryption by trying to factorize the modulus and obtain the private key.

Alternatively, n is not shown during the RSA private key generation in Step 1 as it is randomized during each generation. Unlike the exponent, it is created by multiplying two randomly selected large prime numbers, p and q. This process ensures that each key pair is unique and not predictable. However, you can use the following OpenSSL command on the private key to display the value of n:

```
$ openssl rsa -in private512.key -modulus -noout
```

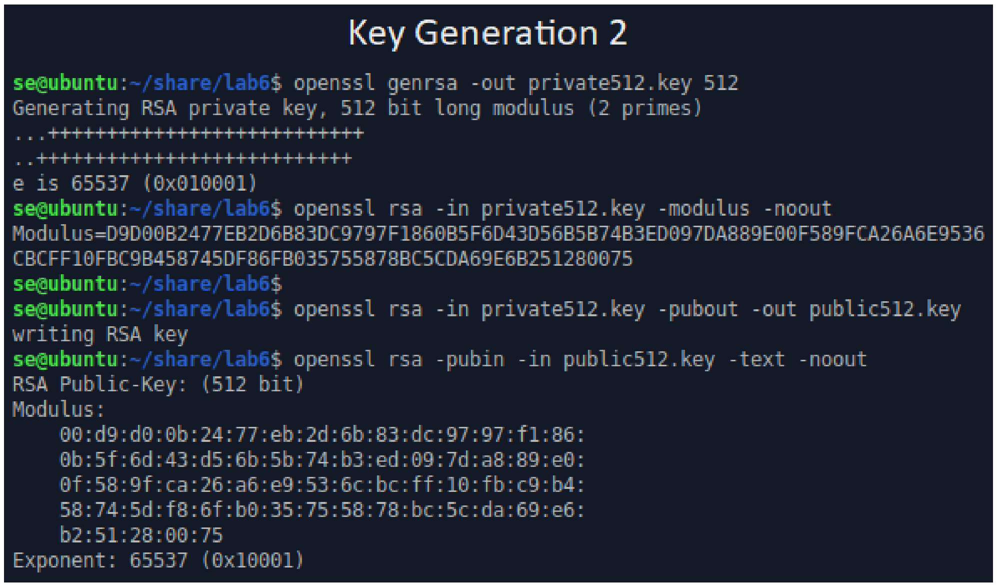
or the following OpenSSL command on the public key:

```
$ openssl rsa -pubin -in public512.key -text -noout
```



Key Generation 1

```
se@ubuntu:~/share/lab6$ openssl genrsa -out private512.key 512
Generating RSA private key, 512 bit long modulus (2 primes)
.....+++++++++++++++++++++++++++
.........+++++++++++++++++++++++++++
e is 65537 (0x010001)
se@ubuntu:~/share/lab6$ openssl rsa -in private512.key -modulus -noout
Modulus=C7A03044CD51C3B0FD22AA64A826B913CF65FD9266DF10DB0F2AA3DDA7749A3E22130999DCC
91178FFC5830845CAE5604455F6B83E5FC16B74A050A08D183D6F

se@ubuntu:~/share/lab6$ openssl rsa -in private512.key -pubout -out public512.key
writing RSA key
se@ubuntu:~/share/lab6$ openssl rsa -pubin -in public512.key -text -noout
RSA Public-Key: (512 bit)
Modulus:
    00:c7:a0:30:44:cd:51:c3:b0:fd:22:aa:64:a8:26:
    b9:13:cf:65:fd:92:66:df:10:db:0f:2a:a3:dd:a7:
    74:9a:3e:22:13:09:99:dc:c9:11:78:ff:c5:83:08:
    45:ca:e5:60:44:55:f6:b8:3e:5f:c1:6b:74:a0:50:
    a0:8d:18:3d:6f
Exponent: 65537 (0x10001)
```

