Cryptography
MET CS789

Team Project Report

Department of Computer Sciences

Professor: Geoffrey A Pascoe

Table of Contents

Team Members

Sri Nikhil Reddy Gudibandi gsnr@bu.edu

Anup Sindagi anups@bu.edu

Proteek Bose proby@bu.edu

Project Overview

Our Team project focuses on implementing and demonstrating the ElGamal and RSA encryption algorithms. Our team, consisting of 4 members, has developed a Python-based application that allows users to perform encryption and decryption tasks as well as simulate eavesdropping (Eve) for both ElGamal and RSA algorithms.

Code Structure

Project/root Folder

- Controller.py: Main function to initiate the application
- elGamalAlgorithm.py: Implements the ElGamal encryption system including roles of Alice, Bob, and Eve.
- rsaAlgorithm.py: Contains the implementation of the RSA encryption system.

Util Folder

- factorizationAlgorithm_PR.py: Provides functions for number factorization, crucial for cryptographic computations.
- keyGenerator_BSGS.py: Includes the Baby-step Giant-step algorithm for logarithm problems in cryptography.
- modularArithmetic.py: Contains essential functions for modular arithmetic operations.
- primalityTest_MR.py: Implements the Miller-Rabin primality test for identifying prime numbers.
- pseudorandomGenerator_BBS.py: Houses the Blum-Blum-Shub pseudorandom number generator.
- pseudorandomGenerator_NR.py: Implements the Naor-Reingold pseudorandom number generator.

Tests Folder

- test_factorizationAlgorithm_PR.py
- test_keyGenerator_BSGS.py
- test_modularArithmetic.py
- test_primalityTest_MR.py:
- test_pseudorandomGenerator_BBS.py
- test_pseudorandomGenerator_NR.py

Deliverables: Cryptography

The project is available on GitHub:
proteekbose/team_Elgamal-RSA (github.com)
https://github.com/proteekbose/team_Elgamal-RSA

Algorithm Implementation

ElGamal Encryption

Alice's Role (Encryption Process)

- **Public Information Generation:**
  - Alice generates five prime numbers using the Miller-Rabin primality test and selects the largest prime, denoted as **p**.
  - She then finds a primitive root of **p**, named **b**.
  - Alice chooses a secret number **r** (either provided or randomly generated).
  - She calculates **br**, which is **b** raised to the power of **r** modulo **p**, and shares **p**, **b**, and **br** as public information.
- **Message Encryption:**
  - To encrypt a message:
    - Alice inputs a plaintext message, **message**.
    - She inputs Bob's public key, **bl**, and a generator, **g**.
    - Alice then calculates two components of the ciphertext, **c1** and **c2**, using ElGamal encryption:
      - **c1** is **g** raised to the power of a random number **k** modulo **p**.
      - **c2** is the product of **message** and **h** (Bob's public key) raised to the power of **k**, modulo **p**.
    - The encrypted message is the tuple **(c1, c2)**.

Bob's Role (Decryption Process)

- **Public Information Generation:**
  - Bob, similar to Alice, generates his own public information by selecting a large prime **p**, a primitive root **b**, and a secret number **r**.
  - He calculates his public key **br** and shares it along with **p** and **b**.
- **Message Decryption:**
  - To decrypt a message received from Alice:
    - Bob inputs the two components of the ciphertext, **c1** and **c2**.
    - He uses his secret key **x** and the prime number **p** to decrypt the message.
    - The decryption involves calculating **s** (which is **c1** raised to the power of **x**, modulo **p**) and its multiplicative inverse **s_inv**.
    - The decrypted message is calculated as the product of **c2** and **s_inv**, modulo **p**.

Eve's Role (Eavesdropping)

- **Eavesdropping and Decryption:**
  - Eve intercepts the encrypted message **(c1, c2)** and also knows the public information **p**, **b**, **br**, and **bl**.
  - She attempts to decrypt the message without the private key:
    - Eve uses the Baby-Step Giant-Step algorithm to find either Alice's or Bob's secret number.
    - Assuming she finds Bob's secret number **l**, she then uses it to decrypt the message in the same way Bob would.
    - Eve calculates **s** (using **c1** and **l**), finds its multiplicative inverse **s_inv**, and then calculates the decrypted message as the product of **c2** and **s_inv**, modulo **p**.

Choose your Cipher algorithm:

Press (1) for RSA
Press (2) for El-Gamal
Press (0) to Exit
2

Welcome to EL-GAMAL crypto system

Choose the role which you want to play:

Press (1) for Alice
Press (2) for Bob
Press (3) for Eve
Press (4) for COMPLETE DEMO
Press (0) to Exit
4

🖋️ Starting EL-GAMAL DEMONSTRATION... 🖋️
🎲 Selecting Random Generator: Blum-Blum-Shub

🔑 Alice and Bob agree on Prime Number: 16525603 and Generator: 8789254

🔒 Alice's Secret Number: 2345641 ➡️ Public Key: 10302720
🔒 Bob's Secret Number: 12721789 ➡️ Public Key: 9434458

⚙️ Generating Message...

🔐 Encrypted Message: (c1: 15794331, c2: 1055362)
✉️ Alice sends encrypted message: (c1: 15794331, c2: 1055362) to Bob
🔓 Decrypted Message: 1055068
📬 Bob decrypts the message: 1055068

🕵️ Eve intercepts the message and attempts to decrypt it...
🔓 Decrypted Message: 1055068
Eve's cracked secret numbers: 2345641, 12721789
🔓 Eve decrypts the message: 1055068

🌈 EL-GAMAL DEMONSTRATION COMPLETED SUCCESSFULLY! 🎉

Encryption (Alice)

Implementation

In the RSA context, Alice's role is primarily to encrypt a message. This is implemented in the Encrypt function. The process involves:

1. Key Generation: Alice uses a public key composed of a modulus and an exponent. The modulus is typically a product of two large prime numbers, and the exponent is a number that is relatively prime to the totient of the modulus.
2. Message Encryption: Alice encrypts her message using the public key. The encryption process involves raising the message to the power of the public exponent and then taking the modulus.

Process

- Input: The plaintext messages.
- Output: The ciphertext, which is calculated using a lambda function calculate_ciphertext that calls ma.fast_exponent to perform the encryption.

Decryption (Bob)

Implementation

Bob's role in the RSA process is to decrypt the message that he receives from Alice. This is implemented in the Decrypt function.

Process

- Key Generation: Bob uses the private key for decryption. The private key is typically derived from the same prime numbers used to generate the public key.
- Message Decryption: Bob decrypts the ciphertext using his private key. The decryption process involves raising the ciphertext to the power of the private exponent and then taking the modulus.

Eavesdropping (Eve)

Implementation

Eve's role as an eavesdropper is implemented to simulate an external entity attempting to decrypt the communication between Alice and Bob without access to the private key. This is done in the Decode function, which attempts to crack the encryption.

Process

- Intercepting Communication: Eve intercepts the encrypted message and Alice's public key.
- Cracking the Private Key: Eve attempts to factorize the modulus to find its prime factors. Using these factors, she calculates the totient and then the private key.
- Decrypting the Message: Using the cracked private key, Eve decrypts the message by performing the same steps as Bob in the decryption process.

Choose your Cipher algorithm:
Press (1) for RSA
Press (2) for El-Gamal
Press (0) to Exit
1

Welcome to RSA crypto system

Choose the actions that you want to take:

Press (1) to Encrypt
Press (2) to Decrypt
Press (3) to Crack
Press (4) to Generate Keys
Press (5) for COMPLETE DEMO
Press (0) to Exit

Please select a function: 5

🚀 Starting RSA Demonstration...
🔐 Generating random keys...

👩 Alice's public key: (modulus: 29872450479757, exponent: 14898640488357)
🔑 Alice's private key: (modulus: 29872450479757, exponent: 5729604034733)

👨 Bob wants to send message '10046951654682' to Alice.
🔒 Bob encrypts the message using Alice's public key.
💬 The encrypted ciphertext is: 17735848996268

📩 Alice receives the encrypted message and decrypts it with her private key.
🔓 She gets the plaintext: 10046951654682

🦹♀ Eve also knows the encrypted message sent to Alice;
🔨 She attempts to crack it without the private key.
🧮 Calculating the factors of the modulus, Eve finds 3301481 and 9048197
These are prime factors of the modulus 29872450479757
🔓 Eve now can find Alice's private key (modulus: 29872450479757, exponent: 5729604034733)
🦹♀ Eve decrypts the message: 10046951654682

✅ RSA Demonstration Finished.

Usage and Manual

Installation Requirements

Runtimes and Libraries

Before running the **CipherMachine** project, ensure that you have the following installed:

- Python 3.6 or higher: Download Python
- An IDE that supports Python (Recommended: PyCharm): Download PyCharm

No additional libraries outside the Python Standard Library are required for this project.

Project Setup

1. **Download the Project**: Download the **CipherMachine** folder from the provided source.
2. **Import into PyCharm**:
    - Open PyCharm IDE.
    - Select **File** > **Open**.
    - Navigate to and select the downloaded **CipherMachine** folder.
    - Click **Open** to load the project.

Running the Algorithms

Using the Controller

1. **Open controller.py**: In PyCharm, navigate to the **controller.py** file in the project structure.
2. **Run the Script**: Right-click on the **controller.py** file and select **Run 'controller.py'**.

Available Functions

The **Cipher-System Application** project includes the following functions:

- (1) Encryption
- (2) Decryption
- (3) Decryption without a private key
- (4) Key Generation
- (5) Complete DEMO

Follow the on-screen prompts to select and execute these functions.

Note: The default bit size for the random number generator is 24 bits. Increasing the bit size will lead to longer calculation times.

Executing Unit Tests

Running Test Cases

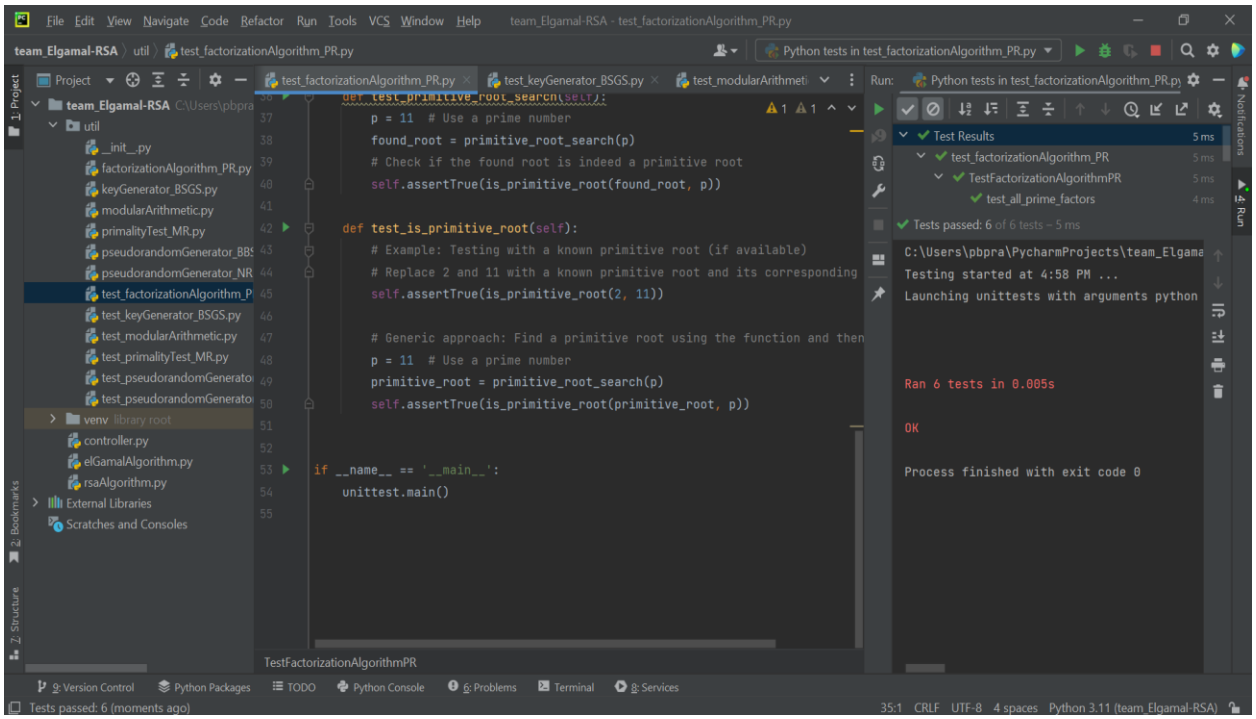
1. **Navigate to the util folder**: In PyCharm, open the **util** folder within the project structure.
2. **Running Individual Tests**:
    - To run a specific test, open the desired test file (e.g., **test_modularArithmetic.py**).
    - Right-click on the file in PyCharm and select **Run 'test_modularArithmetic'**.

