**RSA Assignment**

**Name:** Hoda Gamal Hamouda Ismail

**Code:** 9203673

**Section:** 2

**BN:** 34

**Efficiency Analysis:**

**Figure1: from 28 to 2048 bits**Chart, histogram

Description automatically generatedChart, histogram

Description automatically generated

**Figure2: from 28 to 1024 bits**Chart

Description automatically generated

Chart

Description automatically generated

**Figure3: from 28 to 1024 bits with only these samples (28,32,64,128,256,512,786,1024)**

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

|  |  |
| --- | --- |
| Size of n (bits) | Time of encryption (seconds) |
| 28 | 0.0 |
| 32 | 0.0 |
| 64 | 0.0 |
| 128 | 0.0 |
| 256 | 0.0 |
| 512 | 0.00099945068359375 |
| 786 | 0.0019922256469726562 |
| 1024 | 0.004956245422363281 |

|  |  |
| --- | --- |
| Size of n (bits) | Time of decryption (seconds) |
| 28 | 0.0 |
| 32 | 0.0 |
| 64 | 0.0 |
| 128 | 0.0 |
| 256 | 0.0 |
| 512 | 0.0009999275207519531 |
| 786 | 0.0019943714141845703 |
| 1024 | 0.0040171146392822266 |

**Comment:**

Time of encryption/decryption increases exponentially with increasing of size of n.

This is more obvious when increasing the size of n and increasing the number of samples taken.

but it is still a short time (milliseconds), which makes the algorithm efficient in encryption/decryption.

**Attack Analysis:**

Chart

Description automatically generated

**Comment:**

Time of factorizing n and get d increases exponentially with increasing of size of n.

Sizes until 60 bits, can be broken in a few seconds.

But with increasing the size of n, it takes minutes to hours to get broken.

Which shows us how much RSA algorithm is secure.

And keys of large size such as 512 or 1024 are secure enough to be used if we have computational power and the message will not be useful after days.

**Note:**

The attack may take more time because it’s implemented without a built-in function to factorize n.

This is the function of factorizing n and get private key d:

# function to factoerize n to get p,q then calculate d

def get\_private\_key(n,e):

    # check on primes from 2 to sqrt(n)

    for p in range(2, int(decimal.Decimal(n).sqrt())+1):

        if n % p == 0:

            q = n//p

            break

    # calculate private key using obtaind p,q

    phi=(p-1)\*(q-1)

    d = rsa.mod\_inverse(e,phi)

    return (d)

|  |  |  |  |
| --- | --- | --- | --- |
| Size of n (bits) | Time of factorizing n (seconds) | Size of n (bits) | Time of factorizing n (seconds) |
| 27 | 0.033323049545288086 | 46 | 0.31319713592529297 |
| 28 | 0.0010292530059814453 | 47 | 0.5238082408905029 |
| 29 | 0.0 | 48 | 0.5946090221405029 |
| 30 | 0.0009968280792236328 | 49 | 0.8407526016235352 |
| 31 | 0.001994609832763672 | 50 | 1.2717883586883545 |
| 32 | 0.0019941329956054688 | 51 | 1.3695220947265625 |
| 33 | 0.0029916763305664062 | 52 | 2.924539089202881 |
| 34 | 0.00498652458190918 | 53 | 3.8131909370422363 |
| 35 | 0.005982637405395508 | 54 | 4.522801876068115 |
| 36 | 0.008975744247436523 | 55 | 8.31354546546936 |
| 37 | 0.008977651596069336 | 56 | 10.614036798477173 |
| 38 | 0.017951488494873047 | 57 | 10.497321367263794 |
| 39 | 0.028954744338989258 | 58 | 23.959632873535156 |
| 40 | 0.06283211708068848 | 59 | 30.55836796760559 |
| 41 | 0.05002140998840332 | 60 | 57.029027700424194 |
| 42 | 0.08877754211425781 | 61 | 49.759361267089844 |
| 43 | 0.07183837890625 | 62 | 100.24110698699951 |
| 44 | 0.22040915489196777 | 63 | 307.27827858924866 |
| 45 | 0.26030445098876953 | 64 | 390.27000403404236 |