

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

Encryption:

Text

Description automatically generated

**Computed using Fast exponential**

Decryption:

Text

Description automatically generated

**Computed using Fast exponential**

Analysis

Text

Description automatically generated

Text

Description automatically generated

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 😐

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

Text

Description automatically generatedText

Description automatically generated

Conclusion:

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

Results:

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 😐

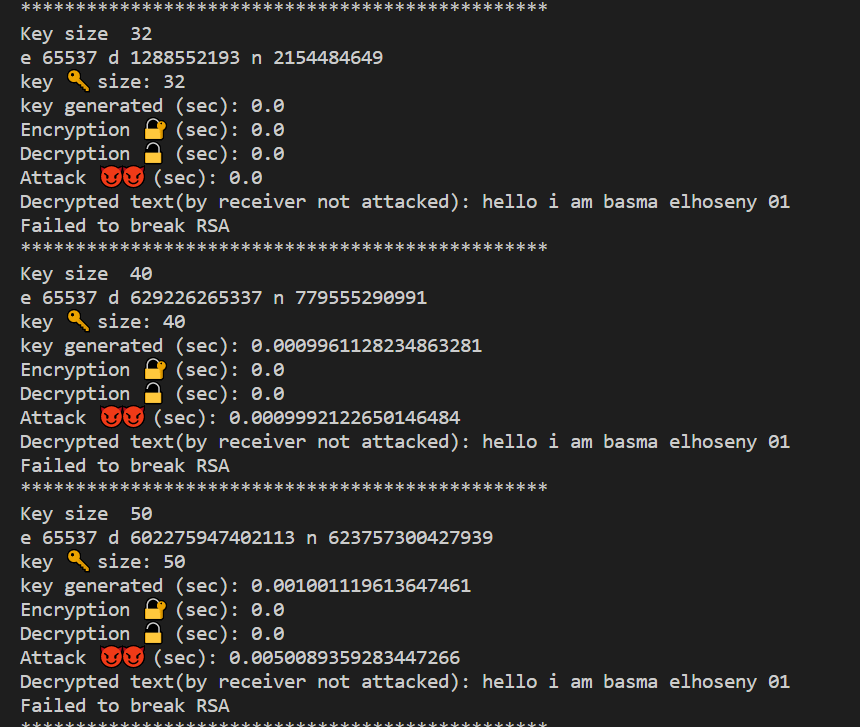
Text

Description automatically generated

Text

Description automatically generated

Key size 32,40,50



Key size 60,64,70

Text

Description automatically generated

Conclusion:

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>