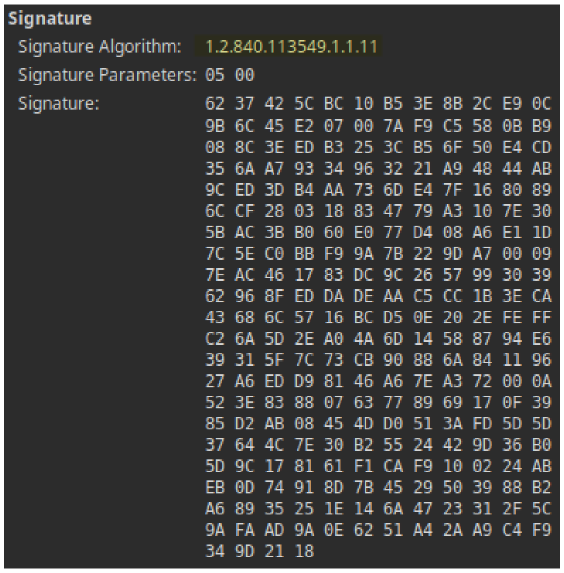


Key Generation 2

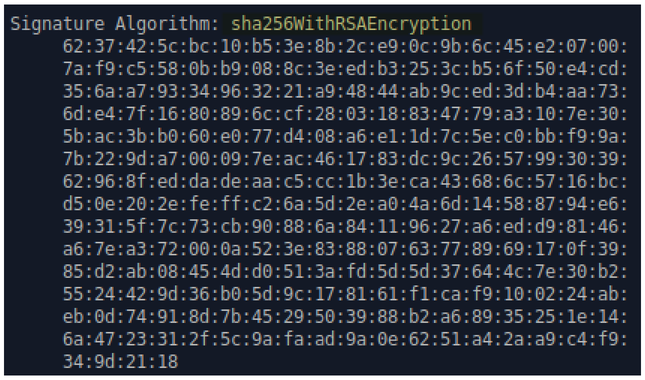
```
se@ubuntu:~/share/lab6$ openssl genrsa -out private512.key 512
Generating RSA private key, 512 bit long modulus (2 primes)
...+++++++++++++++++++++++++++
..+++++++++++++++++++++++++++
e is 65537 (0x010001)
se@ubuntu:~/share/lab6$ openssl rsa -in private512.key -modulus -noout
Modulus=D9D00B2477EB2D6B83DC9797F1860B5F6D43D56B5B74B3ED097DA889E00F589FCA26A6E9536
CBCFF10FBC9B458745DF86FB035755878BC5CDA69E6B251280075
se@ubuntu:~/share/lab6$
se@ubuntu:~/share/lab6$ openssl rsa -in private512.key -pubout -out public512.key
writing RSA key
se@ubuntu:~/share/lab6$ openssl rsa -pubin -in public512.key -text -noout
RSA Public-Key: (512 bit)
Modulus:
    00:d9:d0:0b:24:77:eb:2d:6b:83:dc:97:97:f1:86:
    0b:5f:6d:43:d5:6b:5b:74:b3:ed:09:7d:a8:89:e0:
    0f:58:9f:ca:26:a6:e9:53:6c:bc:ff:10:fb:c9:b4:
    58:74:5d:f8:6f:b0:35:75:58:78:bc:5c:da:69:e6:
    b2:51:28:00:75
Exponent: 65537 (0x10001)
```

2. (12pts) A certificate, Amazon.cer, can be found in the lab 6 folder, answer the following questions:
   a. What is the signature hash algorithm used to create the certificate?

1.2.840.113549.1.1.11 (sha256WithRSAEncryption)



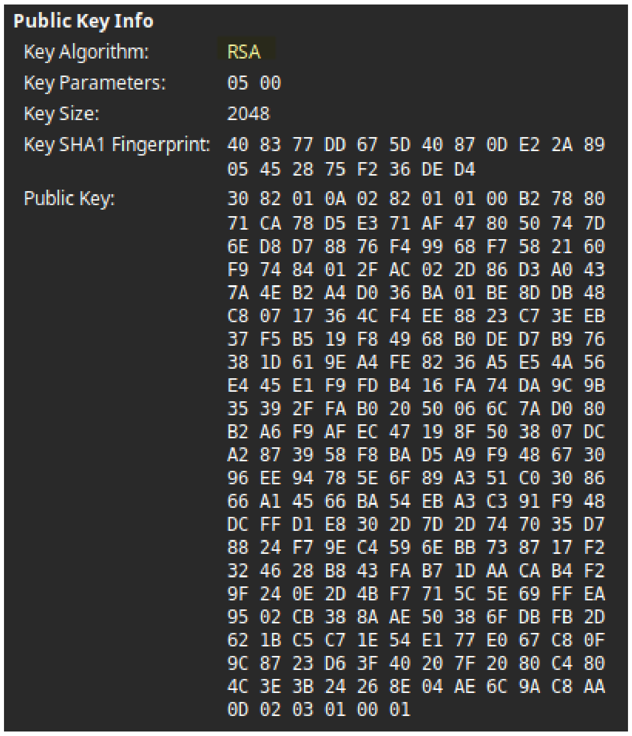
```
Signature
 Signature Algorithm:  1.2.840.113549.1.1.11
 Signature Parameters: 05 00
 Signature:            62 37 42 5C BC 10 B5 3E 8B 2C E9 0C
                       9B 6C 45 E2 07 00 7A F9 C5 58 0B B9
                       08 8C 3E ED B3 25 3C B5 6F 50 E4 CD
                       35 6A A7 93 34 96 32 21 A9 48 44 AB
                       9C ED 3D B4 AA 73 6D E4 7F 16 80 89
                       6C CF 28 03 18 83 47 79 A3 10 7E 30
                       5B AC 3B B0 60 E0 77 D4 08 A6 E1 1D
                       7C 5E C0 BB F9 9A 7B 22 9D A7 00 09
                       7E AC 46 17 83 DC 9C 26 57 99 30 39
                       62 96 8F ED DA DE AA C5 CC 1B 3E CA
                       43 68 6C 57 16 BC D5 0E 20 2E FE FF
                       C2 6A 5D 2E A0 4A 6D 14 58 87 94 E6
                       39 31 5F 7C 73 CB 90 88 6A 84 11 96
                       27 A6 ED D9 81 46 A6 7E A3 72 00 0A
                       52 3E 83 88 07 63 77 89 69 17 0F 39
                       85 D2 AB 08 45 4D D0 51 3A FD 5D 5D
                       37 64 4C 7E 30 B2 55 24 42 9D 36 B0
                       5D 9C 17 81 61 F1 CA F9 10 02 24 AB
                       EB 0D 74 91 8D 7B 45 29 50 39 88 B2
                       A6 89 35 25 1E 14 6A 47 23 31 2F 5C
                       9A FA AD 9A 0E 62 51 A4 2A A9 C4 F9
                       34 9D 21 18
```