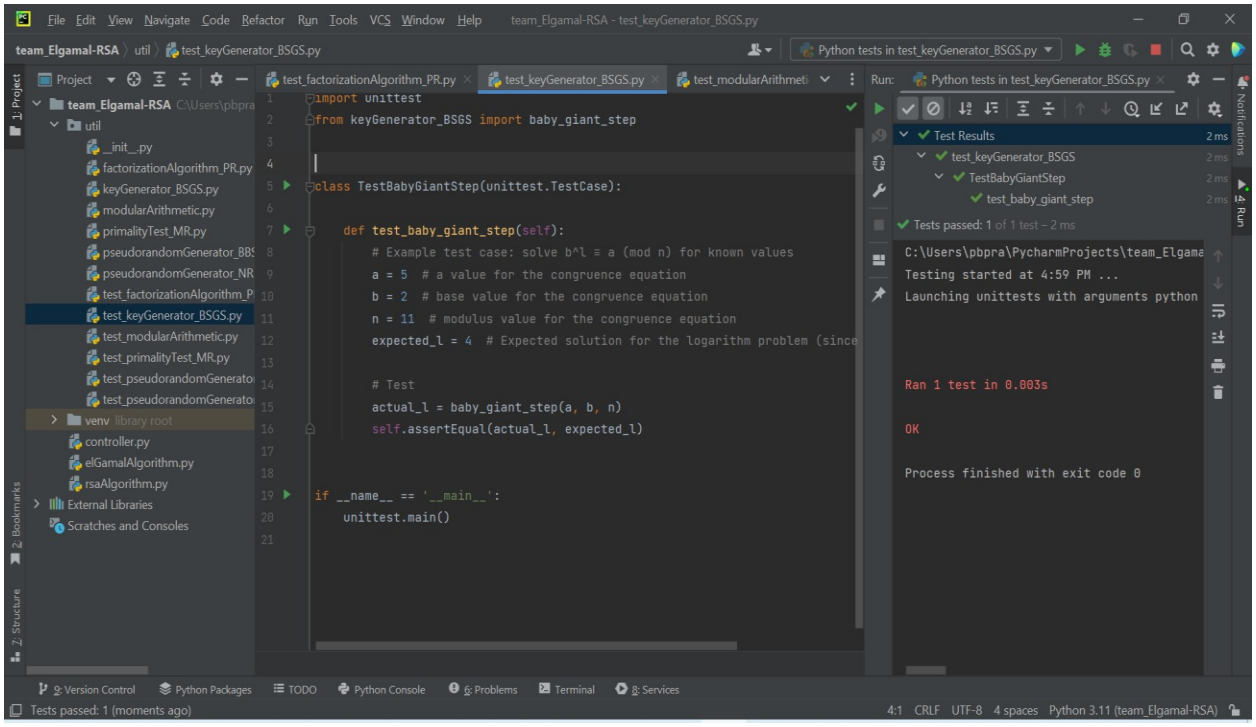
Test Coverage

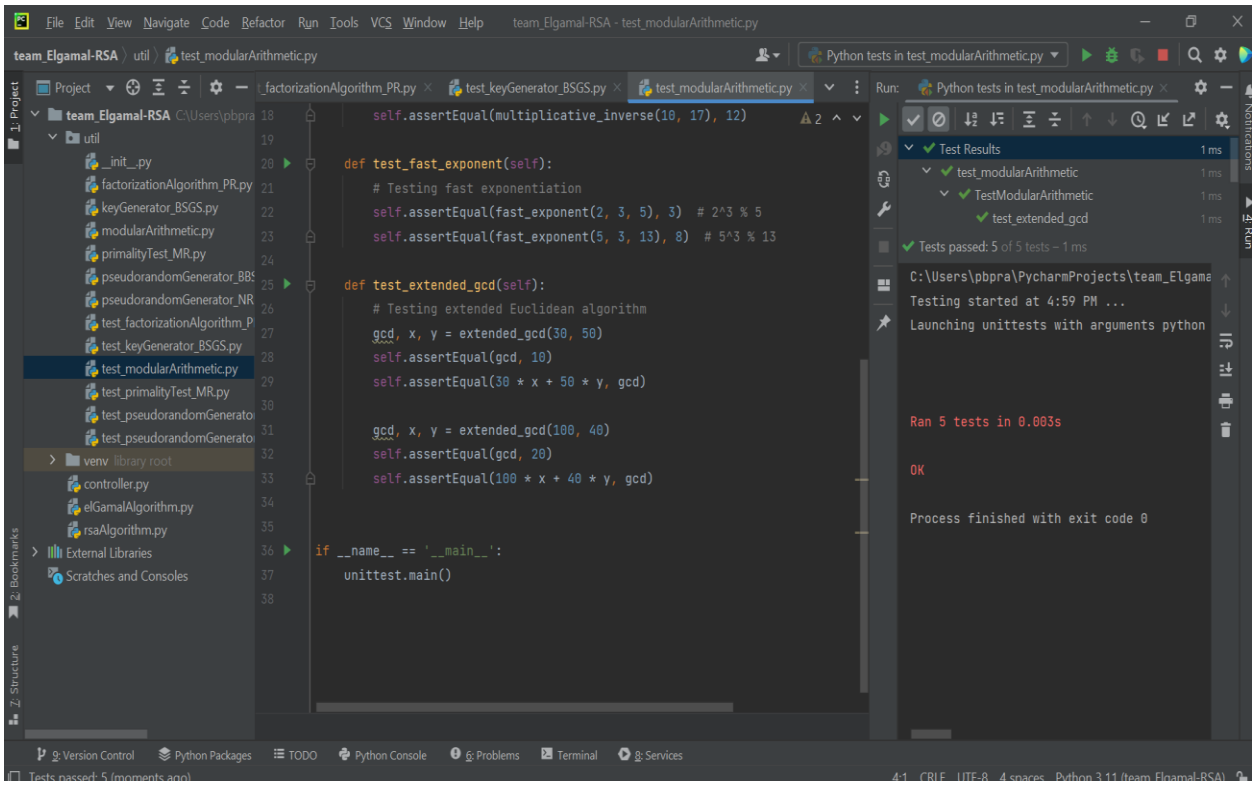
The test cases in the **util** folder are designed to cover a wide range of scenarios to achieve close to 100% test coverage for the individual utilities used in the application.

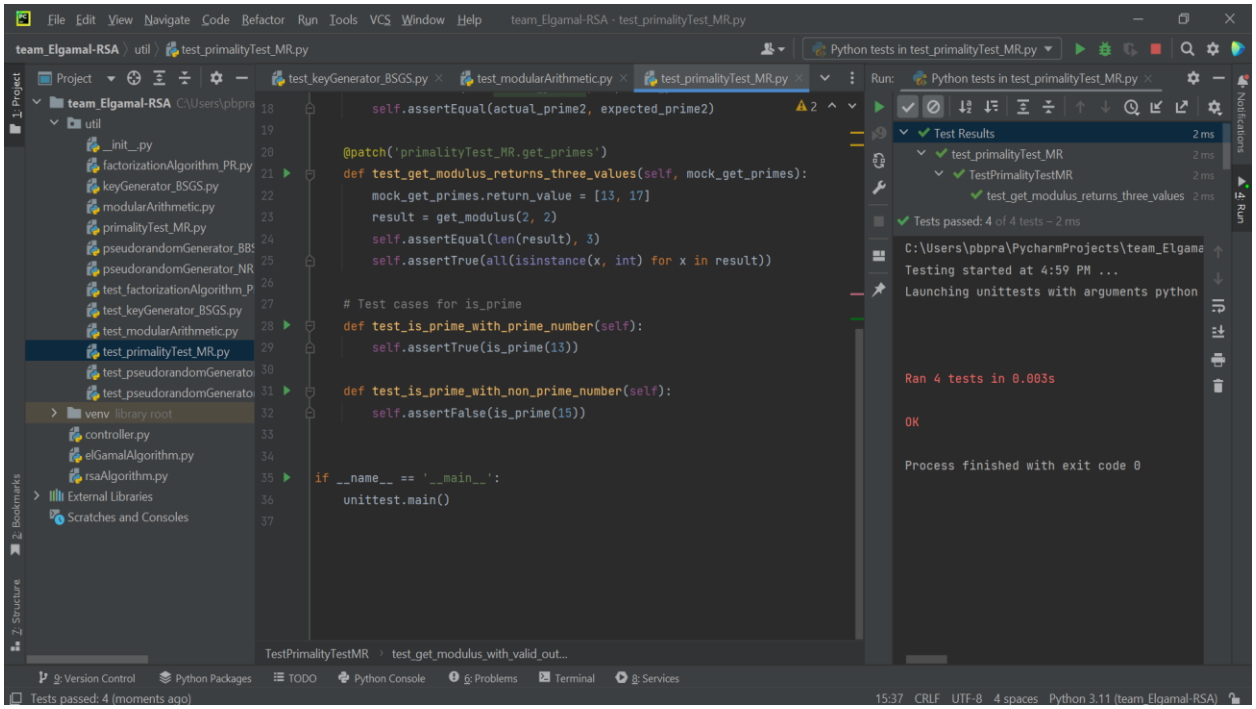
Testing and Validation

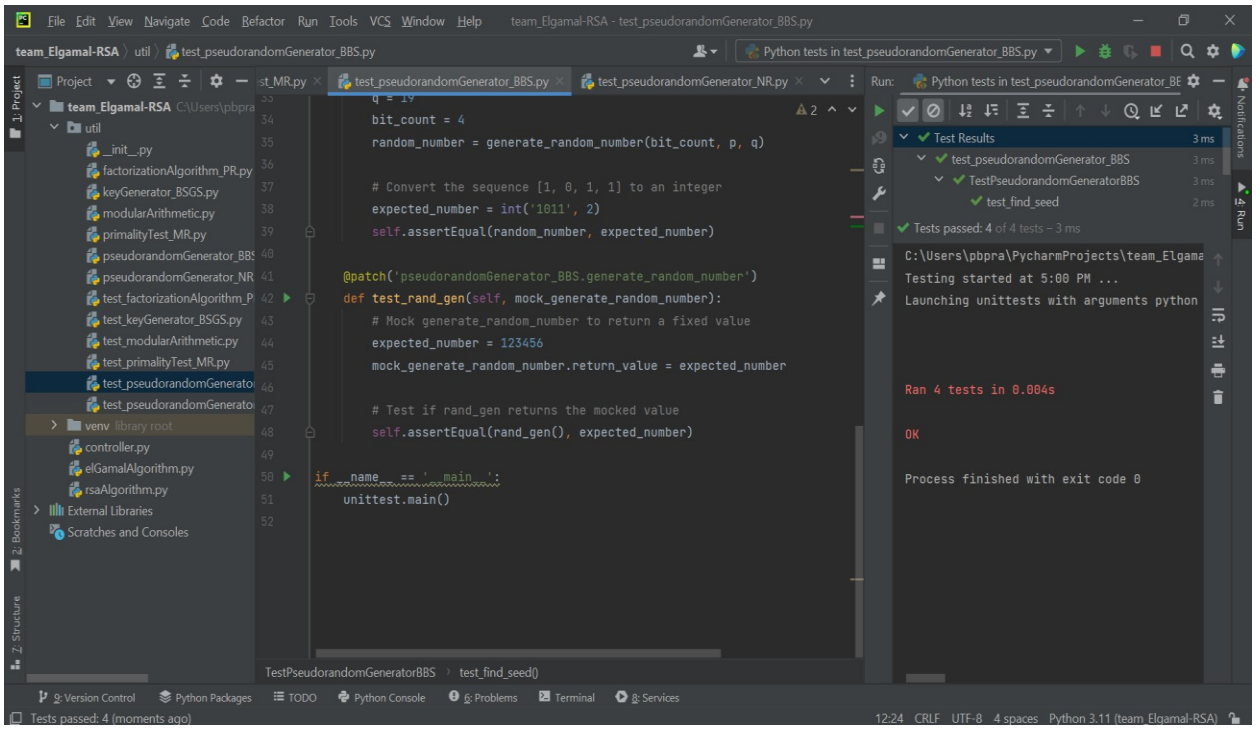
- **Unit Tests**:

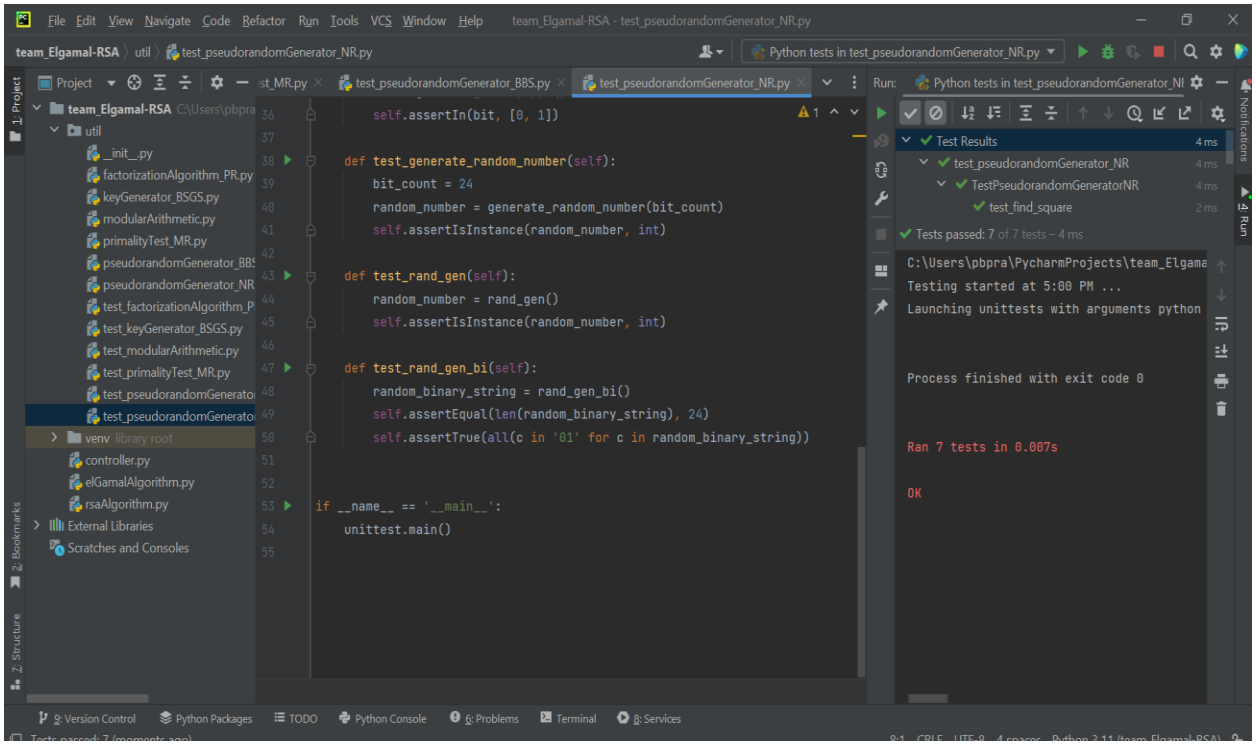





File Edit View Navigate Code Refactor Run Tools VCS Window Help team_Elgamal-RSA - test_modularArithmetic.py
team_Elgamal-RSA util test_modularArithmetic.py
Python tests in test_modularArithmetic.py
Project
team_Elgamal-RSA C:\Users\pbpra
util
__init__.py
factorizationAlgorithm_PR.py
keyGenerator_BSGS.py
modularArithmetic.py
primalityTest_MR.py
pseudorandomGenerator_BBS
pseudorandomGenerator_NR
test_factorizationAlgorithm_P
test_keyGenerator_BSGS.py
test_modularArithmetic.py
test_primalityTest_MR.py
test_pseudorandomGenerato
test_pseudorandomGenerato
venv library root
controller.py
elGamalAlgorithm.py
rsaAlgorithm.py
External Libraries
Scratches and Consoles
factorizationAlgorithm_PR.py
test_keyGenerator_BSGS.py
test_modularArithmetic.py
self.assertEqual(multiplicative_inverse(10, 17), 12)
def test_fast_exponent(self):
# Testing fast exponentiation
self.assertEqual(fast_exponent(2, 3, 5), 3) # 2^3 % 5
self.assertEqual(fast_exponent(5, 3, 13), 8) # 5^3 % 13
def test_extended_gcd(self):
# Testing extended Euclidean algorithm
gcd, x, y = extended_gcd(30, 50)
self.assertEqual(gcd, 10)
self.assertEqual(30 * x + 50 * y, gcd)
gcd, x, y = extended_gcd(100, 40)
self.assertEqual(gcd, 20)
self.assertEqual(100 * x + 40 * y, gcd)
if __name__ == '__main__':
unittest.main()
Run: Python tests in test_modularArithmetic.py
Test Results 1 ms
test_modularArithmetic 1 ms
TestModularArithmetic 1 ms
test_extended_gcd 1 ms
Tests passed: 5 of 5 tests – 1 ms
C:\Users\pbpra\PycharmProjects\team_Elgama
Testing started at 4:59 PM ...
Launching unittests with arguments python
Ran 5 tests in 0.003s
OK
Process finished with exit code 0
9: Version Control
Python Packages
TODO
Python Console
6: Problems
Terminal
8: Services


File Edit View Navigate Code Refactor Run Tools VCS Window Help team_Elgamal-RSA - test_primalityTest_MR.py
team_Elgamal-RSA util test_primalityTest_MR.py
Python tests in test_primalityTest_MR.py
Project
team_Elgamal-RSA C:\Users\pbpra
util
__init__.py
factorizationAlgorithm_PR.py
keyGenerator_BSGS.py
modularArithmetic.py
primalityTest_MR.py
pseudorandomGenerator_BBS
pseudorandomGenerator_NR
test_factorizationAlgorithm_P
test_keyGenerator_BSGS.py
test_modularArithmetic.py
test_primalityTest_MR.py
test_pseudorandomGenerato
test_pseudorandomGenerato
venv library root
controller.py
elGamalAlgorithm.py
rsaAlgorithm.py
External Libraries
Scratches and Consoles
test_keyGenerator_BSGS.py
test_modularArithmetic.py
test_primalityTest_MR.py
self.assertEqual(actual_prime2, expected_prime2)
@patch('primalityTest_MR.get_primes')
def test_get_modulus_returns_three_values(self, mock_get_primes):
mock_get_primes.return_value = [13, 17]
result = get_modulus(2, 2)
self.assertEqual(len(result), 3)
self.assertTrue(all(isinstance(x, int) for x in result))
# Test cases for is_prime
def test_is_prime_with_prime_number(self):
self.assertTrue(is_prime(13))
def test_is_prime_with_non_prime_number(self):
self.assertFalse(is_prime(15))
if __name__ == '__main__':
unittest.main()
TestPrimalityTestMR test_get_modulus_with_valid_out...
Run: Python tests in test_primalityTest_MR.py
Test Results 2 ms
test_primalityTest_MR 2 ms
TestPrimalityTestMR 2 ms
test_get_modulus_returns_three_values 2 ms
Tests passed: 4 of 4 tests – 2 ms
C:\Users\pbpra\PycharmProjects\team_Elgama
Testing started at 4:59 PM ...
Launching unittests with arguments python
Ran 4 tests in 0.003s
OK
Process finished with exit code 0
9: Version Control
Python Packages
TODO
Python Console
6: Problems
Terminal
8: Services
Tests passed: 4 (moments ago)
15:37 CRLF UTF-8 4 spaces Python 3.11 (team_Elgamal-RSA)

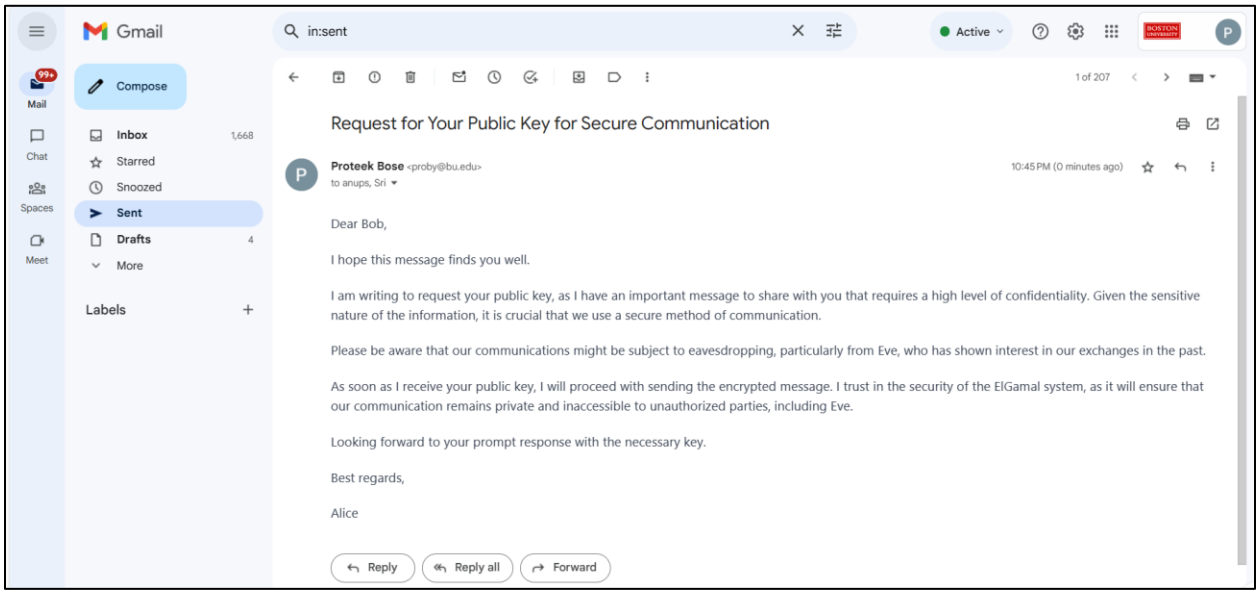

File Edit View Navigate Code Refactor Run Tools VCS Window Help
team_Elgamal-RSA - test_pseudorandomGenerator_BBS.py
team_Elgamal-RSA util test_pseudorandomGenerator_BBS.py
Python tests in test_pseudorandomGenerator_BBS.py
Project
team_Elgamal-RSA C:\Users\pbpra
util
__init__.py
factorizationAlgorithm_PR.py
keyGenerator_BSGS.py
modularArithmetic.py
primalityTest_MR.py
pseudorandomGenerator_BBS
pseudorandomGenerator_NR
test_factorizationAlgorithm_P
test_keyGenerator_BSGS.py
test_modularArithmetic.py
test_primalityTest_MR.py
test_pseudorandomGenerato
test_pseudorandomGenerato
venv library root
controller.py
elGamalAlgorithm.py
rsaAlgorithm.py
External Libraries
Scratches and Consoles
st_MR.py
test_pseudorandomGenerator_BBS.py
test_pseudorandomGenerator_NR.py
q = 19
bit_count = 4
random_number = generate_random_number(bit_count, p, q)
# Convert the sequence [1, 0, 1, 1] to an integer
expected_number = int('1011', 2)
self.assertEqual(random_number, expected_number)
@patch('pseudorandomGenerator_BBS.generate_random_number')
def test_rand_gen(self, mock_generate_random_number):
# Mock generate_random_number to return a fixed value
expected_number = 123456
mock_generate_random_number.return_value = expected_number
# Test if rand_gen returns the mocked value
self.assertEqual(rand_gen(), expected_number)
if __name__ == '__main__':
unittest.main()
TestPseudorandomGeneratorBBS test_find_seed()
Run:
Python tests in test_pseudorandomGenerator_BE
Test Results 3 ms
test_pseudorandomGenerator_BBS 3 ms
TestPseudorandomGeneratorBBS 3 ms
test_find_seed 2 ms
Tests passed: 4 of 4 tests – 3 ms
C:\Users\pbpra\PycharmProjects\team_Elgama
Testing started at 5:00 PM ...
Launching unittests with arguments python
Ran 4 tests in 0.004s
OK
Process finished with exit code 0
9: Version Control
Python Packages
TODO
Python Console
6: Problems
Terminal
8: Services
Tests passed: 4 (moments ago)
12:24 CRLF UTF-8 4 spaces Python 3.11 (team_Elgamal-RSA)

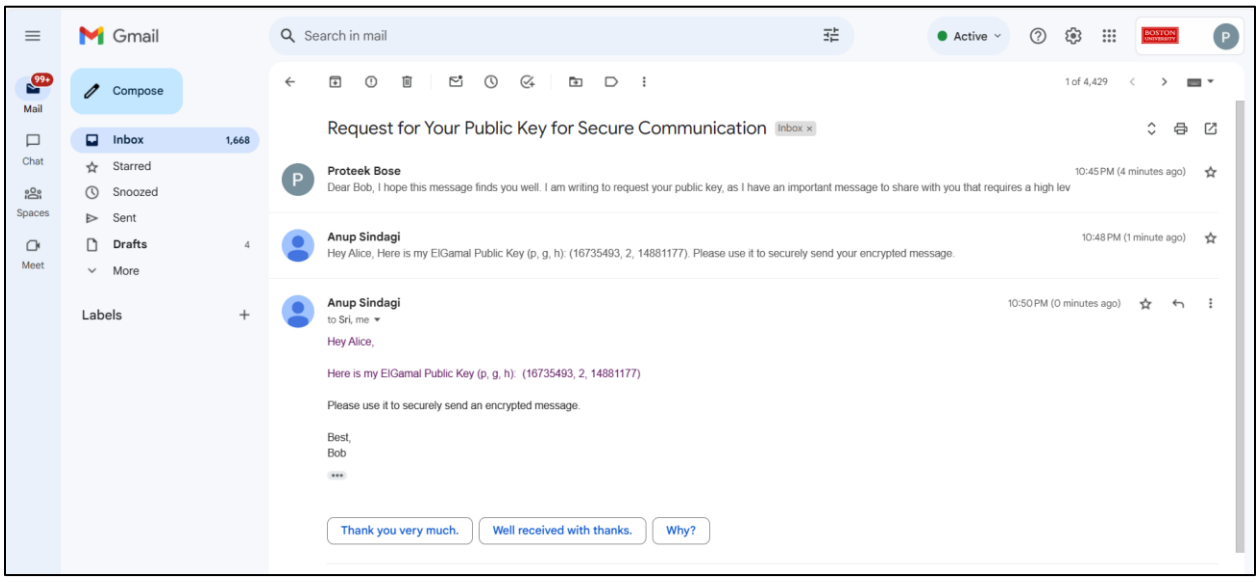

File Edit View Navigate Code Refactor Run Tools VCS Window Help
team_Elgamal-RSA - test_pseudorandomGenerator_NR.py
team_Elgamal-RSA util test_pseudorandomGenerator_NR.py
Python tests in test_pseudorandomGenerator_NR.py
Project
team_Elgamal-RSA C:\Users\pbpra
util
__init__.py
factorizationAlgorithm_PR.py
keyGenerator_BSGS.py
modularArithmetic.py
primalityTest_MR.py
pseudorandomGenerator_BBS
pseudorandomGenerator_NR
test_factorizationAlgorithm_P
test_keyGenerator_BSGS.py
test_modularArithmetic.py
test_primalityTest_MR.py
test_pseudorandomGenerato
test_pseudorandomGenerato
venv library root
controller.py
elGamalAlgorithm.py
rsaAlgorithm.py
External Libraries
Scratches and Consoles
st_MR.py
test_pseudorandomGenerator_BBS.py
test_pseudorandomGenerator_NR.py
self.assertIn(bit, [0, 1])
def test_generate_random_number(self):
bit_count = 24
random_number = generate_random_number(bit_count)
self.assertIsInstance(random_number, int)
def test_rand_gen(self):
random_number = rand_gen()
self.assertIsInstance(random_number, int)
def test_rand_gen_bi(self):
random_binary_string = rand_gen_bi()
self.assertEqual(len(random_binary_string), 24)
self.assertTrue(all(c in '01' for c in random_binary_string))
if __name__ == '__main__':
unittest.main()
Run:
Python tests in test_pseudorandomGenerator_NI
Test Results 4 ms
test_pseudorandomGenerator_NR 4 ms
TestPseudorandomGeneratorNR 4 ms
test_find_square 2 ms
Tests passed: 7 of 7 tests – 4 ms
C:\Users\pbpra\PycharmProjects\team_Elgama
Testing started at 5:00 PM ...
Launching unittests with arguments python
Process finished with exit code 0
Ran 7 tests in 0.007s
OK
9: Version Control
Python Packages
TODO
Python Console
6: Problems
Terminal
8: Services

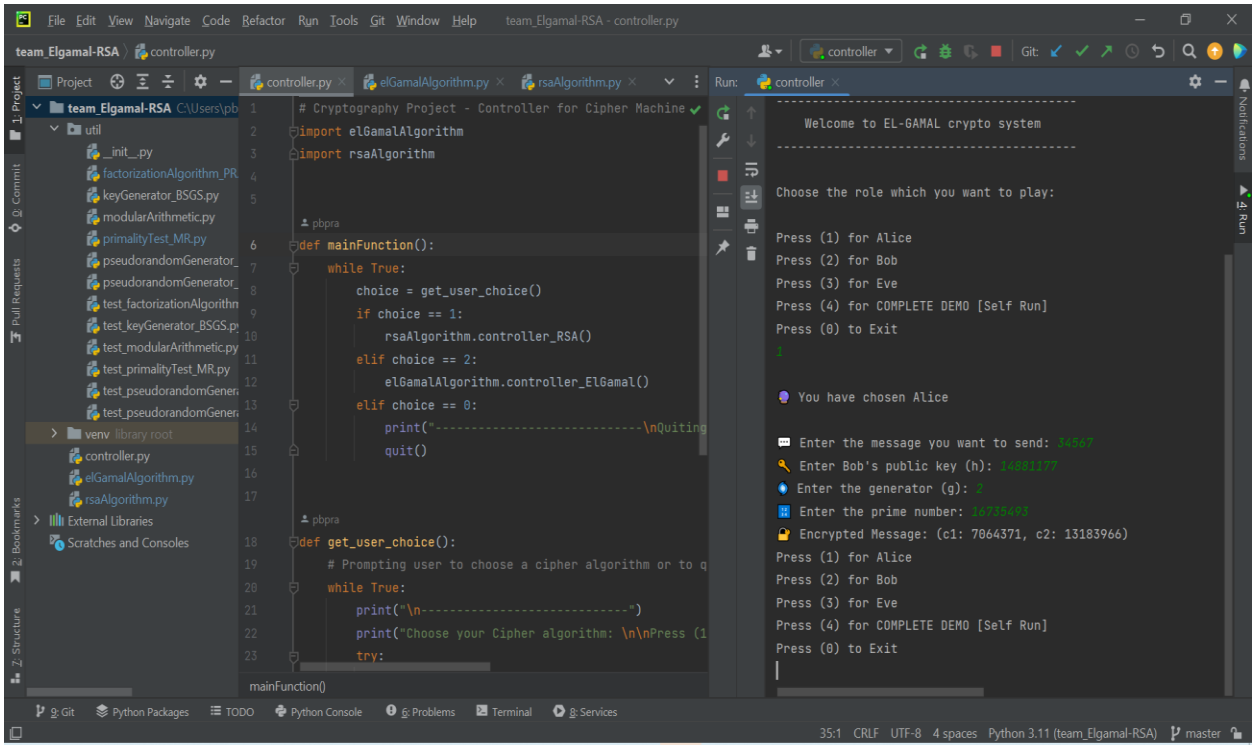
Documentation of Exchanges

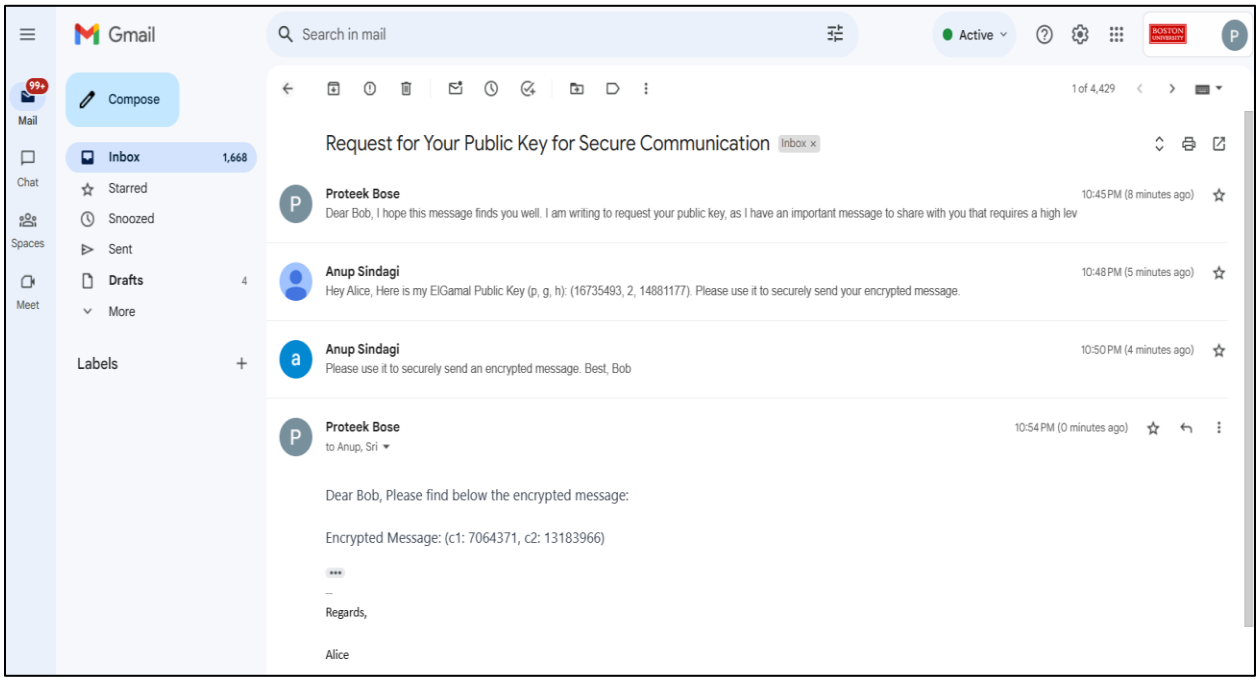
Alice, Bob, and Eve in ElGamal:

