

RSA Project

Name: Basma Elhoseny

Code:9202381

Sec:1 BN:17

Submitted to:

Dr.Samir Shaheen

Eng Kahled Moataz

Key Generation:

* Choosing e [Public]

.Kraft and Washington [1] have this to say about the choice of e:

Text

Description automatically generated

So I choose e to be 65537

* Choosing p & q [private]

Large non equal primes numbers where there totient is relatively prime to e

Text

Description automatically generated

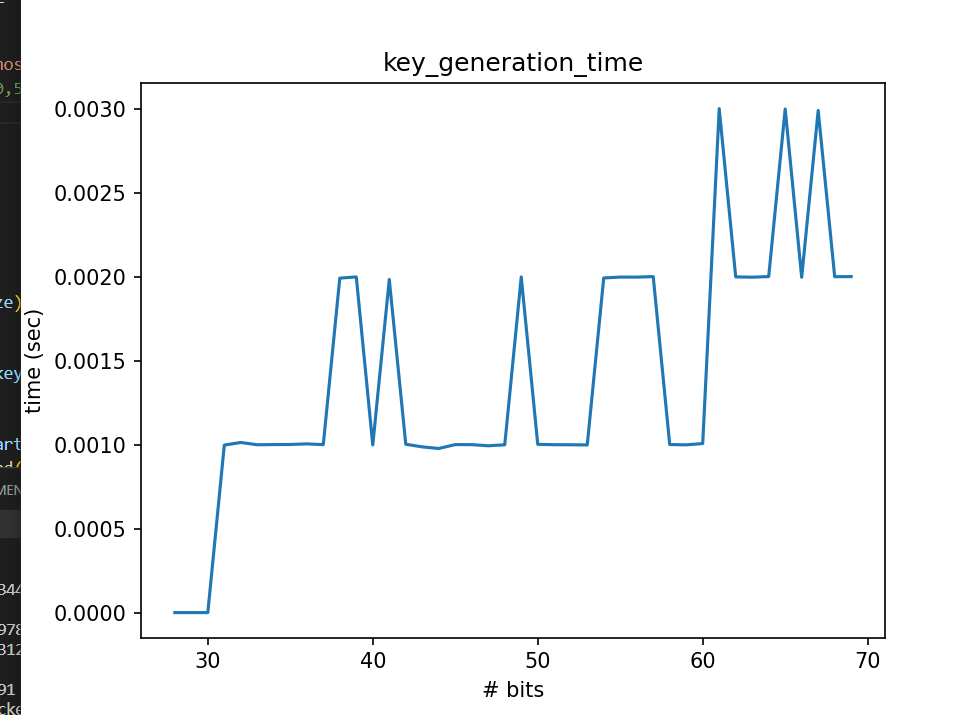
Thus, n=p\*q [Public]

* Computing e [private]

Graphical user interface, text

Description automatically generated

Key Generation Analysis:



Conclusion:

* There are some peaks @ the middles this is dependent on the generated p and q which are random non equal large primes
* It is clear that the Key Genration Algorithm is fast that’s why the RSA is very practical. [more than time for encryption and decryption that’s isn’t the issue because it is performed only once then the users can use the keys for several encryption and decryption which is fast as shown below]

Encryption:

Text

Description automatically generated

**Computed using Fast exponential**

Encryption Analysis:

Chart, histogram

Description automatically generated

Conclusion:

* There are some peaks @ the middles this is dependent on the generated n but they are still In the same range
* It is clear that the Encryption is very fast that’s why the RSA is very practical.

Decryption:

Text

Description automatically generated

**Computed using Fast exponential**

Decryption Analysis:

Chart, histogram

Description automatically generated

Conclusion:

* There are some peaks @ the middles this is dependent on the generated n but they are still In the same range
* It is clear that the Decryption is very fast that’s why the RSA is very practical.

RSA Attack

We will perform Factorization attack for the n to get p and q so we can obtain d 😉

But the problem is that n is very large int so how can we factorize it to its prime factors which will be p and q

I found out that there are 4 techniques for prime factorization.