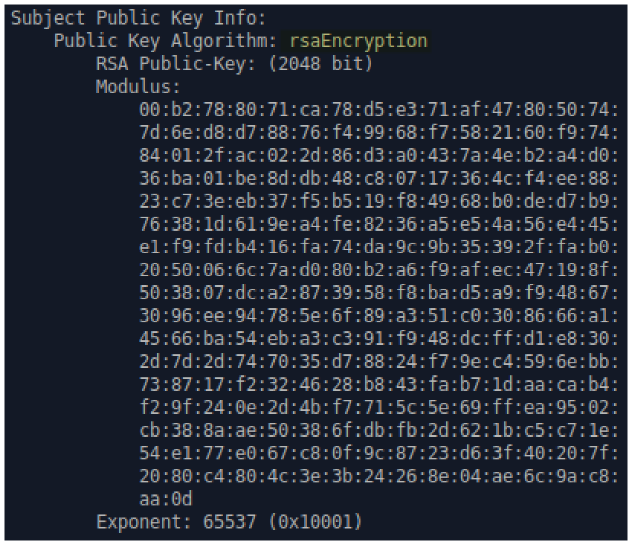
```
Signature Algorithm: sha256WithRSAEncryption
     62:37:42:5c:bc:10:b5:3e:8b:2c:e9:0c:9b:6c:45:e2:07:00:
     7a:f9:c5:58:0b:b9:08:8c:3e:ed:b3:25:3c:b5:6f:50:e4:cd:
     35:6a:a7:93:34:96:32:21:a9:48:44:ab:9c:ed:3d:b4:aa:73:
     6d:e4:7f:16:80:89:6c:cf:28:03:18:83:47:79:a3:10:7e:30:
     5b:ac:3b:b0:60:e0:77:d4:08:a6:e1:1d:7c:5e:c0:bb:f9:9a:
     7b:22:9d:a7:00:09:7e:ac:46:17:83:dc:9c:26:57:99:30:39:
     62:96:8f:ed:da:de:aa:c5:cc:1b:3e:ca:43:68:6c:57:16:bc:
     d5:0e:20:2e:fe:ff:c2:6a:5d:2e:a0:4a:6d:14:58:87:94:e6:
     39:31:5f:7c:73:cb:90:88:6a:84:11:96:27:a6:ed:d9:81:46:
     a6:7e:a3:72:00:0a:52:3e:83:88:07:63:77:89:69:17:0f:39:
     85:d2:ab:08:45:4d:d0:51:3a:fd:5d:5d:37:64:4c:7e:30:b2:
     55:24:42:9d:36:b0:5d:9c:17:81:61:f1:ca:f9:10:02:24:ab:
     eb:0d:74:91:8d:7b:45:29:50:39:88:b2:a6:89:35:25:1e:14:
     6a:47:23:31:2f:5c:9a:fa:ad:9a:0e:62:51:a4:2a:a9:c4:f9:
     34:9d:21:18
```

b. What is the public key cryptography used to create the signature?

RSA



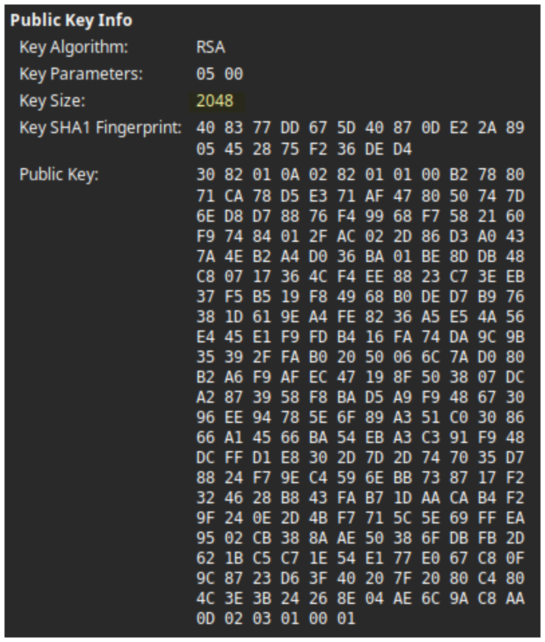
```
Public Key Info
  Key Algorithm:         RSA
  Key Parameters:        05 00
  Key Size:              2048
  Key SHA1 Fingerprint:  40 83 77 DD 67 5D 40 87 0D E2 2A 89
                         05 45 28 75 F2 36 DE D4
  Public Key:            30 82 01 0A 02 82 01 01 00 B2 78 80
                         71 CA 78 D5 E3 71 AF 47 80 50 74 7D
                         6E D8 D7 88 76 F4 99 68 F7 58 21 60
                         F9 74 84 01 2F AC 02 2D 86 D3 A0 43
                         7A 4E B2 A4 D0 36 BA 01 BE 8D DB 48
                         C8 07 17 36 4C F4 EE 88 23 C7 3E EB
                         37 F5 B5 19 F8 49 68 B0 DE D7 B9 76
                         38 1D 61 9E A4 FE 82 36 A5 E5 4A 56
                         E4 45 E1 F9 FD B4 16 FA 74 DA 9C 9B
                         35 39 2F FA B0 20 50 06 6C 7A D0 80
                         B2 A6 F9 AF EC 47 19 8F 50 38 07 DC
                         A2 87 39 58 F8 BA D5 A9 F9 48 67 30
                         96 EE 94 78 5E 6F 89 A3 51 C0 30 86
                         66 A1 45 66 BA 54 EB A3 C3 91 F9 48
                         DC FF D1 E8 30 2D 7D 2D 74 70 35 D7
                         88 24 F7 9E C4 59 6E BB 73 87 17 F2
                         32 46 28 B8 43 FA B7 1D AA CA B4 F2
                         9F 24 0E 2D 4B F7 71 5C 5E 69 FF EA
                         95 02 CB 38 8A AE 50 38 6F DB FB 2D
                         62 1B C5 C7 1E 54 E1 77 E0 67 C8 0F
                         9C 87 23 D6 3F 40 20 7F 20 80 C4 80
                         4C 3E 3B 24 26 8E 04 AE 6C 9A C8 AA
                         0D 02 03 01 00 01
```



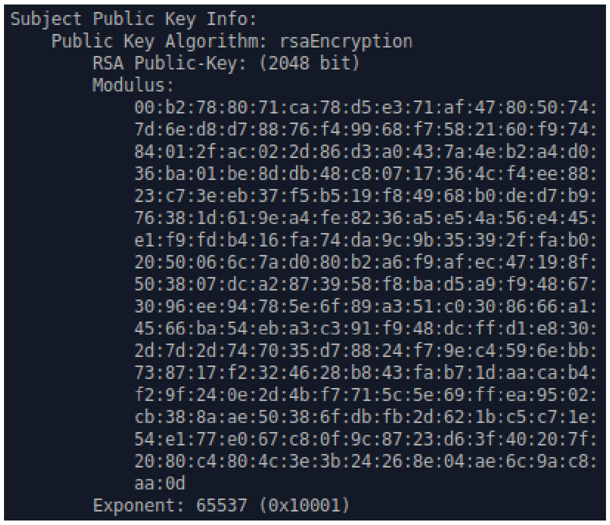
```
Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
        RSA Public-Key: (2048 bit)
        Modulus:
            00:b2:78:80:71:ca:78:d5:e3:71:af:47:80:50:74:
            7d:6e:d8:d7:88:76:f4:99:68:f7:58:21:60:f9:74:
            84:01:2f:ac:02:2d:86:d3:a0:43:7a:4e:b2:a4:d0:
            36:ba:01:be:8d:db:48:c8:07:17:36:4c:f4:ee:88:
            23:c7:3e:eb:37:f5:b5:19:f8:49:68:b0:de:d7:b9:
            76:38:1d:61:9e:a4:fe:82:36:a5:e5:4a:56:e4:45:
            e1:f9:fd:b4:16:fa:74:da:9c:9b:35:39:2f:fa:b0:
            20:50:06:6c:7a:d0:80:b2:a6:f9:af:ec:47:19:8f:
            50:38:07:dc:a2:87:39:58:f8:ba:d5:a9:f9:48:67:
            30:96:ee:94:78:5e:6f:89:a3:51:c0:30:86:66:a1:
            45:66:ba:54:eb:a3:c3:91:f9:48:dc:ff:d1:e8:30:
            2d:7d:2d:74:70:35:d7:88:24:f7:9e:c4:59:6e:bb:
            73:87:17:f2:32:46:28:b8:43:fa:b7:1d:aa:ca:b4:
            f2:9f:24:0e:2d:4b:f7:71:5c:5e:69:ff:ea:95:02:
            cb:38:8a:ae:50:38:6f:db:fb:2d:62:1b:c5:c7:1e:
            54:e1:77:e0:67:c8:0f:9c:87:23:d6:3f:40:20:7f:
            20:80:c4:80:4c:3e:3b:24:26:8e:04:ae:6c:9a:c8:
            aa:0d
        Exponent: 65537 (0x10001)
```

c. What is the size of the public key?

2048



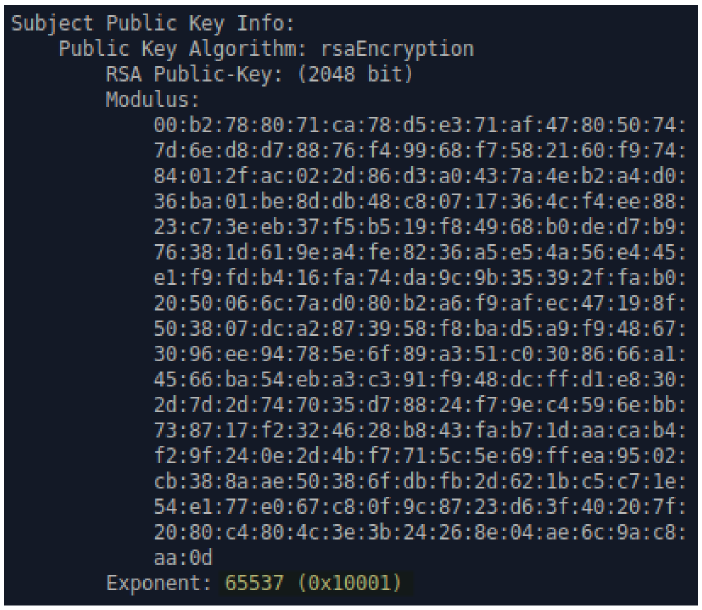
```
Public Key Info
  Key Algorithm:          RSA
  Key Parameters:         05 00
  Key Size:               2048
  Key SHA1 Fingerprint:   40 83 77 DD 67 5D 40 87 0D E2 2A 89
                          05 45 28 75 F2 36 DE D4
  Public Key:             30 82 01 0A 02 82 01 01 00 B2 78 80
                          71 CA 78 D5 E3 71 AF 47 80 50 74 7D
                          6E D8 D7 88 76 F4 99 68 F7 58 21 60
                          F9 74 84 01 2F AC 02 2D 86 D3 A0 43
                          7A 4E B2 A4 D0 36 BA 01 BE 8D DB 48
                          C8 07 17 36 4C F4 EE 88 23 C7 3E EB
                          37 F5 B5 19 F8 49 68 B0 DE D7 B9 76
                          38 1D 61 9E A4 FE 82 36 A5 E5 4A 56
                          E4 45 E1 F9 FD B4 16 FA 74 DA 9C 9B
                          35 39 2F FA B0 20 50 06 6C 7A D0 80
                          B2 A6 F9 AF EC 47 19 8F 50 38 07 DC
                          A2 87 39 58 F8 BA D5 A9 F9 48 67 30
                          96 EE 94 78 5E 6F 89 A3 51 C0 30 86
                          66 A1 45 66 BA 54 EB A3 C3 91 F9 48
                          DC FF D1 E8 30 2D 7D 2D 74 70 35 D7
                          88 24 F7 9E C4 59 6E BB 73 87 17 F2
                          32 46 28 B8 43 FA B7 1D AA CA B4 F2
                          9F 24 0E 2D 4B F7 71 5C 5E 69 FF EA
                          95 02 CB 38 8A AE 50 38 6F DB FB 2D
                          62 1B C5 C7 1E 54 E1 77 E0 67 C8 0F
                          9C 87 23 D6 3F 40 20 7F 20 80 C4 80
                          4C 3E 3B 24 26 8E 04 AE 6C 9A C8 AA
                          0D 02 03 01 00 01
```



```
Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
        RSA Public-Key: (2048 bit)
        Modulus:
            00:b2:78:80:71:ca:78:d5:e3:71:af:47:80:50:74:
            7d:6e:d8:d7:88:76:f4:99:68:f7:58:21:60:f9:74:
            84:01:2f:ac:02:2d:86:d3:a0:43:7a:4e:b2:a4:d0:
            36:ba:01:be:8d:db:48:c8:07:17:36:4c:f4:ee:88:
            23:c7:3e:eb:37:f5:b5:19:f8:49:68:b0:de:d7:b9:
            76:38:1d:61:9e:a4:fe:82:36:a5:e5:4a:56:e4:45:
            e1:f9:fd:b4:16:fa:74:da:9c:9b:35:39:2f:fa:b0:
            20:50:06:6c:7a:d0:80:b2:a6:f9:af:ec:47:19:8f:
            50:38:07:dc:a2:87:39:58:f8:ba:d5:a9:f9:48:67:
            30:96:ee:94:78:5e:6f:89:a3:51:c0:30:86:66:a1:
            45:66:ba:54:eb:a3:c3:91:f9:48:dc:ff:d1:e8:30:
            2d:7d:2d:74:70:35:d7:88:24:f7:9e:c4:59:6e:bb:
            73:87:17:f2:32:46:28:b8:43:fa:b7:1d:aa:ca:b4:
            f2:9f:24:0e:2d:4b:f7:71:5c:5e:69:ff:ea:95:02:
            cb:38:8a:ae:50:38:6f:db:fb:2d:62:1b:c5:c7:1e:
            54:e1:77:e0:67:c8:0f:9c:87:23:d6:3f:40:20:7f:
            20:80:c4:80:4c:3e:3b:24:26:8e:04:ae:6c:9a:c8:
            aa:0d
        Exponent: 65537 (0x10001)
```

d. What is the e used in the public key in the certificate?

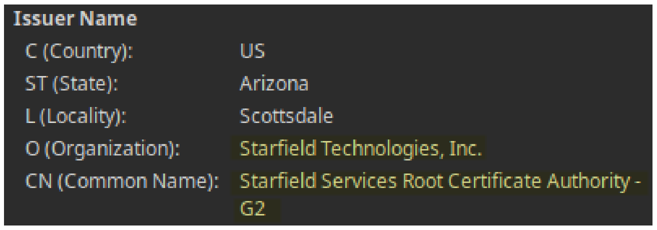
65537 (0x10001)



```
Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
        RSA Public-Key: (2048 bit)
        Modulus:
            00:b2:78:80:71:ca:78:d5:e3:71:af:47:80:50:74:
            7d:6e:d8:d7:88:76:f4:99:68:f7:58:21:60:f9:74:
            84:01:2f:ac:02:2d:86:d3:a0:43:7a:4e:b2:a4:d0:
            36:ba:01:be:8d:db:48:c8:07:17:36:4c:f4:ee:88:
            23:c7:3e:eb:37:f5:b5:19:f8:49:68:b0:de:d7:b9:
            76:38:1d:61:9e:a4:fe:82:36:a5:e5:4a:56:e4:45:
            e1:f9:fd:b4:16:fa:74:da:9c:9b:35:39:2f:fa:b0:
            20:50:06:6c:7a:d0:80:b2:a6:f9:af:ec:47:19:8f:
            50:38:07:dc:a2:87:39:58:f8:ba:d5:a9:f9:48:67:
            30:96:ee:94:78:5e:6f:89:a3:51:c0:30:86:66:a1:
            45:66:ba:54:eb:a3:c3:91:f9:48:dc:ff:d1:e8:30:
            2d:7d:2d:74:70:35:d7:88:24:f7:9e:c4:59:6e:bb:
            73:87:17:f2:32:46:28:b8:43:fa:b7:1d:aa:ca:b4:
            f2:9f:24:0e:2d:4b:f7:71:5c:5e:69:ff:ea:95:02:
            cb:38:8a:ae:50:38:6f:db:fb:2d:62:1b:c5:c7:1e:
            54:e1:77:e0:67:c8:0f:9c:87:23:d6:3f:40:20:7f:
            20:80:c4:80:4c:3e:3b:24:26:8e:04:ae:6c:9a:c8:
            aa:0d
        Exponent: 65537 (0x10001)
```

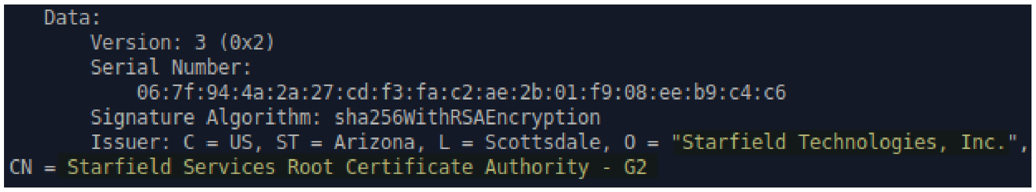
e. Who issued the certificate?

Starfield Services Root Certificate Authority - G2 of Starfield Technologies, Inc.



Issuer Name

C (Country):	US
ST (State):	Arizona
L (Locality):	Scottsdale
O (Organization):	Starfield Technologies, Inc.
CN (Common Name):	Starfield Services Root Certificate Authority - G2



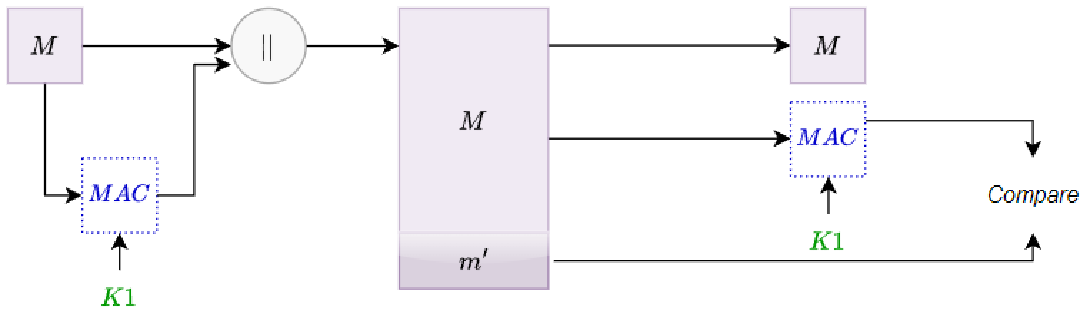
```
    Data:
        Version: 3 (0x2)
        Serial Number:
            06:7f:94:4a:2a:27:cd:f3:fa:c2:ae:2b:01:f9:08:ee:b9:c4:c6
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = US, ST = Arizona, L = Scottsdale, O = "Starfield Technologies, Inc.",
CN = Starfield Services Root Certificate Authority - G2
```

f. Why can you trust the certificate?

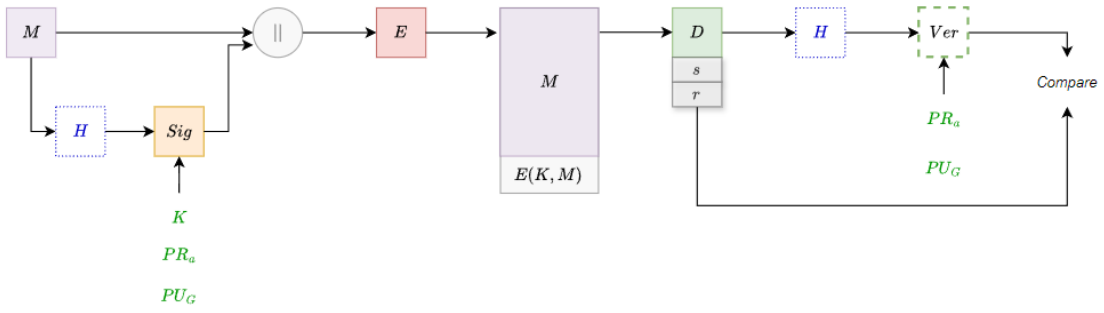
This certificate is trustworthy because it has been issued by a trusted root certificate authority (Starfield Services Root Certificate Authority - G2), it has been signed using RSA encryption, and it is still within its validity timeframe (May 25, 2015 - December 31, 2037)

LAB 7 – (OpenSSL) Create RSA Public/Private Key (4096bits) with Password Protection

1. (8pts) Use symmetric cipher and message authentication code to provide confidentiality, integrity, and authentication. (use the giving cryptographic operations only).



2. (8pts) Use symmetric cipher and digital signature to provide confidentiality, integrity, and authentication. (use the giving cryptographic operations only).



3. (4pts) Compare the two scenarios as designed in a) and b). What are their advantages and limitations?

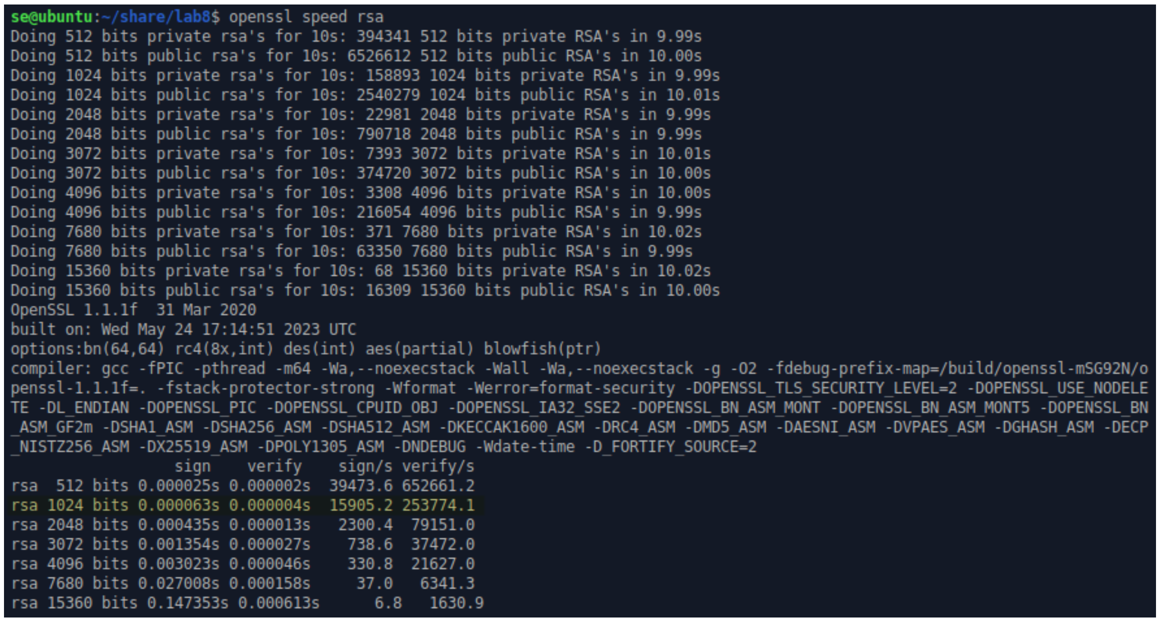
Scenario A, the symmetric cipher and MAC, offers efficient data encryption, ensuring confidentiality while maintaining data integrity and authentication. The addition of a MAC provides an extra layer of security on top of the cipher as it verifies data integrity and source authentication. However, securely managing and distributing the shared secret keys for the symmetric cipher can be challenging. Scenario B, the symmetric cipher with a digital signature, improves security by providing a means to authenticate the sender's identity and prevent data tampering. However, these are susceptible to various attacks, including replay attacks, man-in-the-middle attacks, and key compromise attacks.