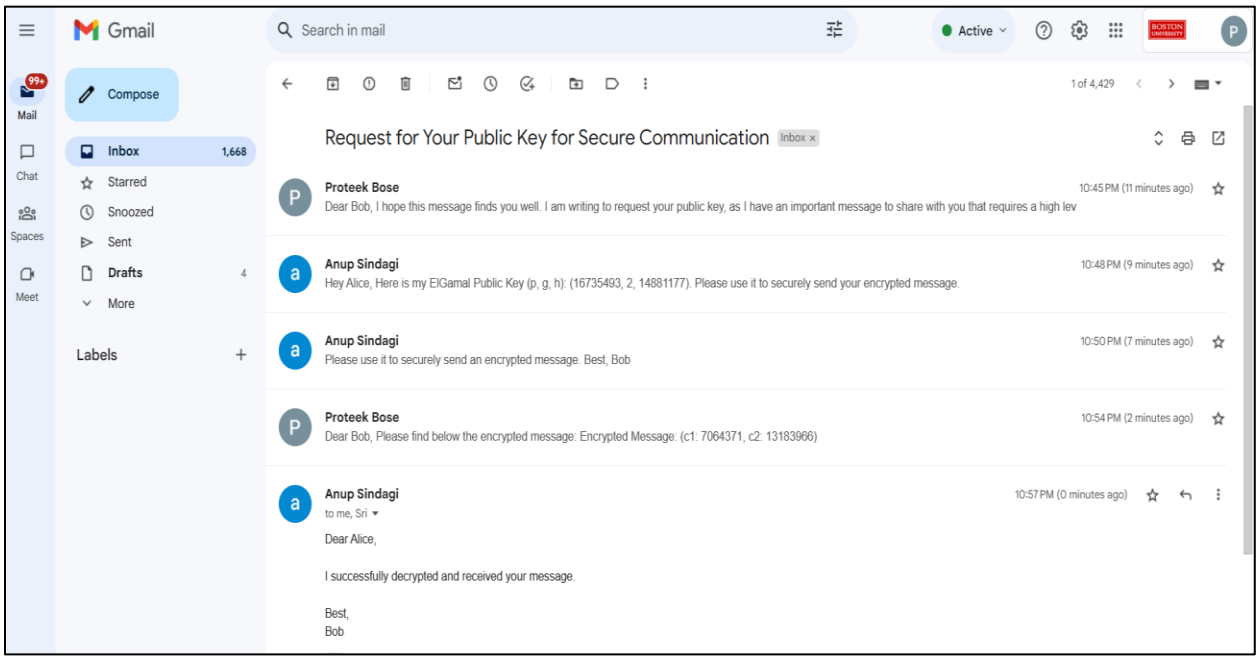
Playing Alice: The message that I sent to bob was “34567”

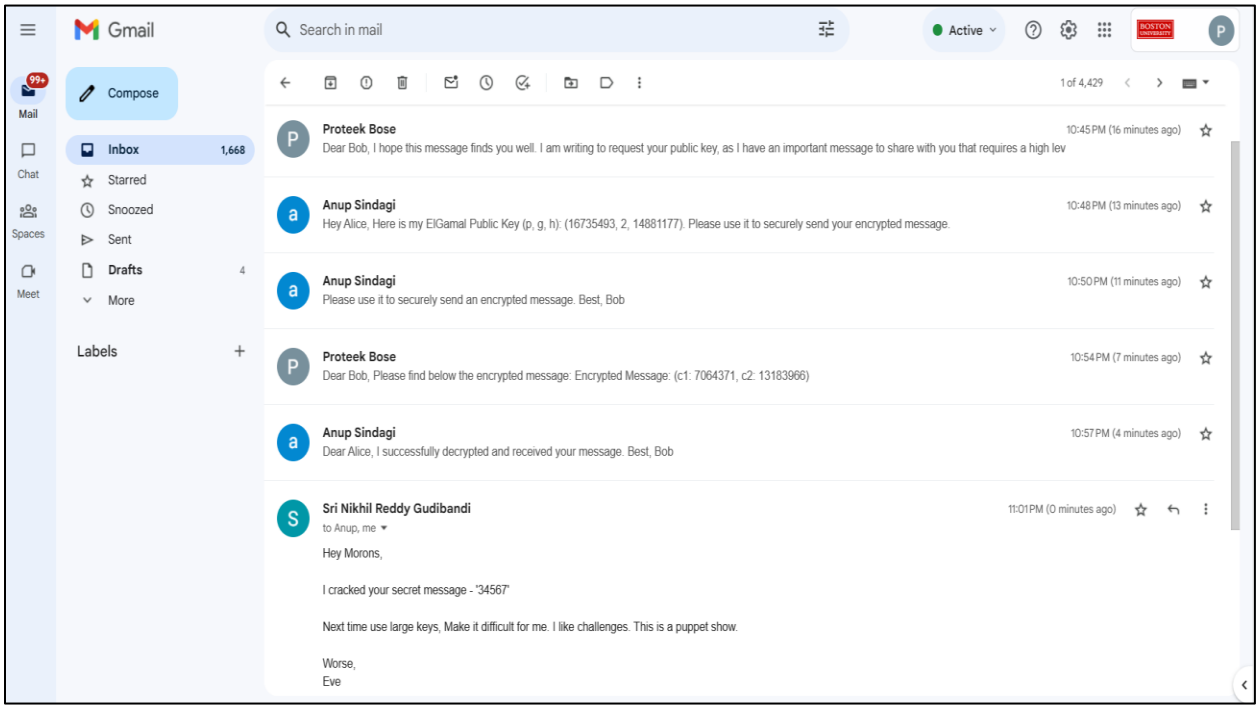




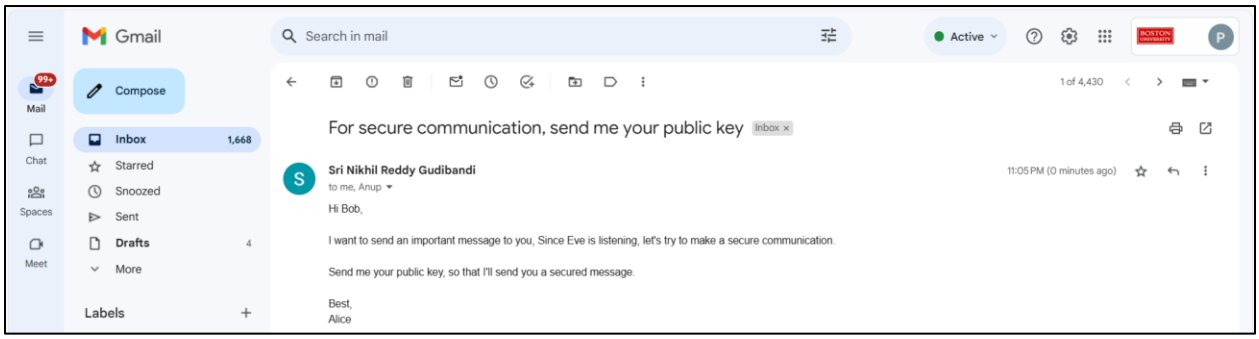

File Edit View Navigate Code Refactor Run Tools Git Window Help
team_Elgamal-RSA - controller.py
team_Elgamal-RSA
controller.py
Project
controller.py
elGamalAlgorithm.py
rsaAlgorithm.py
Run:
controller
# Cryptography Project - Controller for Cipher Machine
import elGamalAlgorithm
import rsaAlgorithm
def mainFunction():
while True:
choice = get_user_choice()
if choice == 1:
rsaAlgorithm.controller_RSA()
elif choice == 2:
elGamalAlgorithm.controller_ElGamal()
elif choice == 0:
print("----------\nQuiting
quit()
def get_user_choice():
# Prompting user to choose a cipher algorithm or to q
while True:
print("\n----------")
print("Choose your Cipher algorithm: \n\nPress (1
try:
mainFunction()
Welcome to EL-GAMAL crypto system
Choose the role which you want to play:
Press (1) for Alice
Press (2) for Bob
Press (3) for Eve
Press (4) for COMPLETE DEMO [Self Run]
Press (0) to Exit
1
You have chosen Alice
Enter the message you want to send: 34567
Enter Bob's public key (h): 14881177
Enter the generator (g): 2
Enter the prime number: 16735493
Encrypted Message: (c1: 7064371, c2: 13183966)
Press (1) for Alice
Press (2) for Bob
Press (3) for Eve
Press (4) for COMPLETE DEMO [Self Run]
Press (0) to Exit
9: Git
Python Packages
TODO
Python Console
6: Problems
Terminal
8: Services
35:1 CRLF UTF-8 4 spaces Python 3.11 (team_Elgamal-RSA)
master

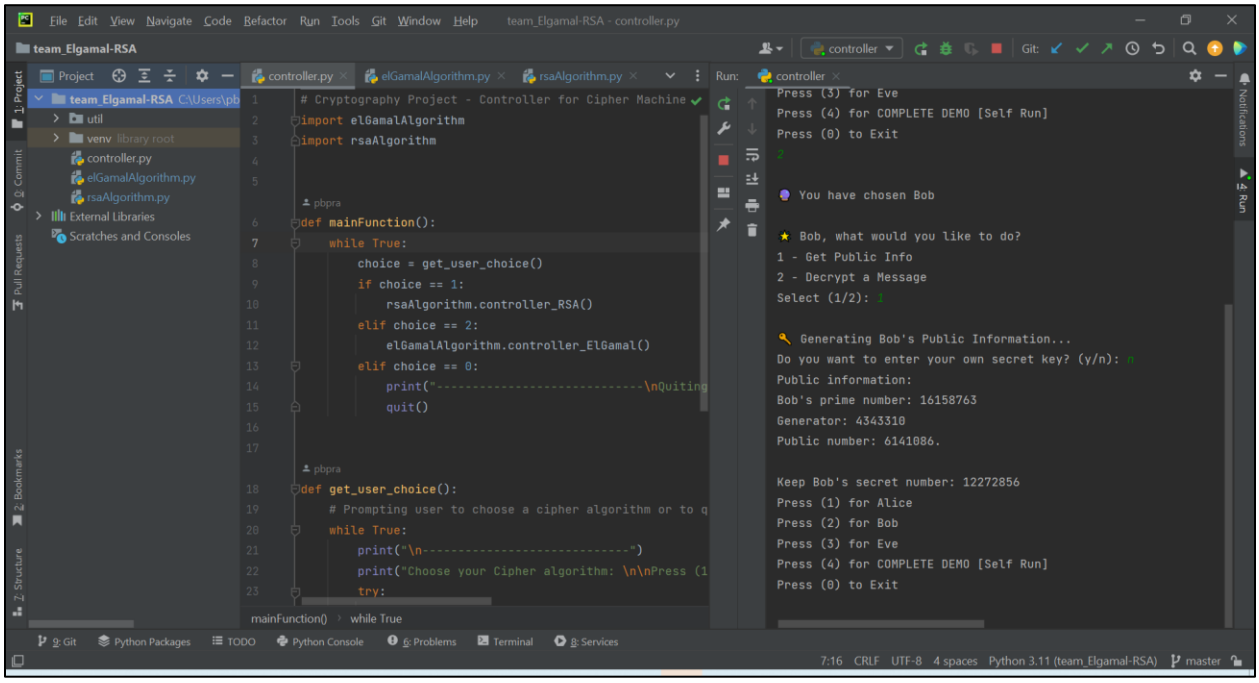

Gmail
Search in mail
Active
Compose
Inbox 1,668
Starred
Snoozed
Sent
Drafts 4
More
Labels
Mail
Chat
Spaces
Meet
1 of 4,429
Request for Your Public Key for Secure Communication
Inbox
Proteek Bose
Dear Bob, I hope this message finds you well. I am writing to request your public key, as I have an important message to share with you that requires a high lev
10:45 PM (8 minutes ago)
Anup Sindagi
Hey Alice, Here is my ElGamal Public Key (p, g, h): (16735493, 2, 14881177). Please use it to securely send your encrypted message.
10:48 PM (5 minutes ago)
Anup Sindagi
Please use it to securely send an encrypted message. Best, Bob
10:50 PM (4 minutes ago)
Proteek Bose
to Anup, Sri
10:54 PM (0 minutes ago)
Dear Bob, Please find below the encrypted message:
Encrypted Message: (c1: 7064371, c2: 13183966)
Regards,
Alice

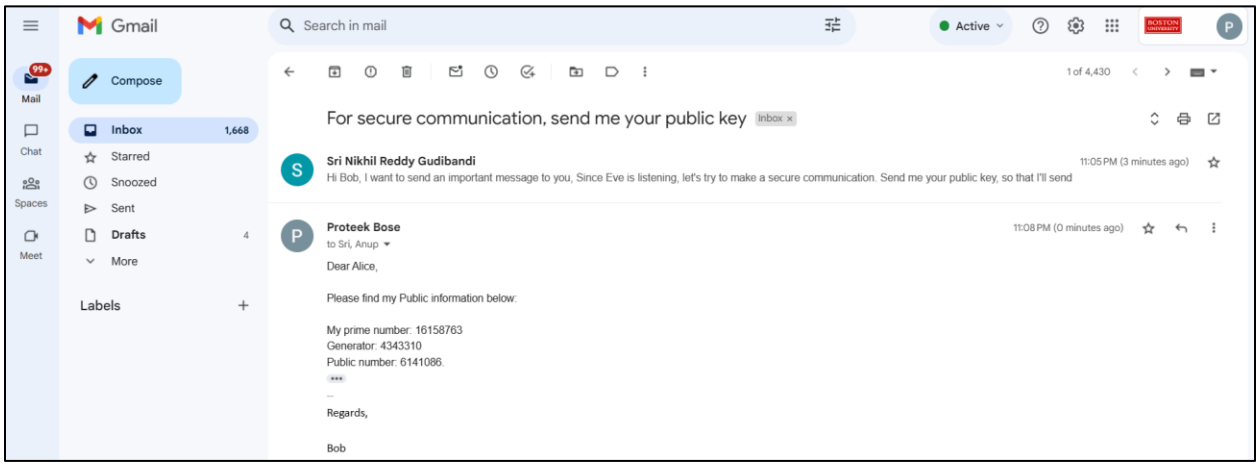

Gmail
Search in mail
Active
Compose
Inbox 1,668
Starred
Snoozed
Sent
Drafts 4
More
Labels
1 of 4,429
Request for Your Public Key for Secure Communication
Inbox
Proteek Bose
Dear Bob, I hope this message finds you well. I am writing to request your public key, as I have an important message to share with you that requires a high lev
10:45 PM (11 minutes ago)
Anup Sindagi
Hey Alice, Here is my ElGamal Public Key (p, g, h): (16735493, 2, 14881177). Please use it to securely send your encrypted message.
10:48 PM (9 minutes ago)
Anup Sindagi
Please use it to securely send an encrypted message. Best, Bob
10:50 PM (7 minutes ago)
Proteek Bose
Dear Bob, Please find below the encrypted message: Encrypted Message: (c1: 7064371, c2: 13183966)
10:54 PM (2 minutes ago)
Anup Sindagi
to me, Sri
10:57 PM (0 minutes ago)
Dear Alice,
I successfully decrypted and received your message.
Best,
Bob

