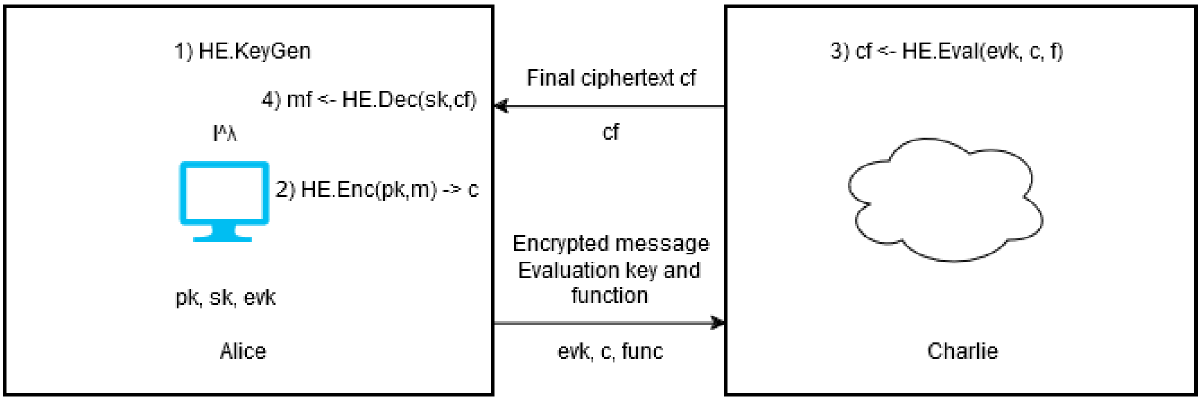
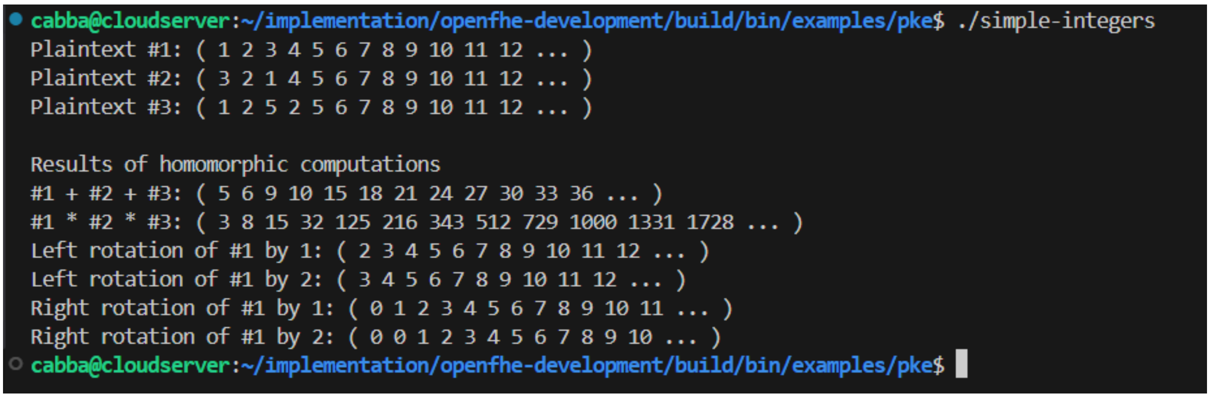
Task – 1



1. Alice generates keys (pk, sk, evk) using λ as security parameter.
2. Message m is encrypted using pk and ciphertext c is created.
3. Ciphertext c is sent to the cloud (Charlie) along with function f and evaluation key.
4. Charlie computes function f on ciphertext c and then encrypts it with eval key.
5. Final ciphertext cf is sent to Alice
6. Alice decrypts it using her private key and gets message back with function f applied on it.

