RSA Implementation

Name: Abhiroop Mukherjee

Enrolment Number: 510519109

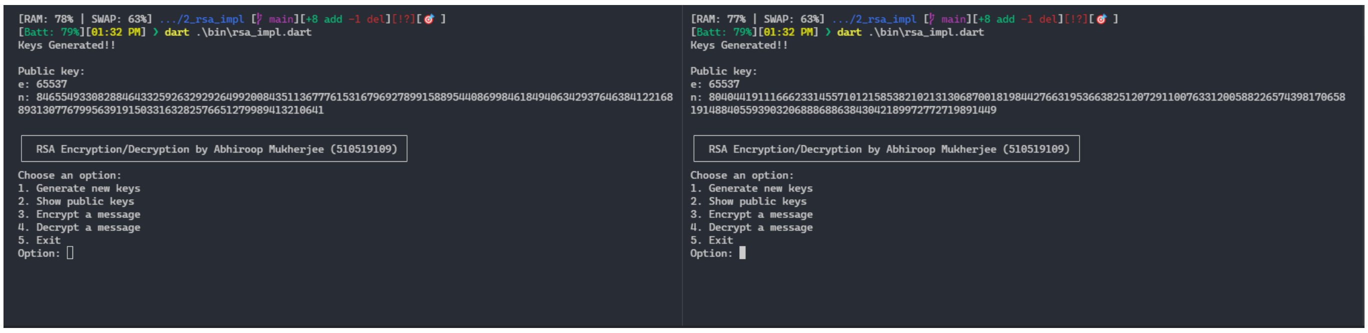
G-Suite ID: 510519109.abhirup@students.iiests.ac.in

Implementation Info

1. Key Size = 256 Bits
2. Used dart's `BigInt` to store the big numbers
3. Implemented all the mathematical algorithm which can be found here

Demo

- Consider the left terminal as sender and right terminal as receiver

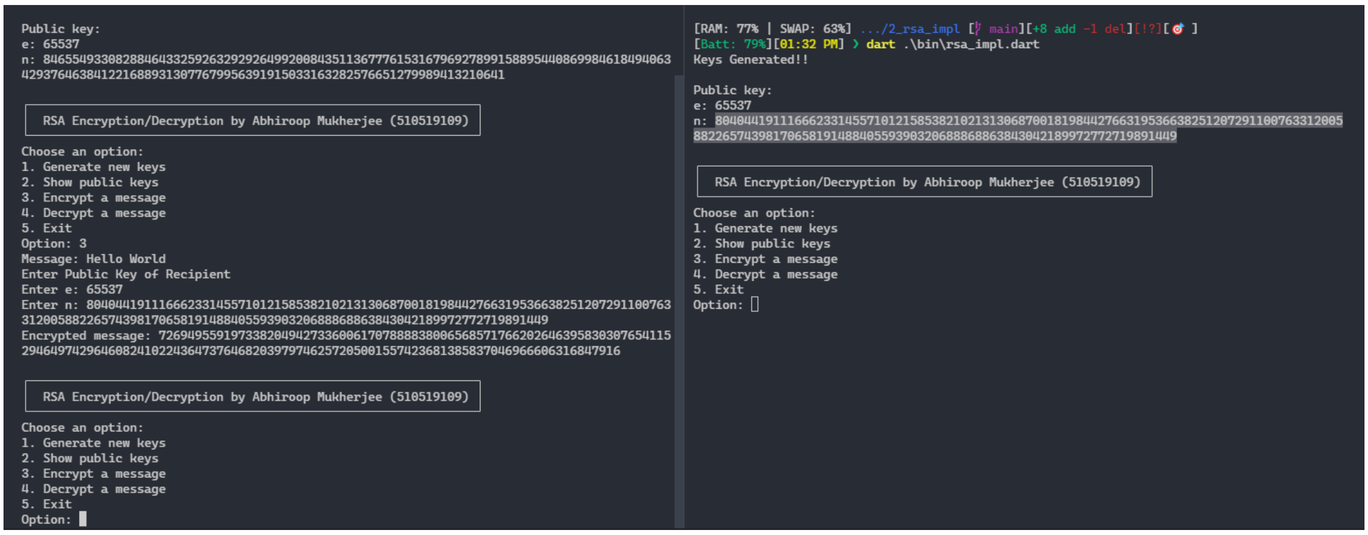


```
[RAM: 78% | SWAP: 63%] .../2_rsa_impl [ main][+8 add -1 del][!?][🎯 ]
[Batt: 79%][01:32 PM] ❯ dart .\bin\rsa_impl.dart
Keys Generated!!

Public key:
e: 65537
n: 84655493308288464332592632929264992008435113677761531679692789915889544086998461849406342937646384122168
89313077679956391915033163282576651279989413210641

 RSA Encryption/Decryption by Abhiroop Mukherjee (510519109)

Choose an option:
1. Generate new keys
2. Show public keys
3. Encrypt a message
4. Decrypt a message
5. Exit
Option:
```

```
[RAM: 77% | SWAP: 63%] .../2_rsa_impl [ main][+8 add -1 del][!?][🎯 ]
[Batt: 79%][01:32 PM] ❯ dart .\bin\rsa_impl.dart
Keys Generated!!

Public key:
e: 65537
n: 80404419111666233145571012158538210213130687001819844276631953663825120729110076331200588226574398170658
19148840559390320688868863843042189972772719891449

 RSA Encryption/Decryption by Abhiroop Mukherjee (510519109)

Choose an option:
1. Generate new keys
2. Show public keys
3. Encrypt a message
4. Decrypt a message
5. Exit
Option:
```