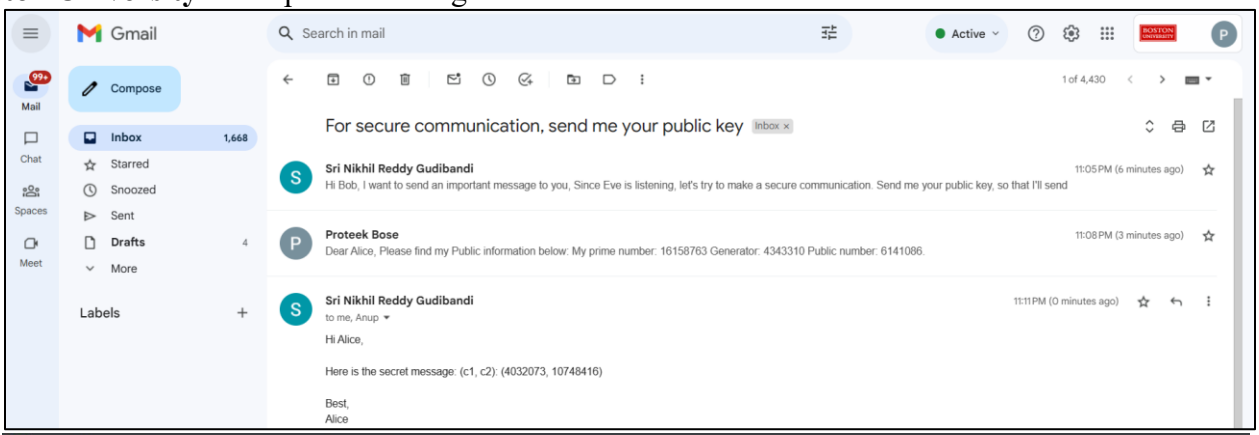

Gmail
Search in mail
Active
Compose
Inbox 1,668
Starred
Snoozed
Sent
Drafts 4
More
Labels
1 of 4,429
Proteek Bose
Dear Bob, I hope this message finds you well. I am writing to request your public key, as I have an important message to share with you that requires a high lev
10:45 PM (16 minutes ago)
Anup Sindagi
Hey Alice, Here is my ElGamal Public Key (p, g, h): (16735493, 2, 14881177). Please use it to securely send your encrypted message.
10:48 PM (13 minutes ago)
Anup Sindagi
Please use it to securely send an encrypted message. Best, Bob
10:50 PM (11 minutes ago)
Proteek Bose
Dear Bob, Please find below the encrypted message: Encrypted Message: (c1: 7064371, c2: 13183966)
10:54 PM (7 minutes ago)
Anup Sindagi
Dear Alice, I successfully decrypted and received your message. Best, Bob
10:57 PM (4 minutes ago)
Sri Nikhil Reddy Gudibandi
to Anup, me
11:01 PM (0 minutes ago)
Hey Morons,
I cracked your secret message - "34567"
Next time use large keys, Make it difficult for me. I like challenges. This is a puppet show.
Worse,
Eve

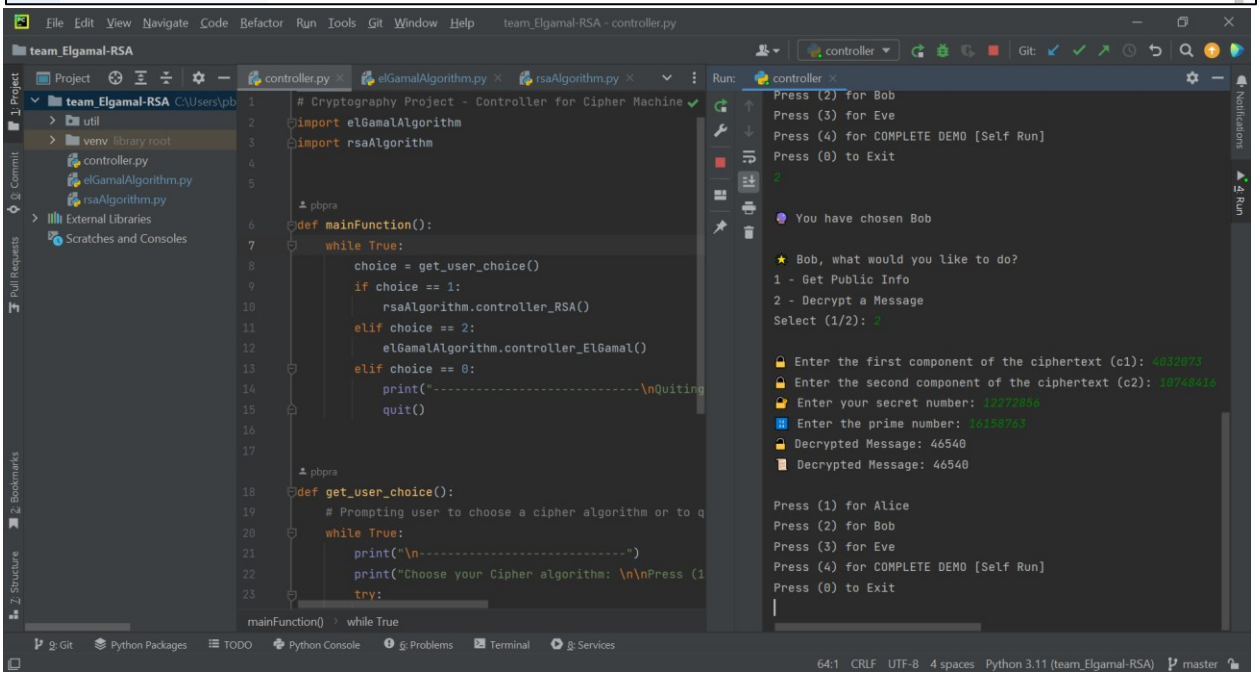
Playing Bob: The message that I received from Alice was "46540"

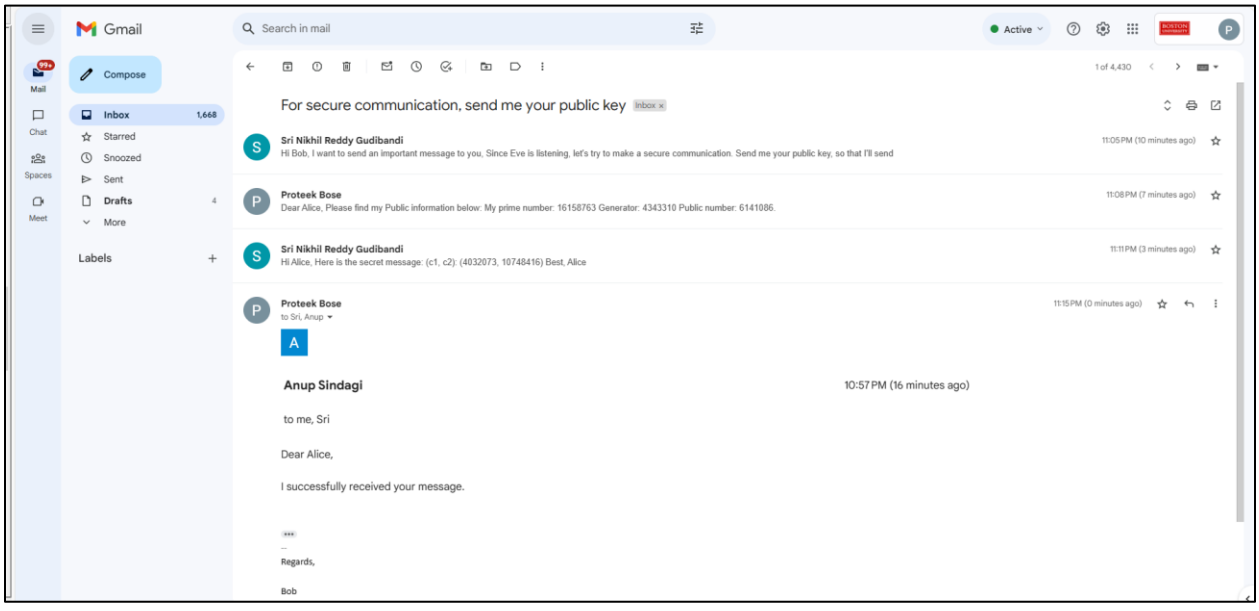


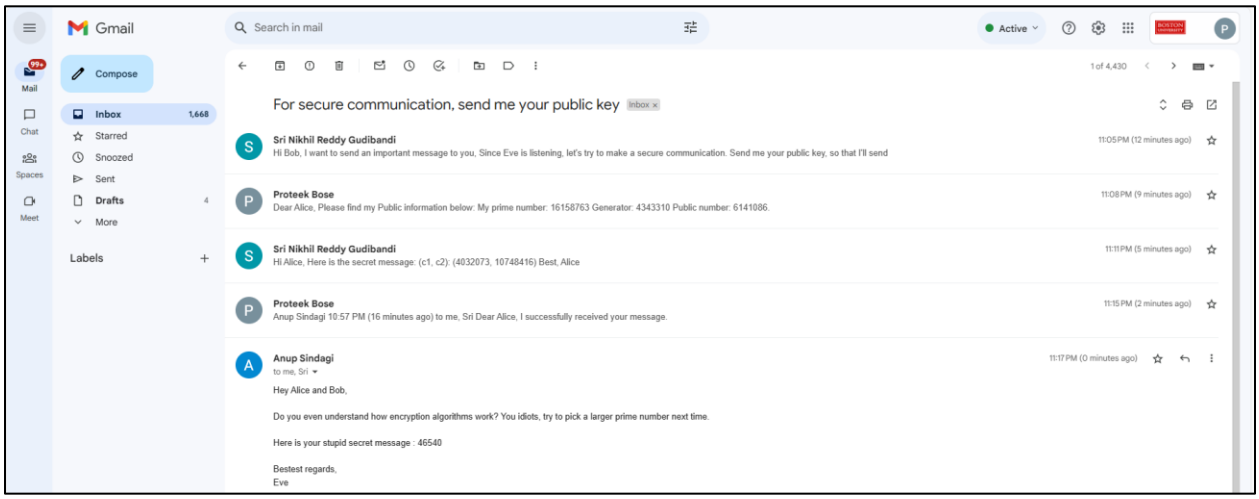




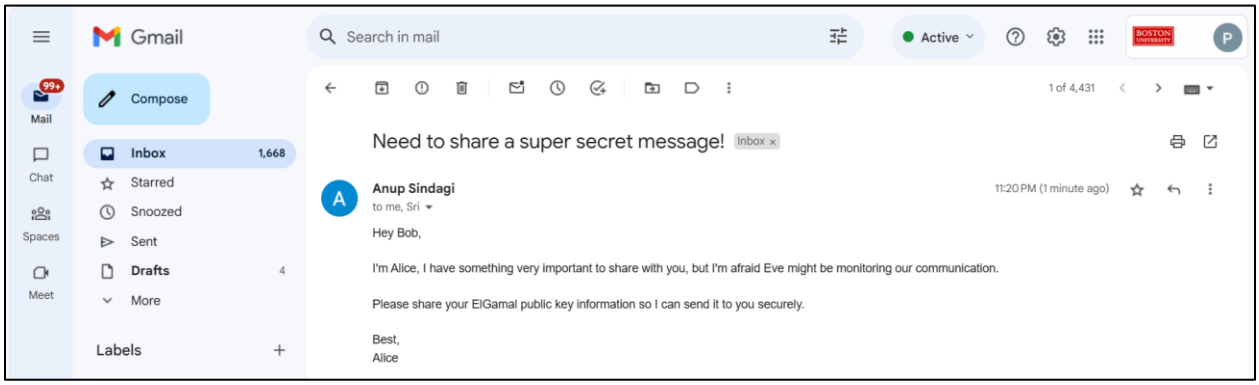

Gmail
Search in mail
Compose
Inbox 1,668
Starred
Snoozed
Sent
Drafts
More
Labels
For secure communication, send me your public key
Sri Nikhil Reddy Gudibandi
Hi Bob, I want to send an important message to you, Since Eve is listening, let's try to make a secure communication. Send me your public key, so that I'll send
11:05 PM (6 minutes ago)
Proteek Bose
Dear Alice, Please find my Public information below: My prime number: 16158763 Generator: 4343310 Public number: 6141086.
11:08 PM (3 minutes ago)
Sri Nikhil Reddy Gudibandi
to me, Anup
Hi Alice,
Here is the secret message: (c1, c2): (4032073, 10748416)
Best,
Alice
11:11 PM (0 minutes ago)

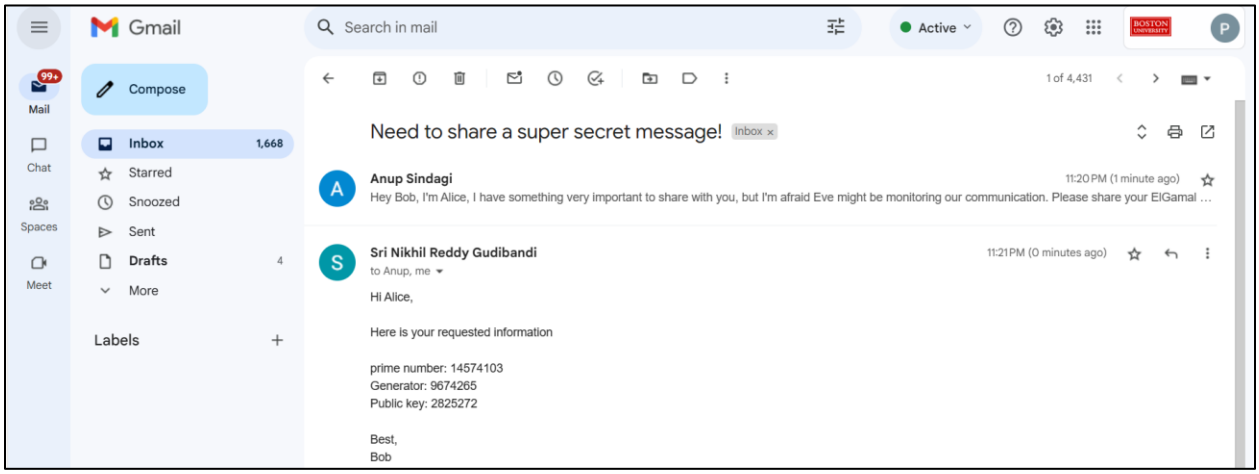

team_Elgamal-RSA
controller.py
elGamalAlgorithm.py
rsaAlgorithm.py
# Cryptography Project - Controller for Cipher Machine
import elGamalAlgorithm
import rsaAlgorithm
def mainFunction():
while True:
choice = get_user_choice()
if choice == 1:
rsaAlgorithm.controller_RSA()
elif choice == 2:
elGamalAlgorithm.controller_ElGamal()
elif choice == 0:
print("----------\nQuiting
quit()
def get_user_choice():
# Prompting user to choose a cipher algorithm or to q
while True:
print("\n----------")
print("Choose your Cipher algorithm: \n\nPress (1
try:
Press (2) for Bob
Press (3) for Eve
Press (4) for COMPLETE DEMO [Self Run]
Press (0) to Exit
2
You have chosen Bob
Bob, what would you like to do?
1 - Get Public Info
2 - Decrypt a Message
Select (1/2): 2
Enter the first component of the ciphertext (c1): 4032073
Enter the second component of the ciphertext (c2): 10748416
Enter your secret number: 12272856
Enter the prime number: 16158763
Decrypted Message: 46540
Decrypted Message: 46540
Press (1) for Alice
Press (2) for Bob
Press (3) for Eve
Press (4) for COMPLETE DEMO [Self Run]
Press (0) to Exit

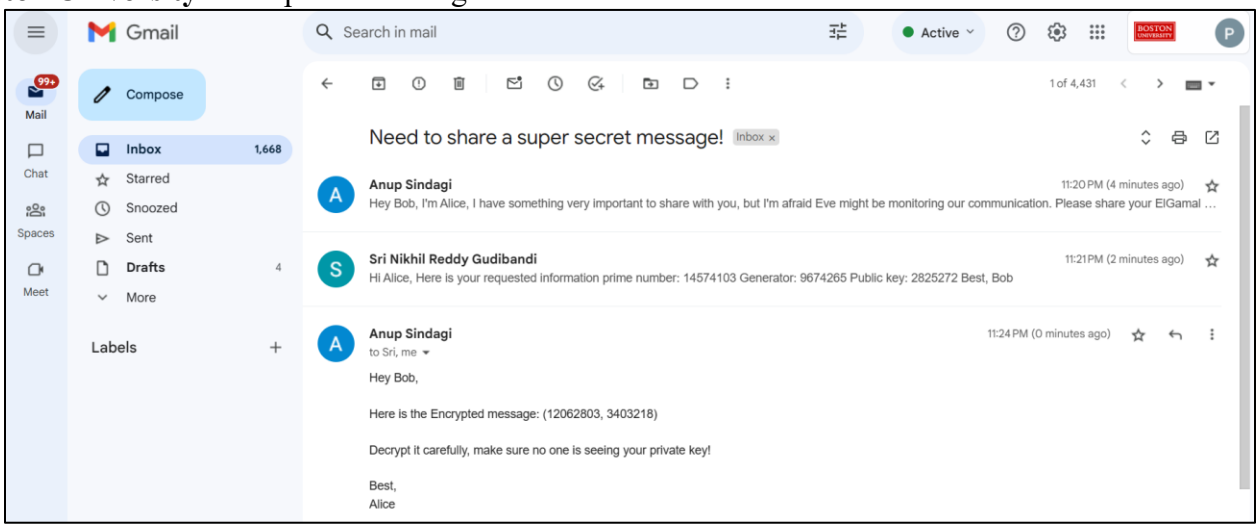

Gmail
Search in mail
Compose
Inbox 1,668
Starred
Snoozed
Sent
Drafts
More
Labels
For secure communication, send me your public key
Sri Nikhil Reddy Gudibandi
Hi Bob, I want to send an important message to you, Since Eve is listening, let's try to make a secure communication. Send me your public key, so that I'll send
11:05 PM (10 minutes ago)
Proteek Bose
Dear Alice, Please find my Public information below: My prime number: 16158763 Generator: 4343310 Public number: 6141086.
11:08 PM (7 minutes ago)
Sri Nikhil Reddy Gudibandi
Hi Alice, Here is the secret message: (c1, c2): (4032073, 10748416) Best, Alice
11:11 PM (3 minutes ago)
Proteek Bose
to Sri, Anup
11:15 PM (0 minutes ago)
A
Anup Sindagi
10:57 PM (16 minutes ago)
to me, Sri
Dear Alice,
I successfully received your message.
Regards,
Bob