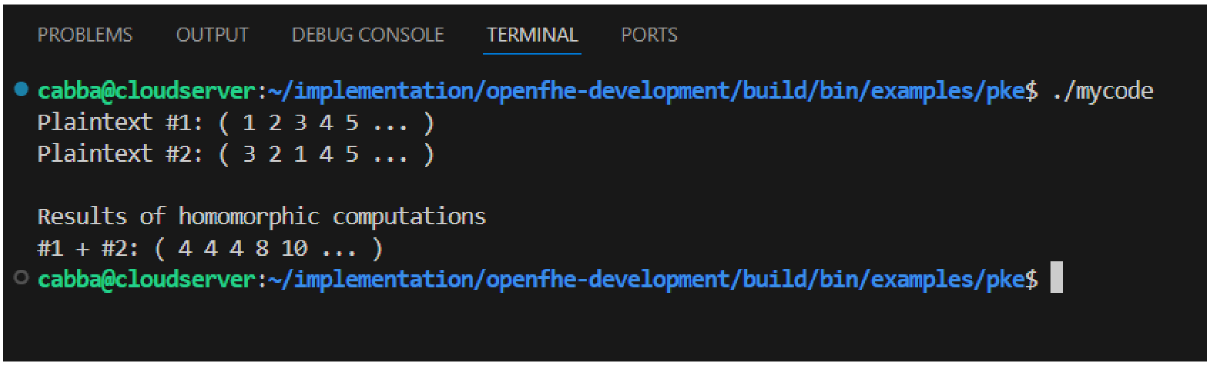
Task – 2



```
● cabba@cloudserver:~/implementation/openfhe-development/build/bin/examples/pke$ ./simple-integers
  Plaintext #1: ( 1 2 3 4 5 6 7 8 9 10 11 12 ... )
  Plaintext #2: ( 3 2 1 4 5 6 7 8 9 10 11 12 ... )
  Plaintext #3: ( 1 2 5 2 5 6 7 8 9 10 11 12 ... )

  Results of homomorphic computations
  #1 + #2 + #3: ( 5 6 9 10 15 18 21 24 27 30 33 36 ... )
  #1 * #2 * #3: ( 3 8 15 32 125 216 343 512 729 1000 1331 1728 ... )
  Left rotation of #1 by 1: ( 2 3 4 5 6 7 8 9 10 11 12 ... )
  Left rotation of #1 by 2: ( 3 4 5 6 7 8 9 10 11 12 ... )
  Right rotation of #1 by 1: ( 0 1 2 3 4 5 6 7 8 9 10 11 ... )
  Right rotation of #1 by 2: ( 0 0 1 2 3 4 5 6 7 8 9 10 ... )
○ cabba@cloudserver:~/implementation/openfhe-development/build/bin/examples/pke$ 
```

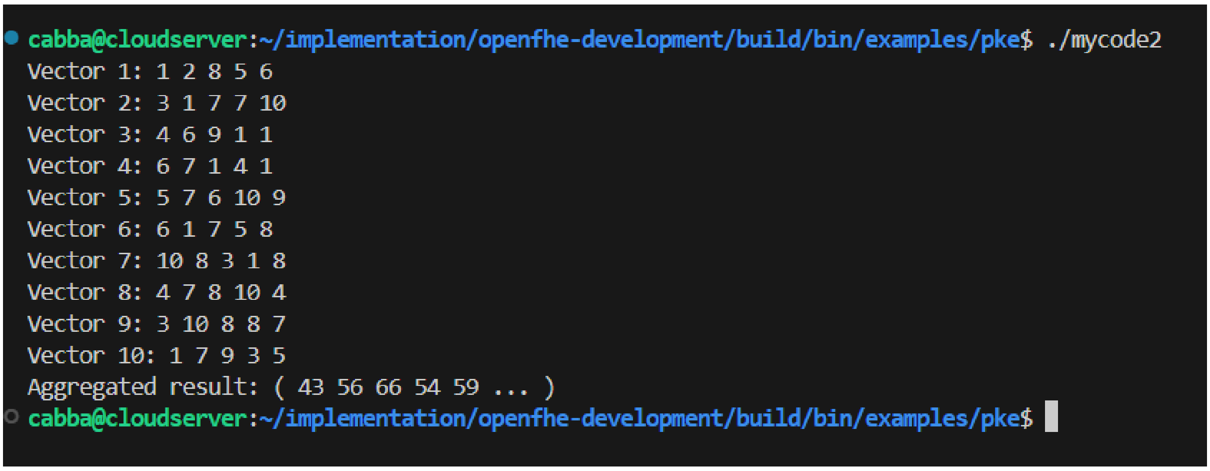
./simple-integers output



```
● cabba@cloudserver:~/implementation/openfhe-development/build/bin/examples/pke$ ./mycode
  Plaintext #1: ( 1 2 3 4 5 ... )
  Plaintext #2: ( 3 2 1 4 5 ... )

  Results of homomorphic computations
  #1 + #2: ( 4 4 4 8 10 ... )
○ cabba@cloudserver:~/implementation/openfhe-development/build/bin/examples/pke$ 
```