LAB 8 – (OpenSSL) Combination of RSA and AES to Encrypt a File

1. (6pts) What is the throughput when you sign and verify a message using a 1024-bit key in RSA? Which one is faster? Why?

RSA 1024-bit Sign throughput:	$15905.2\ operations/s$
RSA 1024-bit Verify throughput:	$253774.1\ operations/s$

The verify throughput is faster than the sign throughput because verifying a signature involves simpler computations with a small public exponent (e.g., 3, 5, 17, 257, or 65537), thus making it quick and inexpensive, whereas generating signatures requires more computationally intensive operations with a private exponent (d).



```
se@ubuntu:~/share/lab8$ openssl speed rsa
Doing 512 bits private rsa's for 10s: 394341 512 bits private RSA's in 9.99s
Doing 512 bits public rsa's for 10s: 6526612 512 bits public RSA's in 10.00s
Doing 1024 bits private rsa's for 10s: 158893 1024 bits private RSA's in 9.99s
Doing 1024 bits public rsa's for 10s: 2540279 1024 bits public RSA's in 10.01s
Doing 2048 bits private rsa's for 10s: 22981 2048 bits private RSA's in 9.99s
Doing 2048 bits public rsa's for 10s: 790718 2048 bits public RSA's in 9.99s
Doing 3072 bits private rsa's for 10s: 7393 3072 bits private RSA's in 10.01s
Doing 3072 bits public rsa's for 10s: 374720 3072 bits public RSA's in 10.00s
Doing 4096 bits private rsa's for 10s: 3308 4096 bits private RSA's in 10.00s
Doing 4096 bits public rsa's for 10s: 216054 4096 bits public RSA's in 9.99s
Doing 7680 bits private rsa's for 10s: 371 7680 bits private RSA's in 10.02s
Doing 7680 bits public rsa's for 10s: 63350 7680 bits public RSA's in 9.99s
Doing 15360 bits private rsa's for 10s: 68 15360 bits private RSA's in 10.02s
Doing 15360 bits public rsa's for 10s: 16309 15360 bits public RSA's in 10.00s
OpenSSL 1.1.1f  31 Mar 2020
built on: Wed May 24 17:14:51 2023 UTC
options:bn(64,64) rc4(8x,int) des(int) aes(partial) blowfish(ptr)
compiler: gcc -fPIC -pthread -m64 -Wa,--noexecstack -Wall -Wa,--noexecstack -g -O2 -fdebug-prefix-map=/build/openssl-mSG92N/o
penssl-1.1.1f=. -fstack-protector-strong -Wformat -Werror=format-security -DOPENSSL_TLS_SECURITY_LEVEL=2 -DOPENSSL_USE_NODELE
TE -DL_ENDIAN -DOPENSSL_PIC -DOPENSSL_CPUID_OBJ -DOPENSSL_IA32_SSE2 -DOPENSSL_BN_ASM_MONT -DOPENSSL_BN_ASM_MONT5 -DOPENSSL_BN
_ASM_GF2m -DSHA1_ASM -DSHA256_ASM -DSHA512_ASM -DKECCAK1600_ASM -DRC4_ASM -DMD5_ASM -DAESNI_ASM -DVPAES_ASM -DGHASH_ASM -DECP
_NISTZ256_ASM -DX25519_ASM -DPOLY1305_ASM -DNDEBUG -Wdate-time -D_FORTIFY_SOURCE=2
                  sign    verify    sign/s verify/s
rsa  512 bits 0.000025s 0.000002s  39473.6 652661.2
rsa 1024 bits 0.000063s 0.000004s  15905.2 253774.1
rsa 2048 bits 0.000435s 0.000013s   2300.4  79151.0
rsa 3072 bits 0.001354s 0.000027s    738.6  37472.0
rsa 4096 bits 0.003023s 0.000046s    330.8  21627.0
rsa 7680 bits 0.027008s 0.000158s     37.0   6341.3
rsa 15360 bits 0.147353s 0.000613s      6.8   1630.9
```

2. (6pts) What is the throughput when you encrypt a 1024-bit message block using DES CBC mode? Compared with a 1024-bit RSA verify throughput, which one is faster? Compared with a 1024-bit RSA sign throughput, which one is faster?

DES CBC 1024-bit throughput:	$\approx 113761.355\ KB/s$ *
RSA 1024-bit Sign throughput:	$15905.200\ operations/s$
RSA 1024-bit Verify throughput:	$253774.100\ operations/s$