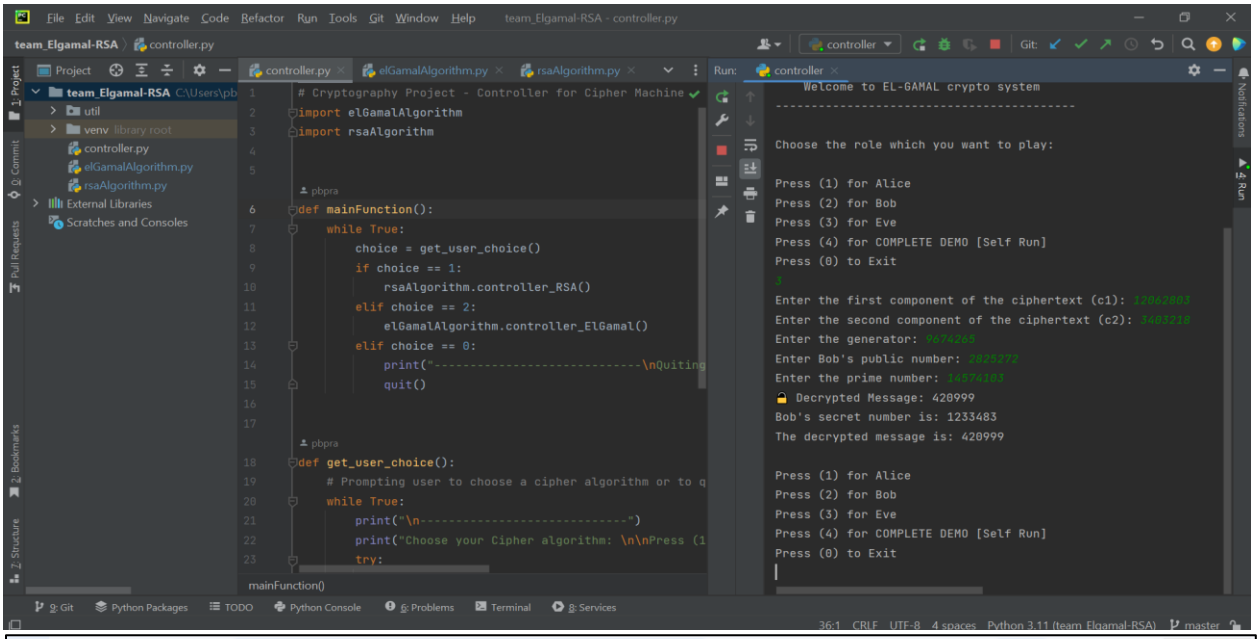


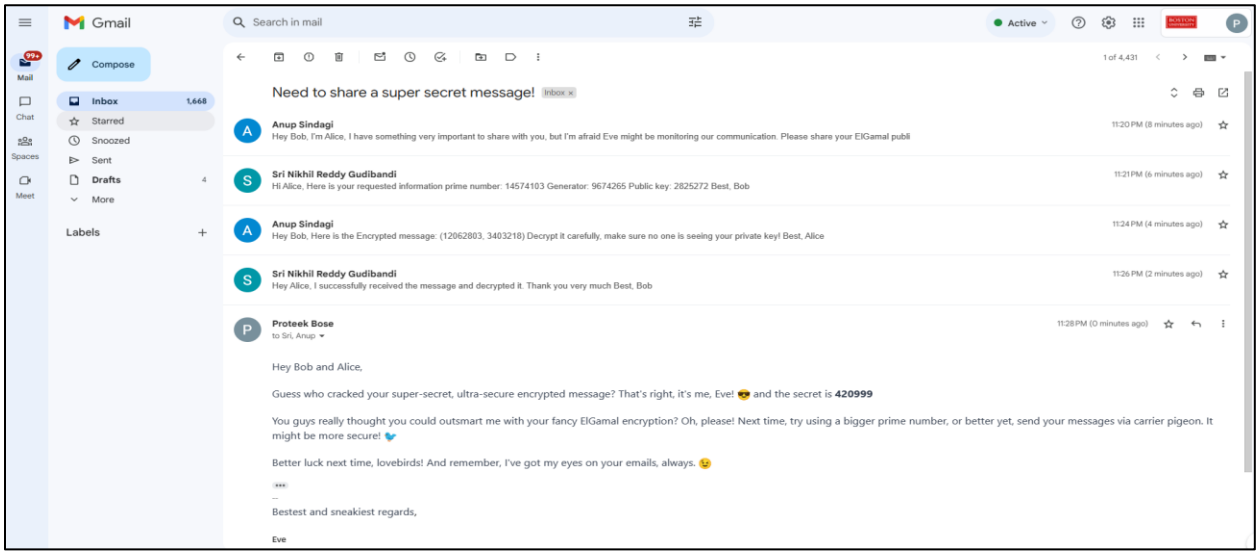
Playing Eve: The message that I cracked is “420999”





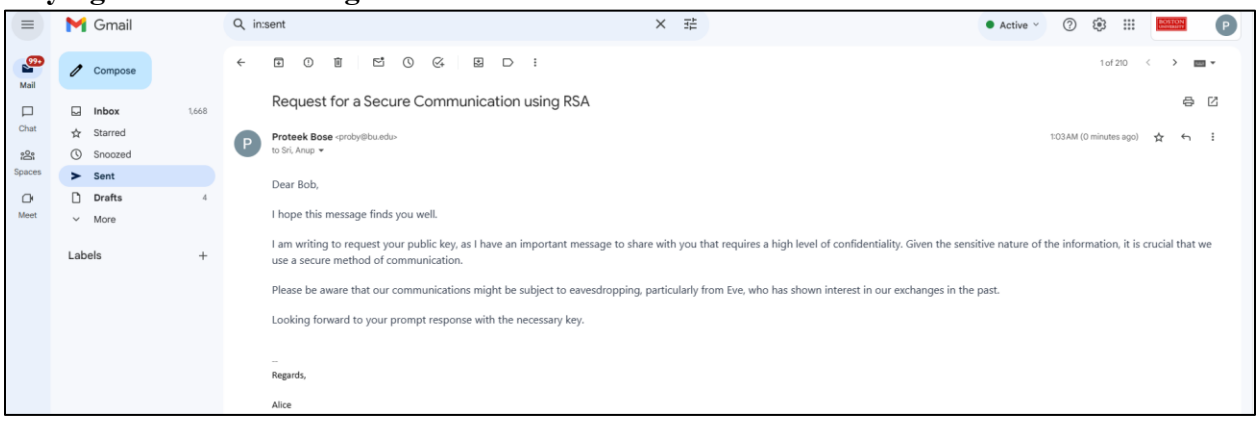

Gmail
Search in mail
Active
Compose
Inbox 1,668
Starred
Snoozed
Sent
Drafts 4
More
Labels
Need to share a super secret message!
Anup Sindagi
11:20 PM (4 minutes ago)
Hey Bob, I'm Alice, I have something very important to share with you, but I'm afraid Eve might be monitoring our communication. Please share your ElGamal ...
Sri Nikhil Reddy Gudibandi
11:21 PM (2 minutes ago)
Hi Alice, Here is your requested information prime number: 14574103 Generator: 9674265 Public key: 2825272 Best, Bob
Anup Sindagi
11:24 PM (0 minutes ago)
to Sri, me
Hey Bob,
Here is the Encrypted message: (12062803, 3403218)
Decrypt it carefully, make sure no one is seeing your private key!
Best,
Alice

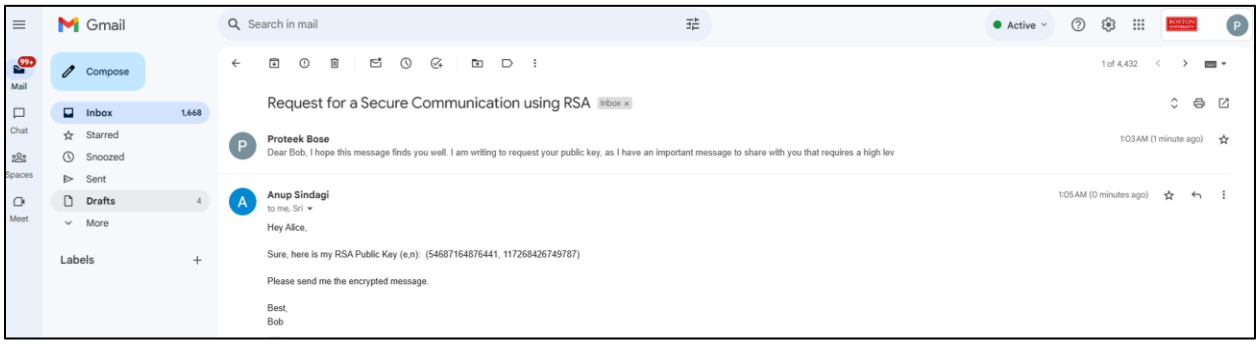

team_Elgamal-RSA - controller.py
controller.py
elGamalAlgorithm.py
rsaAlgorithm.py
# Cryptography Project - Controller for Cipher Machine
import elGamalAlgorithm
import rsaAlgorithm
def mainFunction():
while True:
choice = get_user_choice()
if choice == 1:
rsaAlgorithm.controller_RSA()
elif choice == 2:
elGamalAlgorithm.controller_ElGamal()
elif choice == 0:
print("----------\nQuiting
quit()
def get_user_choice():
# Prompting user to choose a cipher algorithm or to q
while True:
print("\n----------")
print("Choose your Cipher algorithm: \n\nPress (1
try:
Run: controller
Welcome to EL-GAMAL crypto system
Choose the role which you want to play:
Press (1) for Alice
Press (2) for Bob
Press (3) for Eve
Press (4) for COMPLETE DEMO [Self Run]
Press (0) to Exit
3
Enter the first component of the ciphertext (c1): 12062803
Enter the second component of the ciphertext (c2): 3403218
Enter the generator: 9674265
Enter Bob's public number: 2825272
Enter the prime number: 14574103
Decrypted Message: 420999
Bob's secret number is: 1233483
The decrypted message is: 420999
Press (1) for Alice
Press (2) for Bob
Press (3) for Eve
Press (4) for COMPLETE DEMO [Self Run]
Press (0) to Exit

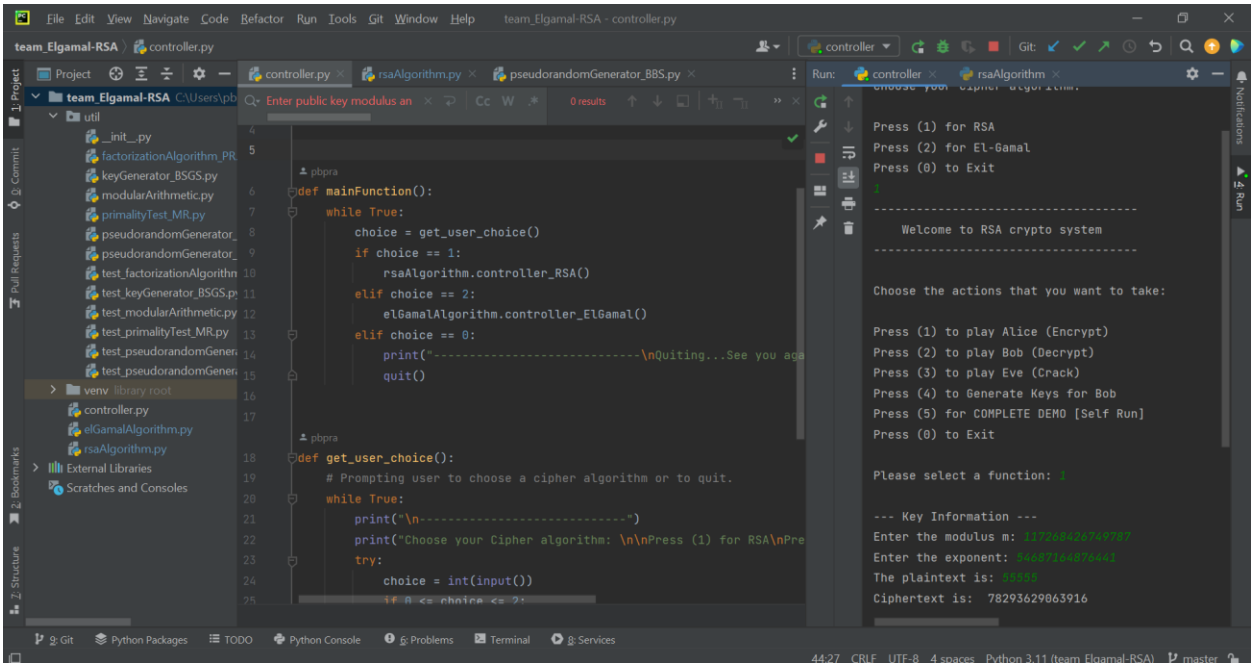

Gmail
Search in mail
Need to share a super secret message!
Anup Sindagi
Hey Bob, I'm Alice, I have something very important to share with you, but I'm afraid Eve might be monitoring our communication. Please share your ElGamal publi
11:20 PM (8 minutes ago)
Sri Nikhil Reddy Gudibandi
Hi Alice, Here is your requested information prime number: 14574103 Generator: 9674265 Public key: 2825272 Best, Bob
11:21 PM (6 minutes ago)
Anup Sindagi
Hey Bob, Here is the Encrypted message: (12062803, 3403218) Decrypt it carefully, make sure no one is seeing your private key! Best, Alice
11:24 PM (4 minutes ago)
Sri Nikhil Reddy Gudibandi
Hey Alice, I successfully received the message and decrypted it. Thank you very much Best, Bob
11:26 PM (2 minutes ago)
Proteek Bose
to Sri, Anup
11:28 PM (0 minutes ago)
Hey Bob and Alice,
Guess who cracked your super-secret, ultra-secure encrypted message? That's right, it's me, Eve! and the secret is 420999
You guys really thought you could outsmart me with your fancy ElGamal encryption? Oh, please! Next time, try using a bigger prime number, or better yet, send your messages via carrier pigeon. It might be more secure!
Better luck next time, lovebirds! And remember, I've got my eyes on your emails, always.
Bestest and sneakiest regards,
Eve