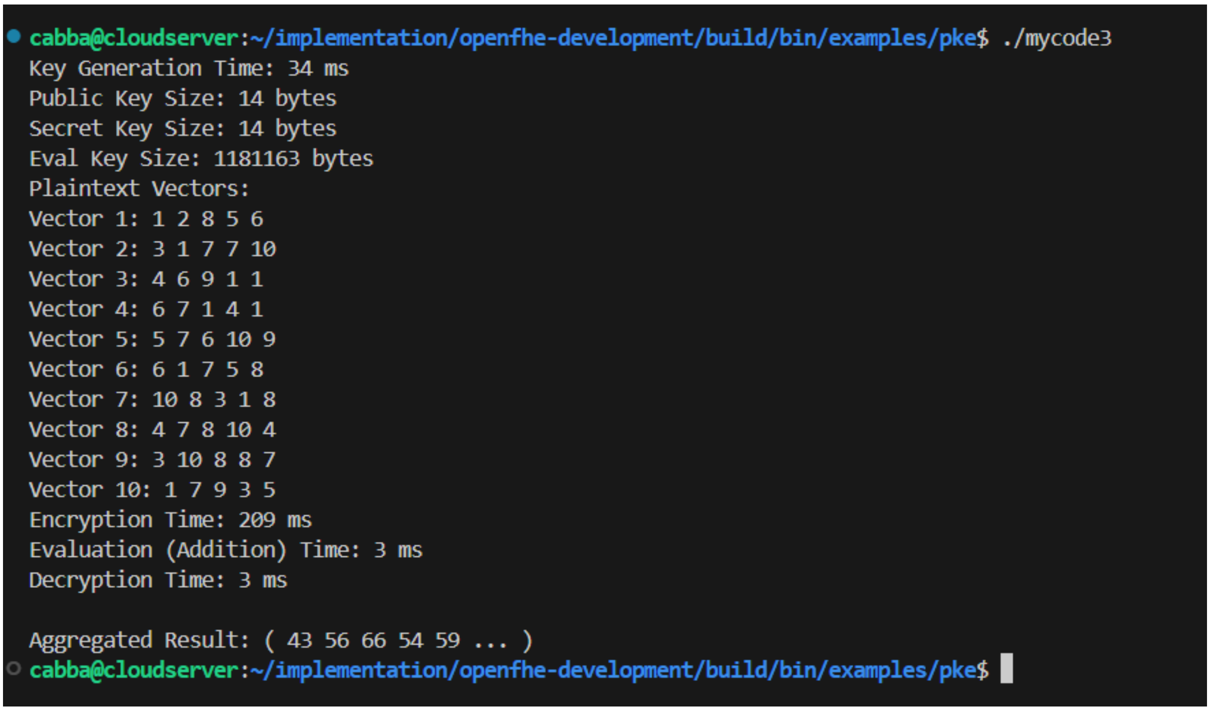
./mycode output



```
● cabba@cloudserver:~/implementation/openfhe-development/build/bin/examples/pke$ ./mycode2
  Vector 1: 1 2 8 5 6
  Vector 2: 3 1 7 7 10
  Vector 3: 4 6 9 1 1
  Vector 4: 6 7 1 4 1
  Vector 5: 5 7 6 10 9
  Vector 6: 6 1 7 5 8
  Vector 7: 10 8 3 1 8
  Vector 8: 4 7 8 10 4
  Vector 9: 3 10 8 8 7
  Vector 10: 1 7 9 3 5
  Aggregated result: ( 43 56 66 54 59 ... )
○ cabba@cloudserver:~/implementation/openfhe-development/build/bin/examples/pke$ 
```

./mycode with 10 vectors output

Task – 3



```
cabba@cloudserver:~/implementation/openfhe-development/build/bin/examples/pke$ ./mycode3
Key Generation Time: 34 ms
Public Key Size: 14 bytes
Secret Key Size: 14 bytes
Eval Key Size: 1181163 bytes
Plaintext Vectors:
Vector 1: 1 2 8 5 6
Vector 2: 3 1 7 7 10
Vector 3: 4 6 9 1 1
Vector 4: 6 7 1 4 1
Vector 5: 5 7 6 10 9
Vector 6: 6 1 7 5 8
Vector 7: 10 8 3 1 8
Vector 8: 4 7 8 10 4
Vector 9: 3 10 8 8 7
Vector 10: 1 7 9 3 5
Encryption Time: 209 ms
Evaluation (Addition) Time: 3 ms
Decryption Time: 3 ms

Aggregated Result: ( 43 56 66 54 59 ... )
cabba@cloudserver:~/implementation/openfhe-development/build/bin/examples/pke$
```

./mycode with 10 vectors and some metrics

Key Generation Time: 34 ms

Public Key Size: 14 bytes

Secret Key Size: 14 bytes

Eval Key Size: 1.8 mega bytes

Encryption Time: 209 ms

Evaluation (Addition) Time: 3 ms

Decryption Time: 3 ms

Based on this output we can see that public key and secret key are similar in size, they actually had only a few bits different which I manually checked and thought they are the same.

Running diff

diff public_key.txt secret_key.txt

1c1

< 0x625bd82f54f0

\ No newline at end of file

```
---

> 0x625bd829c1e0

\ No newline at end of file
```

The eval key is the biggest being 1.8MB in size. The key generation size is 34ms which is quite fast as well. Encryption time is slower around 200ms. Evaluation time is quite small which was unexpected for me. Decryption is fast again 3ms.

Encryption is slow because it involves a lot of complex mathematical computations such as generating noise etc.

Evaluation is fast since its usually a simple function like addition in this case.