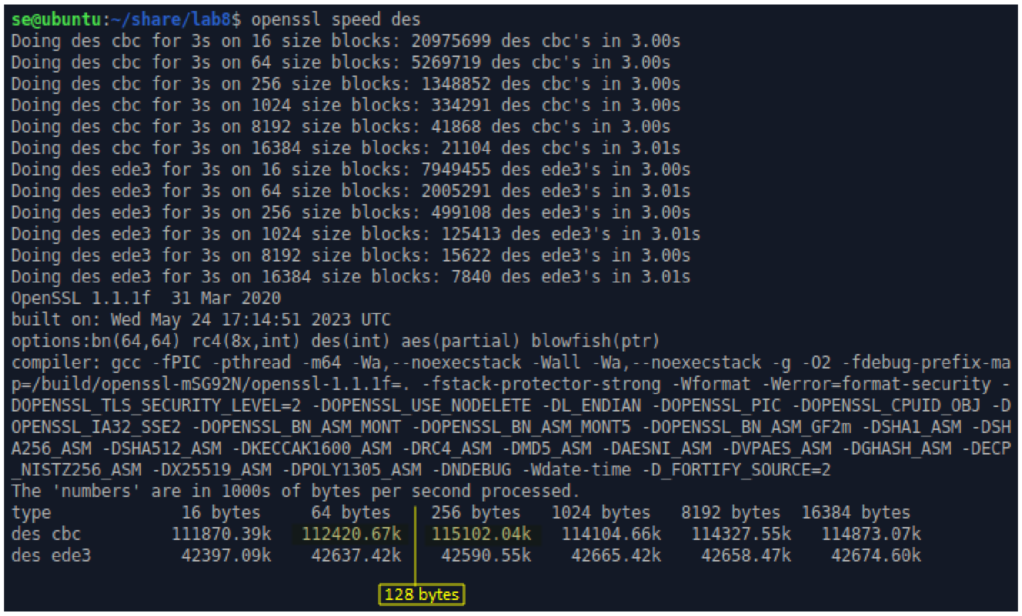
Comparatively, DES CBC encryption is slower than RSA verification but faster than RSA signing for 1024-bit operations.

** While the throughput for 1024 bit is not explicitly stated in the speed test, it can be estimated by finding the average between 64 bytes and 256 bytes:*

$$A = throughput(64bytes) = 112420.67\ \text{KB/s}$$

$$B = throughput(256bytes) = 115102.04\ \text{KB/s}$$

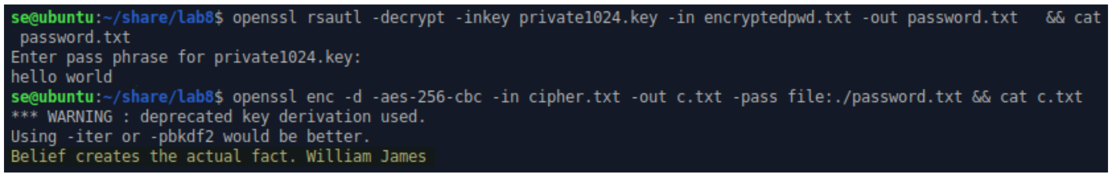
$$C = \ throughput(128bytes) = \frac{A+B}{2} = 113761.355\ \text{KB/s}$$



```
se@ubuntu:~/share/lab8$ openssl speed des
Doing des cbc for 3s on 16 size blocks: 20975699 des cbc's in 3.00s
Doing des cbc for 3s on 64 size blocks: 5269719 des cbc's in 3.00s
Doing des cbc for 3s on 256 size blocks: 1348852 des cbc's in 3.00s
Doing des cbc for 3s on 1024 size blocks: 334291 des cbc's in 3.00s
Doing des cbc for 3s on 8192 size blocks: 41868 des cbc's in 3.00s
Doing des cbc for 3s on 16384 size blocks: 21104 des cbc's in 3.01s
Doing des ede3 for 3s on 16 size blocks: 7949455 des ede3's in 3.00s
Doing des ede3 for 3s on 64 size blocks: 2005291 des ede3's in 3.01s
Doing des ede3 for 3s on 256 size blocks: 499108 des ede3's in 3.00s
Doing des ede3 for 3s on 1024 size blocks: 125413 des ede3's in 3.01s
Doing des ede3 for 3s on 8192 size blocks: 15622 des ede3's in 3.00s
Doing des ede3 for 3s on 16384 size blocks: 7840 des ede3's in 3.01s
OpenSSL 1.1.1f  31 Mar 2020
built on: Wed May 24 17:14:51 2023 UTC
options:bn(64,64) rc4(8x,int) des(int) aes(partial) blowfish(ptr)
compiler: gcc -fPIC -pthread -m64 -Wa,--noexecstack -Wall -Wa,--noexecstack -g -O2 -fdebug-prefix-ma
p=/build/openssl-mSG92N/openssl-1.1.1f=. -fstack-protector-strong -Wformat -Werror=format-security -
DOPENSSL_TLS_SECURITY_LEVEL=2 -DOPENSSL_USE_NODELETE -DL_ENDIAN -DOPENSSL_PIC -DOPENSSL_CPUID_OBJ -D
OPENSSL_IA32_SSE2 -DOPENSSL_BN_ASM_MONT -DOPENSSL_BN_ASM_MONT5 -DOPENSSL_BN_ASM_GF2m -DSHA1_ASM -DSH
A256_ASM -DSHA512_ASM -DKECCAK1600_ASM -DRC4_ASM -DMD5_ASM -DAESNI_ASM -DVPAES_ASM -DGHASH_ASM -DECP
_NISTZ256_ASM -DX25519_ASM -DPOLY1305_ASM -DNDEBUG -Wdate-time -D_FORTIFY_SOURCE=2
The 'numbers' are in 1000s of bytes per second processed.
type             16 bytes     64 bytes    256 bytes   1024 bytes   8192 bytes  16384 bytes
des cbc         111870.39k   112420.67k   115102.04k   114104.66k   114327.55k   114873.07k
des ede3         42397.09k    42637.42k    42590.55k    42665.42k    42658.47k    42674.60k
```

3. (8pts) What is the plaintext you restored from the cipher text?

After decrypting the cipher text with the password obtained from RSA decryption (hello world), the restored plaintext is “Belief creates the actual fact. William James”



```
se@ubuntu:~/share/lab8$ openssl rsautl -decrypt -inkey private1024.key -in encryptedpwd.txt -out password.txt   && cat
 password.txt
Enter pass phrase for private1024.key:
hello world
se@ubuntu:~/share/lab8$ openssl enc -d -aes-256-cbc -in cipher.txt -out c.txt -pass file:./password.txt && cat c.txt
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
Belief creates the actual fact. William James
```