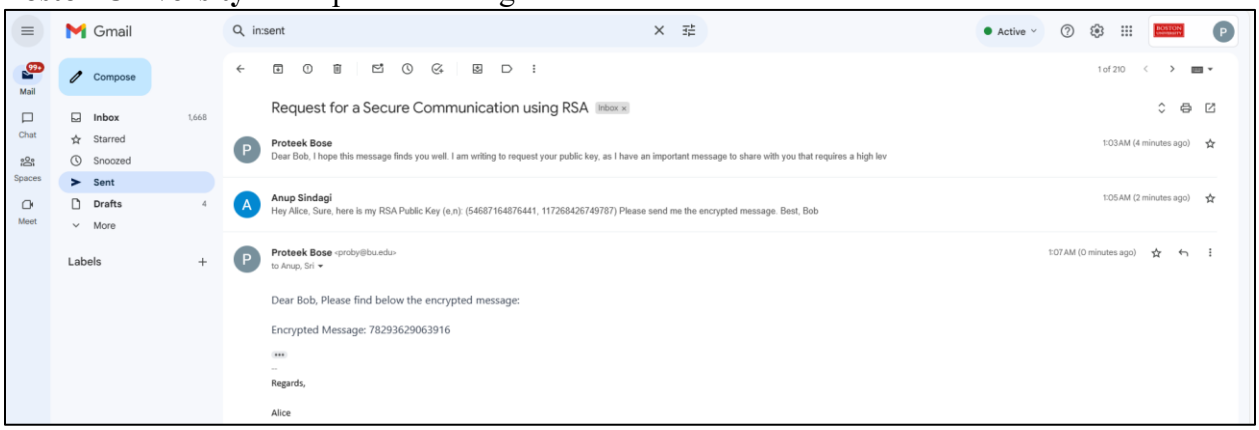
Boston University Metropolitan College
Alice, Bob, and Eve in RSA:

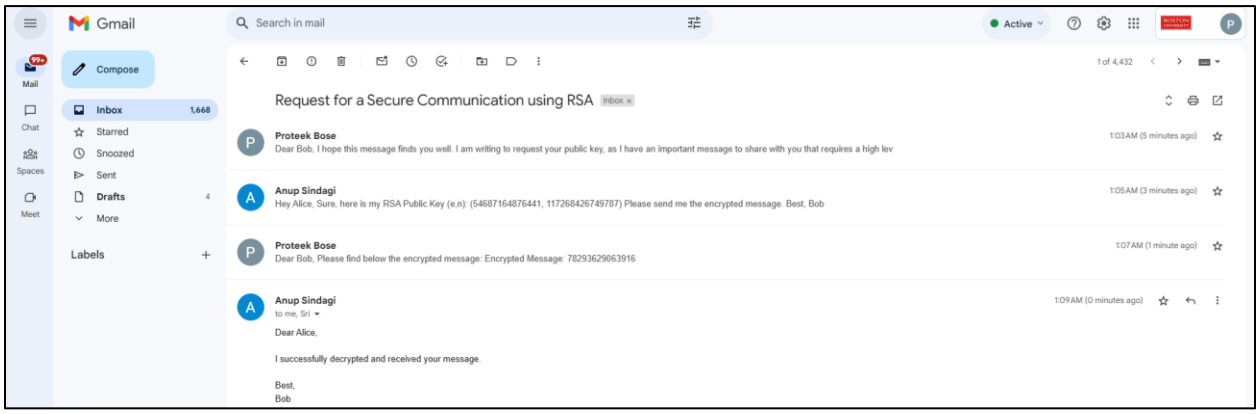
Playing Alice: The message that I sent to bob was “55555”

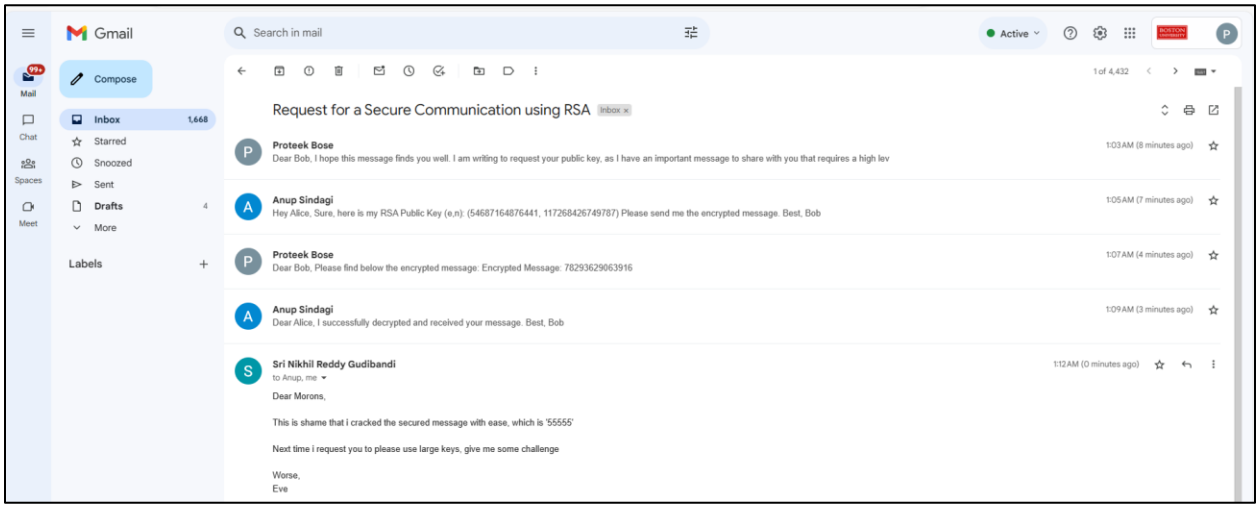




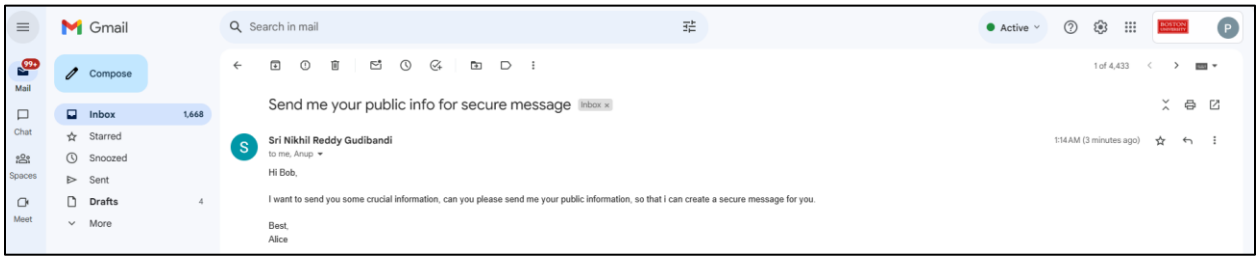


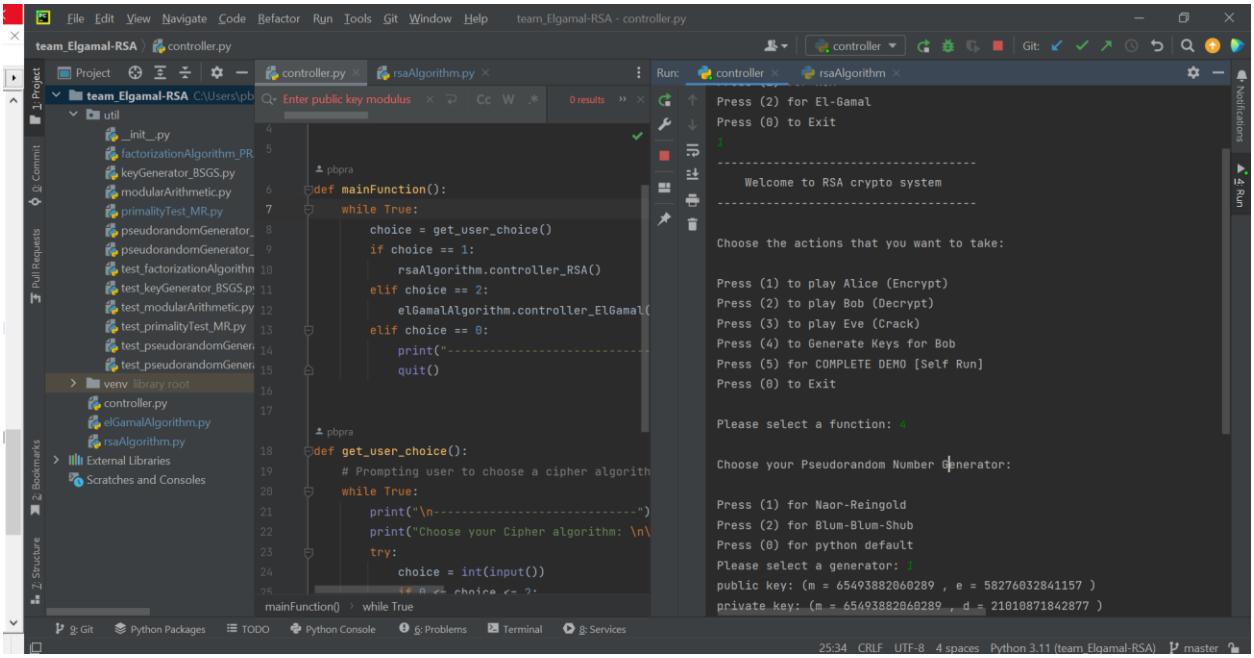

Gmail
in:sent
Compose
Inbox 1,668
Starred
Snoozed
Sent
Drafts 4
More
Labels
Request for a Secure Communication using RSA
Proteek Bose
Dear Bob, I hope this message finds you well. I am writing to request your public key, as I have an important message to share with you that requires a high lev
1:03 AM (4 minutes ago)
Anup Sindagi
Hey Alice, Sure, here is my RSA Public Key (e,n): (54687164876441, 117268426749787) Please send me the encrypted message. Best, Bob
1:05 AM (2 minutes ago)
Proteek Bose <proby@bu.edu>
to Anup, Sri
1:07 AM (0 minutes ago)
Dear Bob, Please find below the encrypted message:
Encrypted Message: 78293629063916
Regards,
Alice

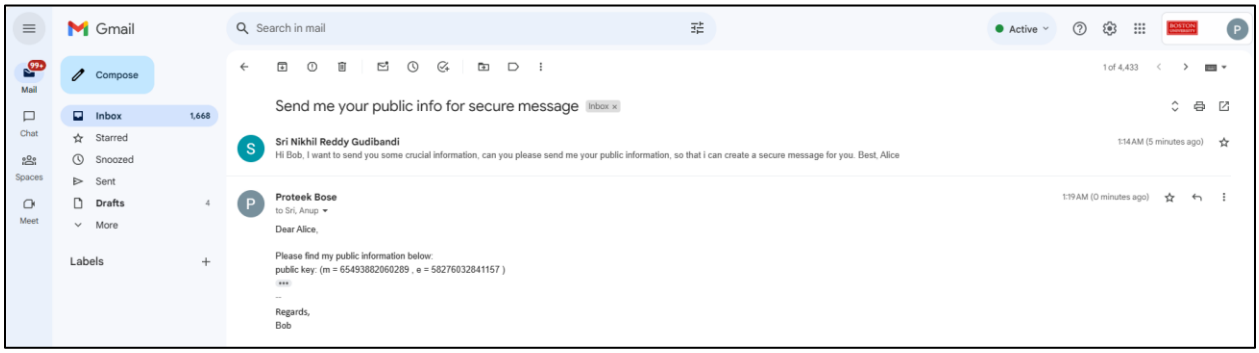

Gmail
Search in mail
Compose
Inbox 1,668
Starred
Snoozed
Sent
Drafts 4
More
Labels
Request for a Secure Communication using RSA
Proteek Bose
Dear Bob, I hope this message finds you well. I am writing to request your public key, as I have an important message to share with you that requires a high lev
1:03 AM (5 minutes ago)
Anup Sindagi
Hey Alice, Sure, here is my RSA Public Key (e,n): (54687164876441, 117268426749787) Please send me the encrypted message. Best, Bob
1:05 AM (3 minutes ago)
Proteek Bose
Dear Bob, Please find below the encrypted message: Encrypted Message: 78293629063916
1:07 AM (1 minute ago)
Anup Sindagi
to me, Sri
1:09 AM (0 minutes ago)
Dear Alice,
I successfully decrypted and received your message.
Best,
Bob

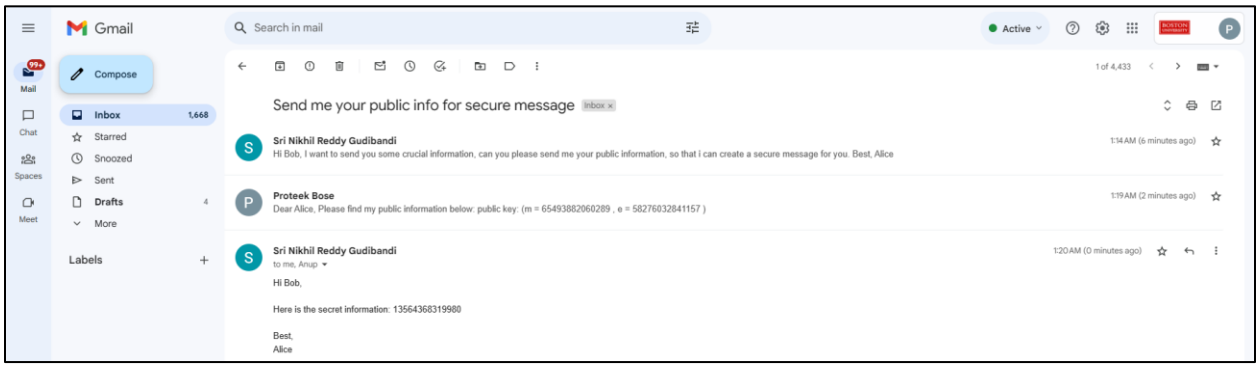

Gmail
Search in mail
Compose
Inbox 1,668
Starred
Snoozed
Sent
Drafts 4
More
Labels
Request for a Secure Communication using RSA
Proteek Bose
Dear Bob, I hope this message finds you well. I am writing to request your public key, as I have an important message to share with you that requires a high lev
1:03 AM (8 minutes ago)
Anup Sindagi
Hey Alice, Sure, here is my RSA Public Key (e,n): (54687164876441, 117268426749787) Please send me the encrypted message. Best, Bob
1:05 AM (7 minutes ago)
Proteek Bose
Dear Bob, Please find below the encrypted message: Encrypted Message: 78293629063916
1:07 AM (4 minutes ago)
Anup Sindagi
Dear Alice, I successfully decrypted and received your message. Best, Bob
1:09 AM (3 minutes ago)
Sri Nikhil Reddy Gudibandi
to Anup, me
1:12 AM (0 minutes ago)
Dear Morons,
This is shame that i cracked the secured message with ease, which is '55555'
Next time i request you to please use large keys, give me some challenge
Worse,
Eve

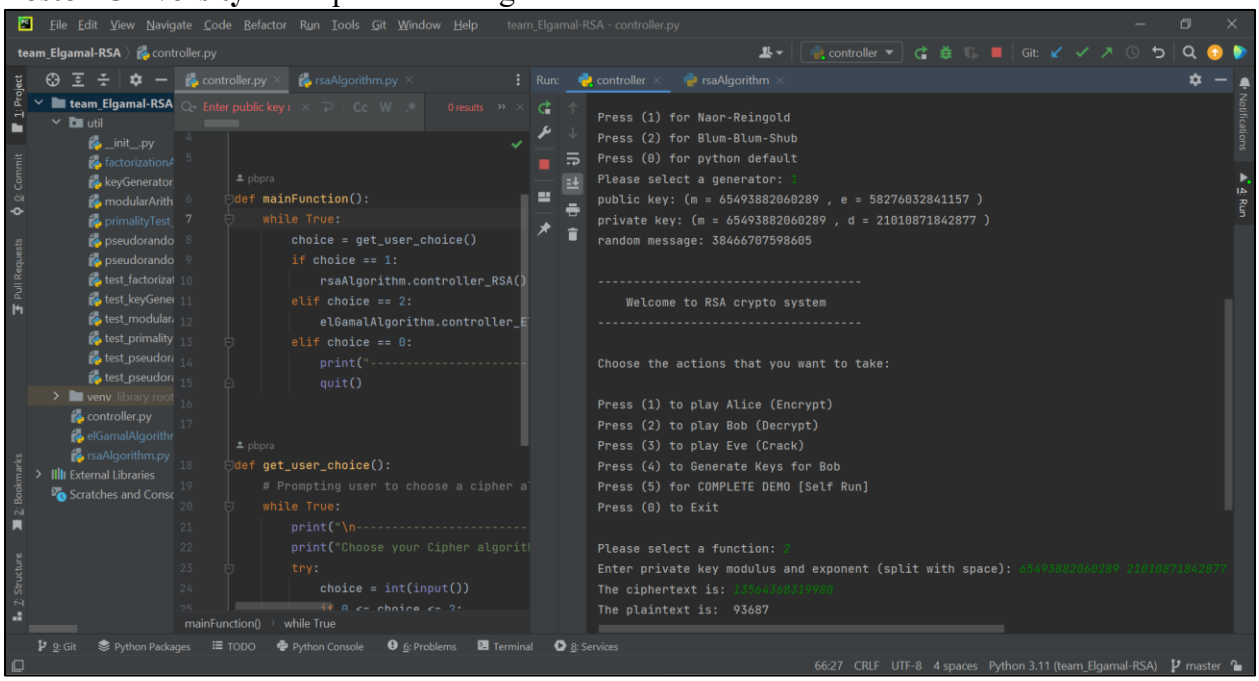
Playing Bob: The message that I received from Alice was “93687”

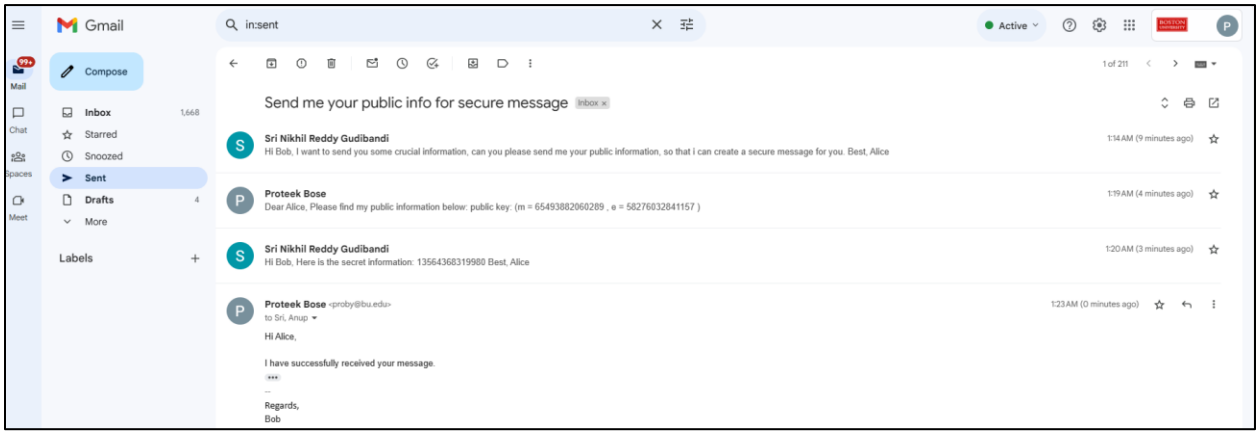


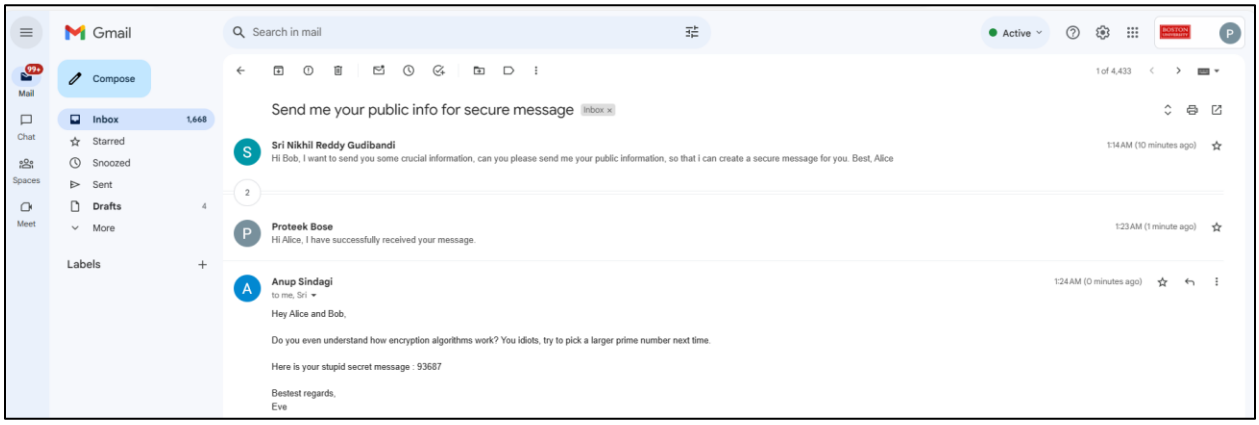




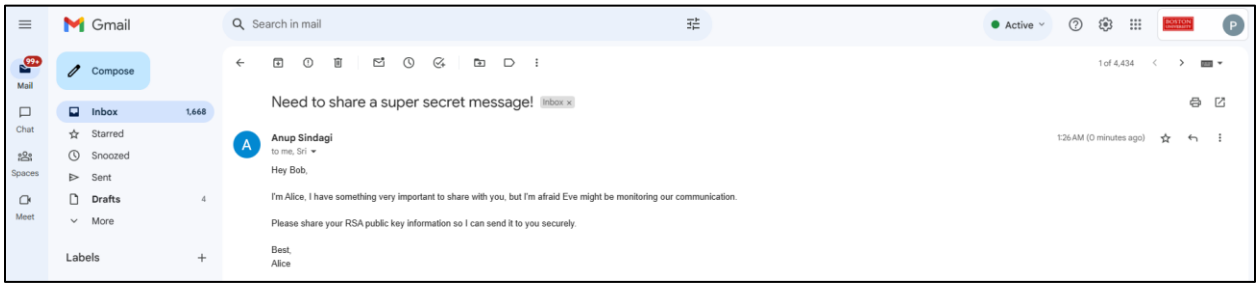


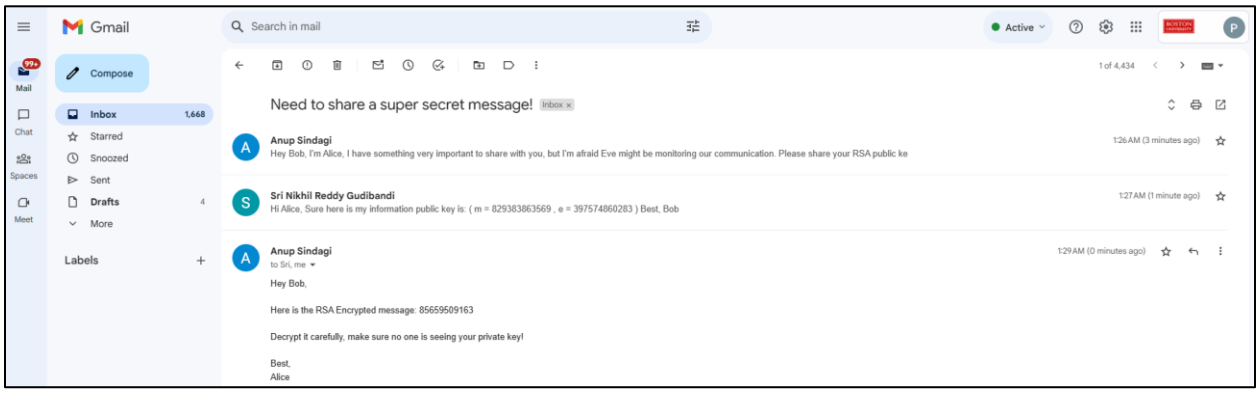

Press (1) for Naor-Reingold
Press (2) for Blum-Blum-Shub
Press (0) for python default
Please select a generator: 1
public key: (m = 65493882060289 , e = 58276032841157)
private key: (m = 65493882060289 , d = 21010871842877)
random message: 38466707598605
Welcome to RSA crypto system
Choose the actions that you want to take:
Press (1) to play Alice (Encrypt)
Press (2) to play Bob (Decrypt)
Press (3) to play Eve (Crack)
Press (4) to Generate Keys for Bob
Press (5) for COMPLETE DEMO [Self Run]
Press (0) to Exit
Please select a function: 2
The ciphertext is: 135643683199
The plaintext is: 93687

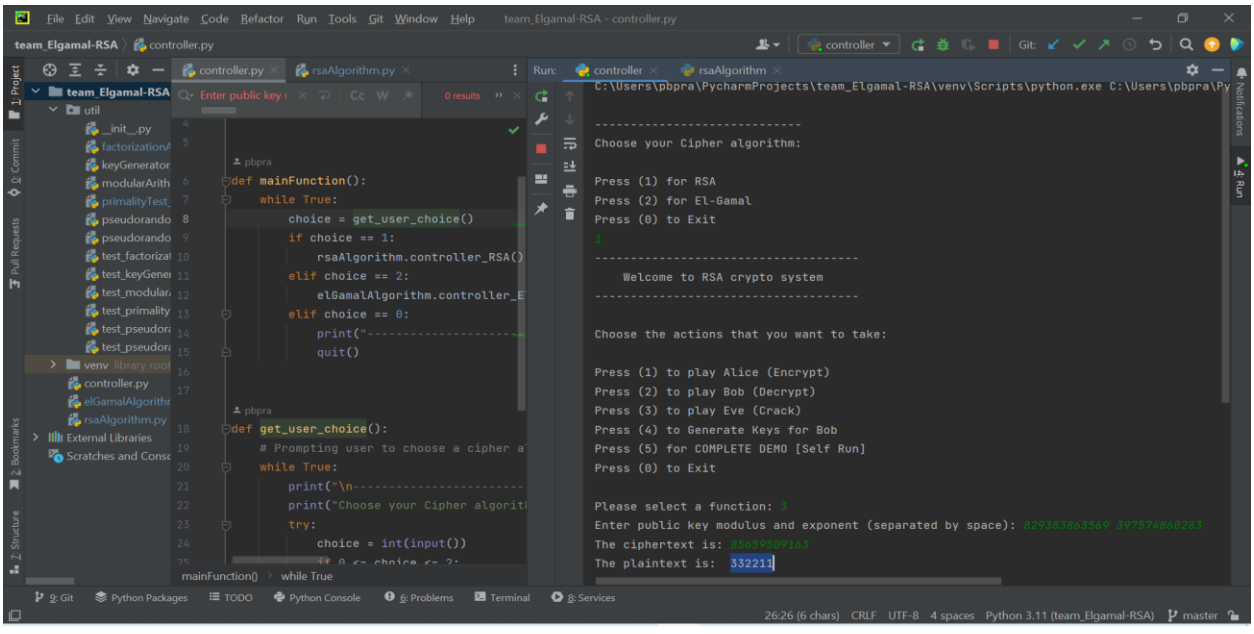

Send me your public info for secure message
Sri Nikhil Reddy Gudibandi
Hi Bob, I want to send you some crucial information, can you please send me your public information, so that I can create a secure message for you. Best, Alice
Proteek Bose
Dear Alice, Please find my public information below: public key: (m = 65493882060289 , e = 58276032841157)
Sri Nikhil Reddy Gudibandi
Hi Bob, Here is the secret information: 135643683199 80 Best, Alice
Proteek Bose
Hi Alice,
I have successfully received your message.
Regards,
Bob

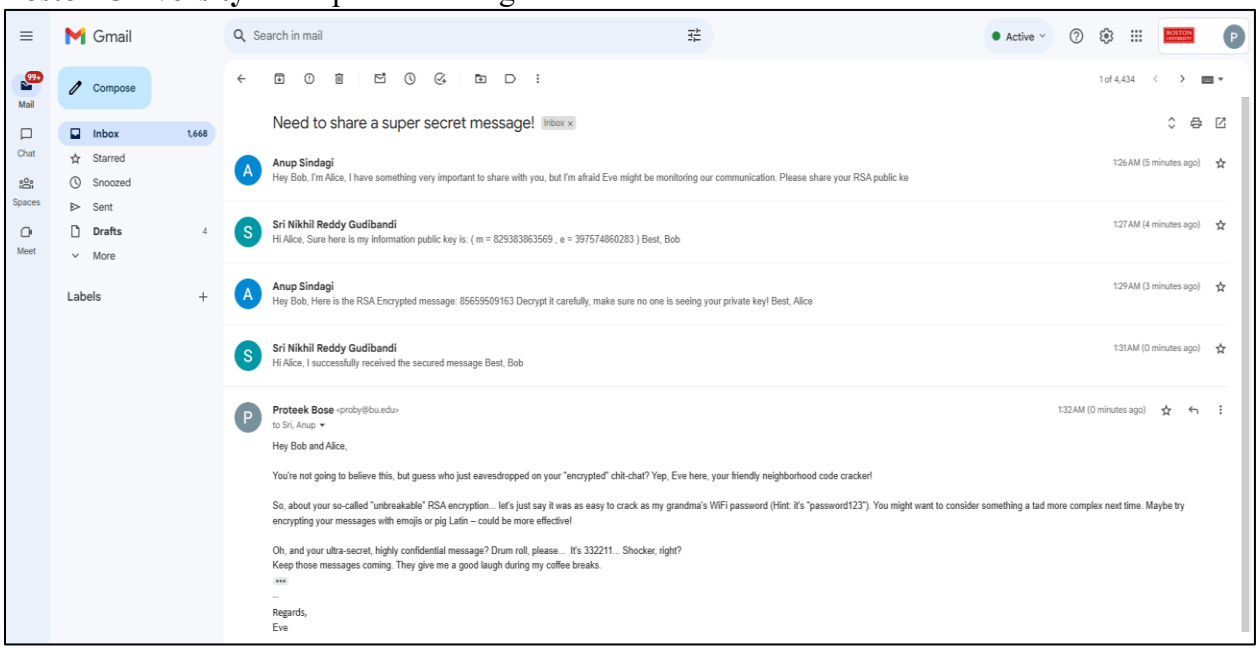

Send me your public info for secure message
Sri Nikhil Reddy Gudibandi
Hi Bob, I want to send you some crucial information, can you please send me your public information, so that I can create a secure message for you. Best, Alice
Proteek Bose
Hi Alice, I have successfully received your message.
Anup Sindagi
Hey Alice and Bob,
Do you even understand how encryption algorithms work? You idiots, try to pick a larger prime number next time.
Here is your stupid secret message : 93687
Bestest regards,
Eve

Playing Eve: The message that I cracked is “332211”









References

- Crypto.stackexchange.com: (2023) | When to use RSA and when ElGamal asymmetric encryption. https://crypto.stackexchange.com/questions/41006/why-is-it-claimed-that-elgamal-is-worse-than-rsa
- Quora: (2023) | What is the difference between an RSA algorithm and an ElGamal algorithm? https://www.quora.com/What-is-the-difference-between-an-RSA-algorithm-and-an-ElGamal-algorithm
- EUDL: (2018) | Comparative Analysis of RSA and ElGamal Cryptographic Public-key Algorithms. https://www.researchgate.net/publication/326373441_Comparative_Analysis_of_RSA_and_ElGamal_Cryptographic_Public-key_Algorithms
- Information Security Stack Exchange: (2018) | New PGP key: RSA/RSA or DSA/ElGamal? https://security.stackexchange.com/questions/72581/new-pgp-key-rsa-rsa-or-dsa-elgamal
- IEEE Xplore: (2017) | A comparison between RSA and ElGamal based untraceable blind signature schemes. https://ieeexplore.ieee.org/document/7400705
- wanl.blue: (2023) | SoloSavings – Home. https://www.bluesunpv.com/bluesun-hjt-n-type-solar-panel-585w-580w-solar-panel-585-w-585watt_p522.html