- We see that terminals have different public keys, we are focussed in the public keys of receiver so Receiver Public Key:
  - e: `65537`
  - n: `8040441911166623314557101215853821021313068700181984427663195366382512072911007633120058822657439817065819148840559390320688868863843042189972772719891449`

- Now let's encrypt 'Hello World' using receiver's public key

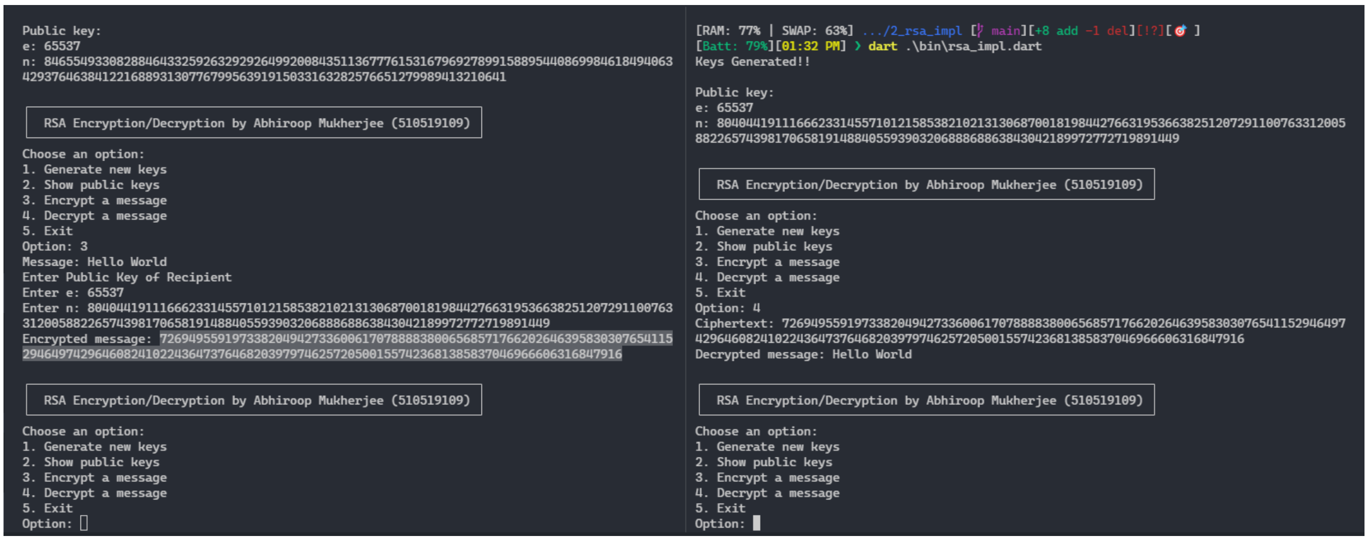


```
Public key:
e: 65537
n: 84655493308288464332592632929264992008435113677761531679692789915889544086998461849406З
4293764638412216889313077679956391915033163282576651279989413210641

 RSA Encryption/Decryption by Abhiroop Mukherjee (510519109)

Choose an option:
1. Generate new keys
2. Show public keys
3. Encrypt a message
4. Decrypt a message
5. Exit
Option: 3
Message: Hello World
Enter Public Key of Recipient
Enter e: 65537
Enter n: 804044191116662331455710121585382102131306870018198442766319536638251207291100763
3120058822657439817065819148840559390320688868863843042189972772719891449
Encrypted message: 72694955919733820494273360061707888838006568571766202646395830307654115
29464974296460824102243647376468203979746257205001557423681385837046966606316847916

 RSA Encryption/Decryption by Abhiroop Mukherjee (510519109)

Choose an option:
1. Generate new keys
2. Show public keys
3. Encrypt a message
4. Decrypt a message
5. Exit
Option: █
```

```
[RAM: 77% | SWAP: 63%] .../2_rsa_impl [ main][+8 add -1 del][!?][🎯 ]
[Batt: 79%][01:32 PM] ❯ dart .\bin\rsa_impl.dart
Keys Generated!!

Public key:
e: 65537
n: 804044191116662331455710121585382102131306870018198442766319536638251207291100763312005
8822657439817065819148840559390320688868863843042189972772719891449

 RSA Encryption/Decryption by Abhiroop Mukherjee (510519109)

Choose an option:
1. Generate new keys
2. Show public keys
3. Encrypt a message
4. Decrypt a message
5. Exit
Option: ▯
```

