Batch: A3 Roll No. : 16010121045

Experiment No. 10

Title: Homomorphic encryption and its applications

Objective: To implement Homomorphic encryption

Expected Outcome of Experiment:

CO	Outcome
CO4	Realize advances in the field of cryptography

Books/ Journals/ Websites referred:

Abstract: -

Related Theory:

What is homomorphic encryption?

Homomorphic encryption is a new approach that allows you to process and compute directly on *encrypted* data. That means you don't have the risk that comes with decrypting it. This approach is based on mathematical algorithms which compute the action. With this technology, data can remain secure both:

- While it is processed
- Without ever sharing the encryption keys or needing to unencrypt the data

Types of homomorphic encryption

Homomorphic encryption is still an emerging technology, but it is already categorized into three different types of homomorphic encryption, described below. The fundamental difference between the different types is the frequency of mathematical operations that can be performed on the ciphertext.

- **Partially Homomorphic Encryption (PHE).** In PHE, 'partially' means that only a select mathematical functions can be processed on encrypted values. So only one action — either addition or multiplication — is allowed to be performed an unlimited number of times on the ciphertext.
- **Somewhat Homomorphic Encryption (SHE). '**Somewhat' is more general than PHE in that it supports homomorphic operations with additions *and* multiplications. However, the primary con here is that it can perform only a limited number of operations.
- **Fully Homomorphic Encryption (FHE).** Here, **'**fully' is the operative word. Where PHE and SHE have limited operations, fully homomorphic encryption has the capability of using both addition and multiplication with no limit on the number of times they're performed on the ciphertext.

Process followed with screenshots:

```
python3 -u "/Users/pargat/Documents/COLLEGE/AC/Programs/rsa.py"
pargat@Router Programs % python3 -u "/Users/pargat/Documents/COLLEGE/AC/Programs/rsa.py"
Enter value of p (Prime number): 13
Enter value of q (Prime number): 11
Public key <e,n> = <119,143>
Private key <d,n> = <119,143>
Enter plain text: 12
Encrypted text: [23, 138]
12
pargat@Router Programs % python3 -u "/Users/pargat/Documents/COLLEGE/AC/Programs/rsa.py"
Enter value of p (Prime number): 13
Enter value of q (Prime number): 11
Public key <e,n> = <119,143>
Private key <d,n> = <119,143>
Enter plain text: 10
Encrypted text: [23, 22]
10
pargat@Router Programs % python3 -u "/Users/pargat/Documents/COLLEGE/AC/Programs/rsa.py"
Enter value of p (Prime number): 13
Enter value of q (Prime number): 11
Public key <e,n> = <119,143>
Private key <d,n> = <119,143>
Enter plain text: 120
Encrypted text: [23, 138, 22]
120
pargat@Router Programs %
```

Postlab questions:

1. **Comment on advantages and disadvantages of Homomorphic Encryption.**

Key Advantages of Homomorphic Encryption

With homomorphic encryption, organizations can establish a higher standard of data security without breaking business processes or application functionality. These organizations can ensure data privacy, while still deriving intelligence from their sensitive data. This is incredibly important these days in the wake of GDPR.

Use cases of homomorphic encryption include cloud workload protection (or "lift and shift" to cloud), aggregate analytics (privacy preserving encryption), information supply chain consolidation (containing your data to mitigate breach risk), and automation and orchestration (operating and triggering off of encrypted data for machine-to-machine communication).

Disadvantages of Homomorphic Encryption

But homomorphic encryption still falls short in the real world. In fact, many security folks would consider it complete BS. It is still, despite dramatic improvement over the years, incredibly slow and non-performant, making it a non-starter for most business applications. There are important open questions about its underlying encryption strength as well with recent analysis suggesting that the method leaks privacy information and may be subject to

compromise. And organizations cannot run ad-hoc/discovery-based queries with its methodology.

One of the most significant disadvantages is that homomorphic encryption requires either application modifications or dedicated and specialized client-server applications in order to make it work functionally. . And organizations cannot run ad-hoc/discovery-based queries with its methodology. This increases your total cost of ownership and distracts your organization from more important and strategic initiatives.

Perhaps even more significantly, there are important open questions about homomorphic encryption's underlying crypto strength with recent analysis suggesting that the method leaks privacy information and may be subject to compromise.

2. List 3 applications of Homomorphic Encryption.

Traditionally the business or user used the Advanced Encryption Standard (AES) which needed secret keys which led to security issues. With the public cloud comes the ease of encryption which, in turn, can help drive homomorphic encryption. That's because the cloud can:

- Directly operate on encrypted data.
- Return the data in encrypted format to the source of the data.

Homomorphic encryption is an emerging technology that is still in the works of being developed. Homomorphic encryption will benefit many organizations from healthcare to Supply Chain, and with traditional IT moving to the cloud makes homomorphic encryption a great secure technology to use in the future.

Conclusion: