Jigyas Sharma

Dr. Han Wang

EECS 700

15 November 2023

Homework 3: Report

Task 1:

a) To produce a random nonnegative matrix for Alice and Bob for their respective sizes, we use the numpy library.

```
# Alice generates a 5x8 matrix
A = np.random.randint(low=0, high=100, size=(5, 8))
# Bob generates an 8x4 matrix
B = np.random.randint(low=0, high=100, size=(8, 4))
```

b) The design of the cryptographic protocol that can be used to securely compute the product of AxB is as follows:
Step 1: Alice produces a key pair using paillier crypto system.
Step 2: Alice then sends her public key to Bob.
Step 3: Bob multiplies his matrix with some random number(in our case 4) before encrypting his matrix. So, even if Alice can decrypt this matrix, the information is redacted.
Step 4: Alice receives the encrypted matrix. Since, there is paillier encryption used, Alice can use the homomorphic properties to multiply the encrypted matrix with her plain text value. Alice, then sends this multiplied matrix back to Bob.
Step 5: Bob receives the multiplied matrix which he then divides by his random value(4) to reveal the actual answer of matrix multiplication.

During these steps Bob never finds out the value of Alice's matrix and Alice never finds out the value of Bob's matrix.

c) *Code for Alice and Bob's Computer is provided in the zip file as **Alice.py** and **Bob.py***

d) The screenshots below post the value. The first set of screenshots are with 512 key size and the second set is with 1024 key size. The print statement only reveals an object, however, when using 512 key size the computation takes less time and resources when compared to 1024 key size.

```
jigya@Darksst:~/Desktop/Github_Projects/Data-Privacy-and-Data-Security-Models/Encryption/Homomorphic_Encryption$ python3 Alice.py
Alice's Matrix A:
[[0 6 2 7 5 5 1 8]
 [2 0 2 8 2 8 5 7]
 [2 8 4 0 6 7 2 8]
 [5 5 9 8 9 1 9 5]
 [6 9 9 7 1 8 6 6]]
Alice's Server Started. Waiting for Connection...
Connected by ('127.0.0.1', 36264)
Last Ciphertext (Encrypted Matrix C):
[[<phe.paillier.EncryptedNumber object at 0x7fee07c3eb30>
  <phe.paillier.EncryptedNumber object at 0x7fee07c3ebf0>
  <phe.paillier.EncryptedNumber object at 0x7fee07c3ec50>
  <phe.paillier.EncryptedNumber object at 0x7fee07c3ecb0>]
 [<phe.paillier.EncryptedNumber object at 0x7fee07c3ed10>
  <phe.paillier.EncryptedNumber object at 0x7fee07c3ed70>
  <phe.paillier.EncryptedNumber object at 0x7fee07c3edd0>
  <phe.paillier.EncryptedNumber object at 0x7fee07c3ee30>]
 [<phe.paillier.EncryptedNumber object at 0x7fee07c3ee90>
  <phe.paillier.EncryptedNumber object at 0x7fee07c3eef0>
  <phe.paillier.EncryptedNumber object at 0x7fee07c3ef50>
  <phe.paillier.EncryptedNumber object at 0x7fee07c3efb0>]
 [<phe.paillier.EncryptedNumber object at 0x7fee07c3f010>
  <phe.paillier.EncryptedNumber object at 0x7fee07c3f070>
  <phe.paillier.EncryptedNumber object at 0x7fee07c3f0d0>
  <phe.paillier.EncryptedNumber object at 0x7fee07c3f130>]
 [<phe.paillier.EncryptedNumber object at 0x7fee07c3f190>
  <phe.paillier.EncryptedNumber object at 0x7fee07c3f1f0>
  <phe.paillier.EncryptedNumber object at 0x7fee07c3f250>
  <phe.paillier.EncryptedNumber object at 0x7fee07c3f2b0>]]
Decrypted Product (A x B):
[[ 520.  408.  772.  284.]
 [ 604.  540.  904.  388.]
 [ 580.  284.  776.  340.]
 [ 612.  696. 1276.  472.]
 [ 656.  768. 1236.  524.]]
Sending decrypted result to Bob.
```