- we got the ciphertext as `72694955919733820494273360061707888838006568571766202646395830307654115294649742964 60824102243647376468203979746257205001557423681385837046966606316847916`

- now lets decrypt the ciphertext using receiver's private key



```
Public key:
e: 65537
n: 846554933082884643325926329292649920084351136777615316796927899158895440869984618494063
4293764638412216889313077679956391915033163282576651279989413210641

 RSA Encryption/Decryption by Abhiroop Mukherjee (510519109)

Choose an option:
1. Generate new keys
2. Show public keys
3. Encrypt a message
4. Decrypt a message
5. Exit
Option: 3
Message: Hello World
Enter Public Key of Recipient
Enter e: 65537
Enter n: 804044191116662331455710121585382102131306870018198442766319536638251207291100763
3120058822657439817065819148840559390320688868863843042189972772719891449
Encrypted message: 72694955919733820494273360061707888838006568571766202646395830307654115
29464974296460824102243647376468203979746257205001557423681385837046966606316847916

 RSA Encryption/Decryption by Abhiroop Mukherjee (510519109)

Choose an option:
1. Generate new keys
2. Show public keys
3. Encrypt a message
4. Decrypt a message
5. Exit
Option: ▯
```

```
[RAM: 77% | SWAP: 63%] .../2_rsa_impl [ main][+8 add -1 del][!?][🎯 ]
[Batt: 79%][01:32 PM] ❯ dart .\bin\rsa_impl.dart
Keys Generated!!

Public key:
e: 65537
n: 804044191116662331455710121585382102131306870018198442766319536638251207291100763312005
8822657439817065819148840559390320688868863843042189972772719891449

 RSA Encryption/Decryption by Abhiroop Mukherjee (510519109)

Choose an option:
1. Generate new keys
2. Show public keys
3. Encrypt a message
4. Decrypt a message
5. Exit
Option: 4
Ciphertext: 726949559197338204942733600617078888380065685717662026463958303076541152946497
4296460824102243647376468203979746257205001557423681385837046966606316847916
Decrypted message: Hello World

 RSA Encryption/Decryption by Abhiroop Mukherjee (510519109)

Choose an option:
1. Generate new keys
2. Show public keys
3. Encrypt a message
4. Decrypt a message
5. Exit
Option: █
```

- we can see that the exact plaintext is decrypted from the receiver side