1. Brute Force Factorization
2. Fermat Factorization
3. Pollard’s p-1 Algorithm
4. Pollard’s Rho Algorithm

So which Algorithm to use ??! 🤔🤔

Let’s Analyze the results then say the reasoning behind these results.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # of bits | Brute Force | Fermat | Pollard’s p-1 | Pollard’s Rho |
| 8 | 0.0 | 0.0 | 0.0 | 0.0 |
| 16 | 0.0 | 0.0 | 0.0 | 0.0 |
| 32 | 0.0240325927734375 | 0.0009987354278564453 | 0.01399850845336914 | 0.0009925365447998047 |
| 64 | Time out | 15.73941946029663 | 45.64180850982666 | 0.16299867630004883 |
| 128 | Time out | Time out | Time out | Time out |

Note: of course these timings are machine dependent and varies from one run to another our aim here is to see the order not the actual value 🤩

Note: here key size (# bits) are for p and q for ex 8 bits means p & q are each 4 bit so that p\*q min=4 and max=8

It is clear that Pollard’s Rho is the best one so I used it for the Analysis below

Attack Analysis:

Due to using Pollard’s Rho Algorithm which is very efficient and very fast 🚘🚘🚘

Chart, histogram

Description automatically generated

Pollard’s Rho Algorithm

To sense so let’s get results by using brute force prime factorization for the Attack I think I succeeded to implement a very efficient RSA breaker

NB: I stopped @ 70 bits because larger than that I took a lot of time and this is logic because we said that this algorithm is very fast for numbers with small factors, but slower in cases where all factors are large ☺

Note: below I took only till 50 bits for the time sack and machine health 😂

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

Run (1) Run(2)

Brute Force Factorization

Another run such that the peak at the end isn’t found this proves that these timing dependent on cause and machine [differs from one run to another]

**It is clear that is very slow compared to the Pollard’s Rho Algorithm**

Chart

Description automatically generated with medium confidence

Results Analysis for some # of bits:

Caution: I took care of what is kept private and what is public here I only print the e and p and q which are only know to the owner so that we only visualize the numbers In the part if chat room these are known only to the owner not even logged in anywhere

Key size 8,10,16,20

NB 📝: These sizes aren’t suitable here because n < message bucket(5 letters ➡ max 5 spaces 😎) But here we are just Testing the Break Analysis to see how key size affect the breaking item

I.e. this key size isn’t suitable for our application here 😐

Text

Description automatically generated

Text

Description automatically generated

So what is the min value of n ??!

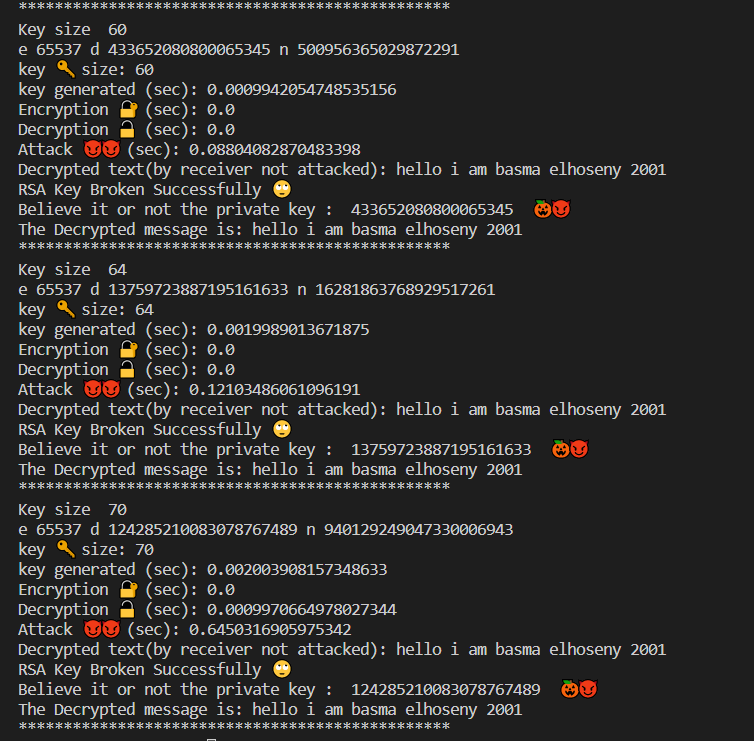
Max message bucket will be 5 spaces = 69343956)10 = 100 0010 0010 0001 1010 1101 0100)2

27 bits, so the min acceptable n >69343956

Good Key Sizes 32-40-50-60-65-70

Text

Description automatically generated



Video:

<https://drive.google.com/drive/folders/1y--vq5CDvzyUk2_BfM4KPEUN__YGvTKA?usp=share_link>

References:

<https://www.johndcook.com/blog/2018/12/12/rsa-exponent/#:~:text=A%20somewhat%20surprising%20detail%20of,number%2C%20specifically%20e%20%3D%2065537>.

<https://www.nku.edu/~christensen/Mathematical%20attack%20on%20RSA.pdf>