Figure 1: Alice's Computer 512 key size

```
jigya@Darksst:~/Desktop/Github_Projects/Data-Privacy-and-Data-Security-Models/Encryption/Homomorphic_Encryption$ python3 Bob.py
Connected to Alice's Server
Bob's Matrix B:
[[1 7 2 3]
 [1 1 3 1]
 [1 4 8 2]
 [2 9 8 2]
 [4 0 6 2]
 [7 3 8 5]
 [5 2 9 4]
 [6 1 3 1]]
Encrypted Blinded Matrix B:
[[<phe.paillier.EncryptedNumber object at 0x7fe7f204a680>
  <phe.paillier.EncryptedNumber object at 0x7fe7f204a9b0>
  <phe.paillier.EncryptedNumber object at 0x7fe7b4918640>
  <phe.paillier.EncryptedNumber object at 0x7fe7b4918850>]
 [<phe.paillier.EncryptedNumber object at 0x7fe7b49188e0>
  <phe.paillier.EncryptedNumber object at 0x7fe7b49189d0>
  <phe.paillier.EncryptedNumber object at 0x7fe7b4918a60>
  <phe.paillier.EncryptedNumber object at 0x7fe7b4918ac0>]
 [<phe.paillier.EncryptedNumber object at 0x7fe7b4918b20>
  <phe.paillier.EncryptedNumber object at 0x7fe7b4918b80>
  <phe.paillier.EncryptedNumber object at 0x7fe7b4918be0>
  <phe.paillier.EncryptedNumber object at 0x7fe7b4918c40>]
 [<phe.paillier.EncryptedNumber object at 0x7fe7b4918ca0>
  <phe.paillier.EncryptedNumber object at 0x7fe7b4918d00>
  <phe.paillier.EncryptedNumber object at 0x7fe7b4918d60>
  <phe.paillier.EncryptedNumber object at 0x7fe7b4918dc0>]
 [<phe.paillier.EncryptedNumber object at 0x7fe7b4918e20>
  <phe.paillier.EncryptedNumber object at 0x7fe7b4918e80>
  <phe.paillier.EncryptedNumber object at 0x7fe7b4918ee0>
  <phe.paillier.EncryptedNumber object at 0x7fe7b4918f40>]
 [<phe.paillier.EncryptedNumber object at 0x7fe7b4918fa0>
  <phe.paillier.EncryptedNumber object at 0x7fe7b4919000>
  <phe.paillier.EncryptedNumber object at 0x7fe7b4919060>
  <phe.paillier.EncryptedNumber object at 0x7fe7b49190c0>]
 [<phe.paillier.EncryptedNumber object at 0x7fe7b4919120>
  <phe.paillier.EncryptedNumber object at 0x7fe7b4919180>
  <phe.paillier.EncryptedNumber object at 0x7fe7b49191e0>
  <phe.paillier.EncryptedNumber object at 0x7fe7b4919240>]
 [<phe.paillier.EncryptedNumber object at 0x7fe7b49192a0>
  <phe.paillier.EncryptedNumber object at 0x7fe7b4919300>
  <phe.paillier.EncryptedNumber object at 0x7fe7b4919360>
  <phe.paillier.EncryptedNumber object at 0x7fe7b49193c0>]]
Final Result (Product of A and B):
[[130. 102. 193.  71.]
 [151. 135. 226.  97.]
 [145.  71. 194.  85.]
 [153. 174. 319. 118.]
 [164. 192. 309. 131.]]
```

