# RSA

## Security Project 1

Fall 2018

Submitted to

Dr. Samir Shahin

Eng. Dina ElReedy

Team Members:

| Name | Email | Section | Bench Number |
|---|---|---|---|
| Omar Osama | omarosamasobeih@yahoo.com | 1 | 32 |
| Mina Magdy | mina_mego5@yahoo.com | 2 | 23 |

## Introduction:

RSA (Rivest–Shamir–Adleman) is one of the first public-key cryptosystems and is widely used for secure data transmission. In such a cryptosystem, the encryption key is public and it is different from the decryption key which is kept secret (private). In RSA, this asymmetry is based on the practical difficulty of the factorization of the product of two large prime numbers, the "factoring problem". The acronym RSA is made of the initial letters of the surnames of Ron Rivest, Adi Shamir, and Leonard Adleman, who first publicly described the algorithm in 1978. Clifford Cocks, an English mathematician working for the British intelligence agency Government Communications Headquarters (GCHQ), had developed an equivalent system in 1973, but this was not declassified until 1997.

## Language and tools:

Implemented RSA in Python using package management system pip and JetBrains' python IDE PyCharm

## Main Components:

1- Generator: module responsible for public and private key generation

2- Property Owner: module responsible for message decryption represents key owner

3- Communicating Party: module responsible for message encryption represents message sender

4- Attacker: module that implements 2 different attacking techniques (brute force – chosen cypher)

## Generator:

### Key generation steps

1- Assign ***e*** = 2^16 + 1 (standard)

2- Randomly generate p, q such that they are primes (use Miller test to check primality) and the Euler ***totient*** of their product and ***e*** are coprime

3- Compute ***n*** = ***p*** * ***q***

4- Compute ***totient*** = (***p*** – 1) * (***q*** – 1)

5- Compute ***d*** = ***e*** ^ -1 mod ***totient***

6- public key = (***e***, ***n***) and private key = (***d***, ***n***)

## Property Owner:

### Message decryption steps

1- Take the cypher text ***c*** as an input

2- Retrieve initial message ***m*** using private key = (***d***, ***n***)

Compute ***m = c*** ^ ***d*** mod ***n***

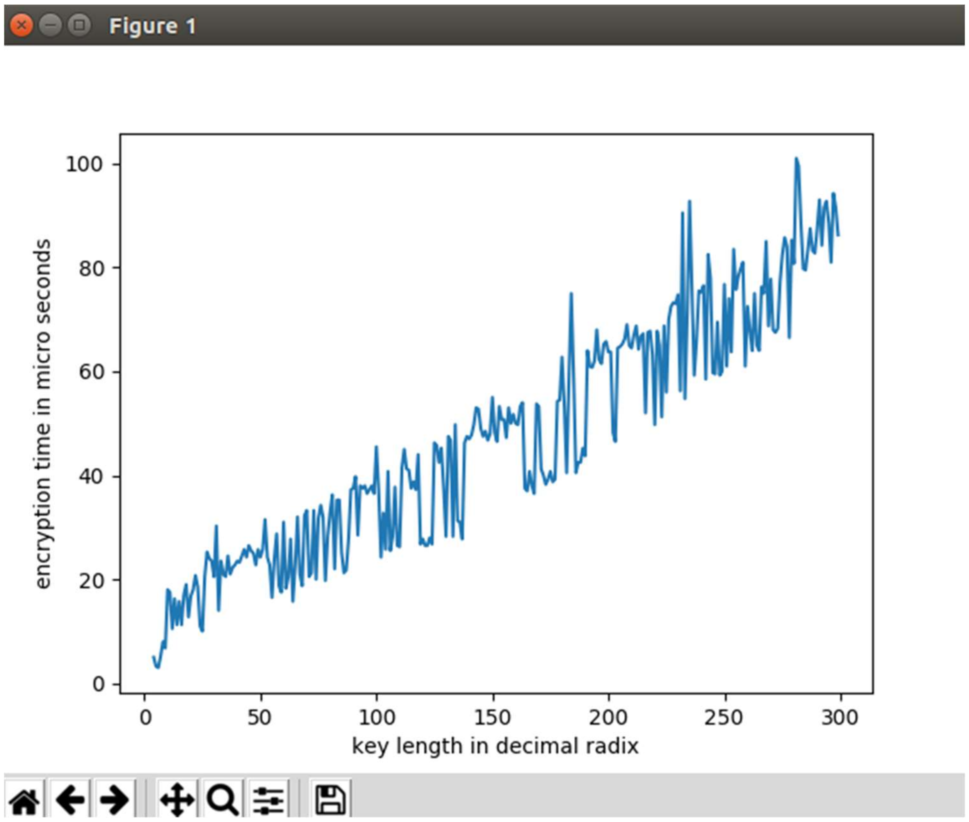
## Communicating Party:

### Message encryption steps

1- Take the input message ***m*** as an input

2- Generate cypher text ***c*** using public key = (***e***, ***n***)

Compute ***c = m*** ^ ***e*** mod ***n***



As shown in the figure the encryption time is directly proportional to the key length

Encryption time α key length

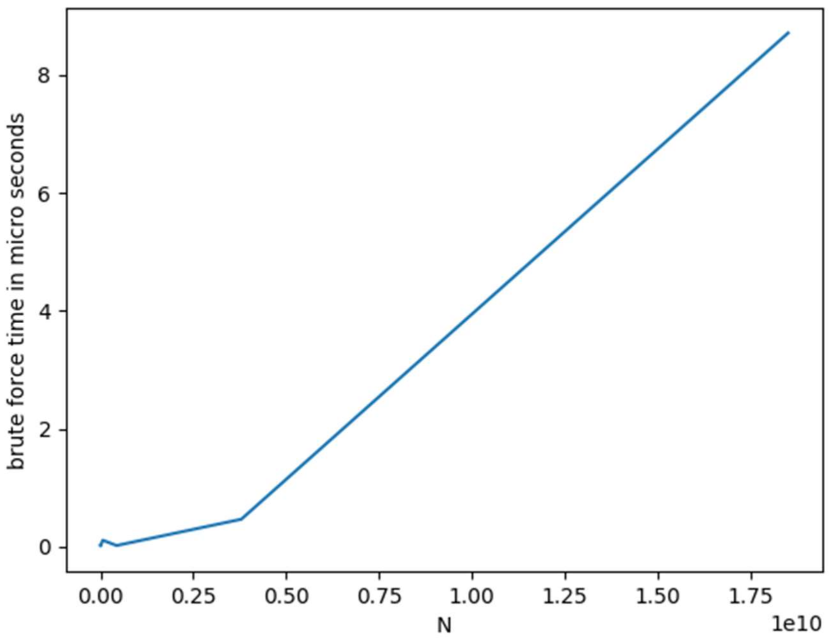
## Attacker:

### Brute force attack steps

1- Use prime factorization to get ***p***, ***q***

2- Calculate ***totient*** = (***p*** – 1) * (***q*** – 1)

3- Compute private key ***d*** = ***e*** ^ -1 mod ***totient***



As shown in the figure the brute force attack time is directly proportional to the value of ***N***

brute force attack time ***α N***

### Chosen cypher attack steps

1- Take cyphered message ***c*** as an input

2- Randomly generate ***r*** such that greatest common divisor of ***n*** and ***r*** = 1

3- Compute chosen cypher = ((***r*** ^ ***e*** mod ***n***) * cyphered message) mod ***n***

4- Deceive the property owner and have him sign the chosen cypher to get signed message

5- Retrieve initial message ***m***

Compute ***m*** = (signed message * (***r*** ^ ***-1*** mod ***n***)) mod ***n***