Figure 2: Bob's Computer 512 key size

```
jigya@Darksst:~/Desktop/Github_Projects/Data-Privacy-and-Data-Security-Models/Encryption/Homomorphic_Encryption$ python3 Alice.py
Alice's Matrix A:
[[5 9 1 8 7 5 0 3]
 [3 7 1 4 4 4 1 9]
 [5 7 8 3 1 0 9 5]
 [0 6 1 4 3 9 6 8]
 [2 2 3 2 8 7 2 5]]
Alice's Server Started. Waiting for Connection...
Connected by ('127.0.0.1', 58854)
Last Ciphertext (Encrypted Matrix C):
[[<phe.paillier.EncryptedNumber object at 0x7f286c63e920>
  <phe.paillier.EncryptedNumber object at 0x7f286c63e9e0>
  <phe.paillier.EncryptedNumber object at 0x7f286c63ea40>
  <phe.paillier.EncryptedNumber object at 0x7f286c63eaa0>]
 [<phe.paillier.EncryptedNumber object at 0x7f286c63eb00>
  <phe.paillier.EncryptedNumber object at 0x7f286c63eb60>
  <phe.paillier.EncryptedNumber object at 0x7f286c63ebc0>
  <phe.paillier.EncryptedNumber object at 0x7f286c63ec20>]
 [<phe.paillier.EncryptedNumber object at 0x7f286c63ec80>
  <phe.paillier.EncryptedNumber object at 0x7f286c63ece0>
  <phe.paillier.EncryptedNumber object at 0x7f286c63ed40>
  <phe.paillier.EncryptedNumber object at 0x7f286c63eda0>]
 [<phe.paillier.EncryptedNumber object at 0x7f286c63ee00>
  <phe.paillier.EncryptedNumber object at 0x7f286c63ee60>
  <phe.paillier.EncryptedNumber object at 0x7f286c63eec0>
  <phe.paillier.EncryptedNumber object at 0x7f286c63ef20>]
 [<phe.paillier.EncryptedNumber object at 0x7f286c63ef80>
  <phe.paillier.EncryptedNumber object at 0x7f286c63efe0>
  <phe.paillier.EncryptedNumber object at 0x7f286c63f040>
  <phe.paillier.EncryptedNumber object at 0x7f286c63f0a0>]]
Decrypted Product (A x B):
[[700. 672. 736. 688.]
 [624. 708. 764. 640.]
 [876. 588. 968. 836.]
 [624. 792. 740. 648.]
 [548. 608. 568. 488.]]
Sending decrypted result to Bob.
```

Figure 3: Alice's Computer 1024 key size

```
jigya@Darksst:~/Desktop/Github_Projects/Data-Privacy-and-Data-Security-Models/Encryption/Homomorphic_Encryption$ python3 Bob.py
Connected to Alice's Server
Bob's Matrix B:
[[4 1 8 3]
 [9 7 6 4]
 [9 1 7 7]
 [1 1 4 9]
 [5 6 2 3]
 [2 5 2 0]
 [4 4 5 5]
 [4 8 9 7]]
Encrypted Blinded Matrix B:
[[<phe.paillier.EncryptedNumber object at 0x7fc588fae680>
  <phe.paillier.EncryptedNumber object at 0x7fc588fae9b0>
  <phe.paillier.EncryptedNumber object at 0x7fc54b874640>
  <phe.paillier.EncryptedNumber object at 0x7fc54b874850>]
 [<phe.paillier.EncryptedNumber object at 0x7fc54b8748e0>
  <phe.paillier.EncryptedNumber object at 0x7fc54b8749d0>
  <phe.paillier.EncryptedNumber object at 0x7fc54b874a60>
  <phe.paillier.EncryptedNumber object at 0x7fc54b874ac0>]
 [<phe.paillier.EncryptedNumber object at 0x7fc54b874b20>
  <phe.paillier.EncryptedNumber object at 0x7fc54b874b80>
  <phe.paillier.EncryptedNumber object at 0x7fc54b874be0>
  <phe.paillier.EncryptedNumber object at 0x7fc54b874c40>]
 [<phe.paillier.EncryptedNumber object at 0x7fc54b874ca0>
  <phe.paillier.EncryptedNumber object at 0x7fc54b874d00>
  <phe.paillier.EncryptedNumber object at 0x7fc54b874d60>
  <phe.paillier.EncryptedNumber object at 0x7fc54b874dc0>]
 [<phe.paillier.EncryptedNumber object at 0x7fc54b874e20>
  <phe.paillier.EncryptedNumber object at 0x7fc54b874e80>
  <phe.paillier.EncryptedNumber object at 0x7fc54b874ee0>
  <phe.paillier.EncryptedNumber object at 0x7fc54b874f40>]
 [<phe.paillier.EncryptedNumber object at 0x7fc54b874fa0>
  <phe.paillier.EncryptedNumber object at 0x7fc54b875000>
  <phe.paillier.EncryptedNumber object at 0x7fc54b875060>
  <phe.paillier.EncryptedNumber object at 0x7fc54b8750c0>]
 [<phe.paillier.EncryptedNumber object at 0x7fc54b875120>
  <phe.paillier.EncryptedNumber object at 0x7fc54b875180>
  <phe.paillier.EncryptedNumber object at 0x7fc54b8751e0>
  <phe.paillier.EncryptedNumber object at 0x7fc54b875240>]
 [<phe.paillier.EncryptedNumber object at 0x7fc54b8752a0>
  <phe.paillier.EncryptedNumber object at 0x7fc54b875300>
  <phe.paillier.EncryptedNumber object at 0x7fc54b875360>
  <phe.paillier.EncryptedNumber object at 0x7fc54b8753c0>]]
Final Result (Product of A and B):
[[175. 168. 184. 172.]
 [156. 177. 191. 160.]
 [219. 147. 242. 209.]
 [156. 198. 185. 162.]
 [137. 152. 142. 122.]]
```

Figure 4: Bob's Computer 1024 key size

e) Scenario 1: When a plaintext value is multiplied by an encrypted value it gives out the correct matrix value. This can be done by calculating the matrix multiplication value from

Alice and Bob's original matrix. Since, in the above situation Bob's matrix was encrypted however, Alice was able to multiply the values with her plaintext. We can also verify this by storing the normal matrix multiplication and comparing it to the decrypted matrix multiplication. (Refer to figure 5)

```
[46] # Perform the encrypted matrix multiplication
     C = np.empty([5, 4], dtype=object)  # Initialize an empty matrix for the result

     for i in range(5):  # Iterate over rows of A
         for j in range(4):  # Iterate over columns of encrypted_B
             sum = public_key.encrypt(0)  # Start with an encrypted 0
             for k in range(8):  # Iterate over the elements in the row/column pair
                 # Convert A[i][k] to a standard Python integer and then perform scalar multiplication
                 product = encrypted_B[k][j] * int(A[i][k])  # Scalar multiplication
                 sum = sum + product  # Homomorphic addition
             C[i][j] = sum

     # C now contains the encrypted result of A x B
```

```
# Decrypting the result matrix C
decrypted_C = np.zeros([5, 4])  # Initialize an empty matrix for the decrypted result

for i in range(5):
    for j in range(4):
        decrypted_C[i][j] = private_key.decrypt(C[i][j])

print("Decrypted Matrix (Product of A and B):")
print(decrypted_C)
# Performing normal matrix multiplication of A and B
normal_product = np.dot(A, B)
print("Multiplication Values without Encryption")
print(normal_product)
```

```
Decrypted Matrix (Product of A and B):
[[22487. 22724. 17145. 21731.]
 [24791. 22627. 27072. 30930.]
 [32752. 28475. 28872. 34730.]
 [32567. 27288. 26067. 33878.]
 [25714. 28086. 26906. 30224.]]
Multiplication Values without Encryption
[[22487 22724 17145 21731]
 [24791 22627 27072 30930]
 [32752 28475 28872 34730]
 [32567 27288 26067 33878]
 [25714 28086 26906 30224]]
```

Figure 5: Cross checking encrypted and normal matrix multiplication

Scenario 2: When trying to multiply two encrypted values, for example both the matrices are encrypted, the value will not hold. Consider, this simple program below which tries to multiply two encrypted values at which point the compiler throws an error. (Refer figure 6)

```python
from phe import paillier

# Generate keys
public_key, private_key = paillier.generate_paillier_keypair()

# Encrypt two numbers
encrypted_num1 = public_key.encrypt(5)
encrypted_num2 = public_key.encrypt(3)

# Attempt to multiply two encrypted numbers
try:
    result = encrypted_num1 * encrypted_num2
    print("Encrypted Multiplication Result:", result)
except Exception as e:
    print("Error occurred:", e)
```

```
Error occurred: Good luck with that...
```

Figure 6: Two encrypted values multiplication throws an error

Task 2:

Input boolean vectors:
Alice : 0 0 0 0 1 1 1 0 0 0
Bob: 1 1 1 1 0 0 0 0 1 0
SFDL PROGRAM:

```
/*
 * Calculate the scalar product of two Boolean vectors
 */
program ScalarProduct {
   const N = 10;
   type Bool = Int<1>; // Assuming Bool as 1-bit integer
   type Vector = Bool[N];
   type AliceInput = Vector;
   type BobInput = Vector;
   type ScalarProductResult = Int<32>; // 32-bit integer to hold the scalar product
   type Input = struct {AliceInput alice, BobInput bob};
   type Output = struct {ScalarProductResult result};

   function Output output(Input input) {
      int<32> sum = 0;
      int<8> i;

      for (i = 0 to N-1) {
         sum = sum + (input.alice[i] * input.bob[i]); // Self-referencing assignment
      }

      Output result;
      result.result = sum;
      return result;
   }
}
```