

JCD 891 – MINOR PROJECT

Development of Fully Homomorphic Encryption Scheme

Submitted By: Tarun Chauhan
(2022JCS2670)

Submitted To: Prof. Ashok K Bhateja

Contents

- Problem Statement
- Homomorphic Encryption
- Why Homomorphic Encryption?
- Partially Homomorphic Encryption Schemes
 1. RSA
 2. GM Cryptosystem
 3. Paillier Cryptosystem
- Somewhat Homomorphic Encryption Schemes
 1. BGN Cryptosystem
- Limitations





Problem Statement

Problem - For a third party to work on any encrypted data, it needs to decrypt the data first. This creates a privacy issue for the user, as the third party service providers like cloud storage, file sharing, collaboration, servers can keep physically identifying elements of users long after the user has ended the relationship with the services.

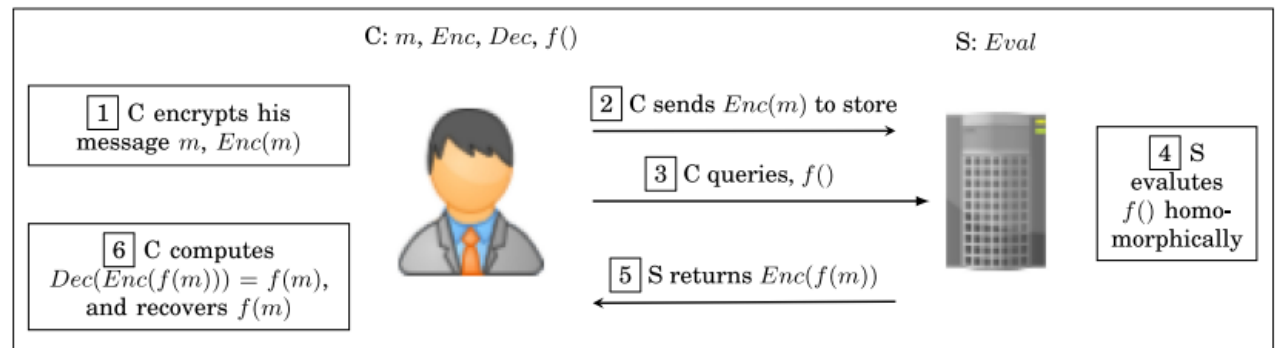
Solution – We need to develop a scheme that can operate on encrypted data. This scheme is referred to as Homomorphic Encryption scheme.

Homomorphic Encryption

- Homomorphic encryption is a form of encryption that allows computations to be performed on encrypted data without first having to decrypt it.
- An encryption scheme is called homomorphic over an operation " * ", if it supports the following equation:

$$E(m1) * E(m2) = E(m1 * m2), \forall m1, m2 \in M$$

- An HE scheme is primarily characterized by four operations: KeyGen, Enc, Dec, and Eval.



Why Homomorphic Encryption?

- Confidentiality Problem
- Ability to compute over ciphertext instead of plaintext
- One could use information without knowing the content of that information
- Privacy guaranteed

Homomorphic Encryption is of three types:

1. Partially Homomorphic Encryption(PHE)
2. Somewhat Homomorphic Encryption(SWHE)
3. Fully Homomorphic Encryption(FHE)

Partially Homomorphic Encryption Schemes

1. RSA

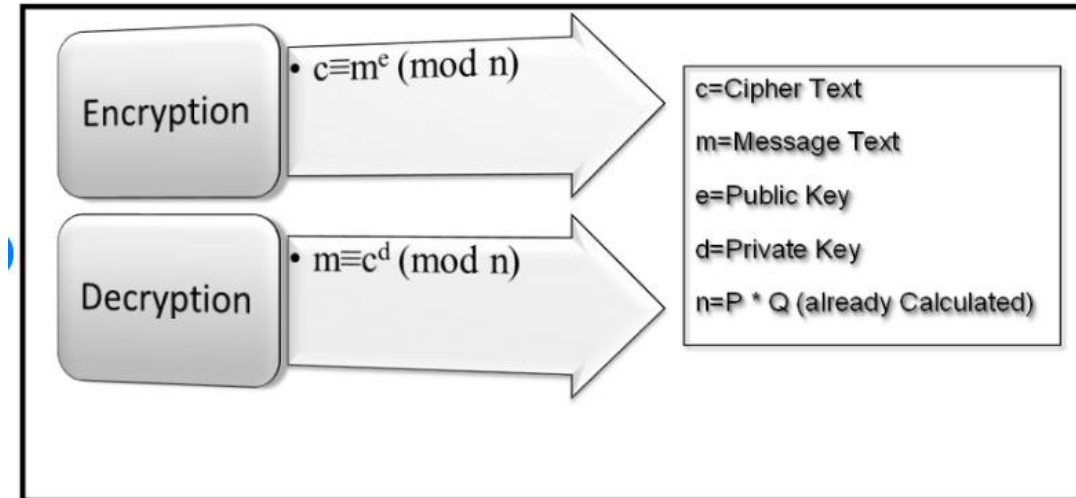
Security Assumption - Hardness of Factoring Problem

Homomorphic Property:

$m_1, m_2 \in M$

$$\begin{aligned} E(m_1) * E(m_2) &= (m_1^e \pmod n) * (m_2^e \pmod n) \\ &= (m_1 * m_2)^e \pmod n \\ &= E(m_1 * m_2) \end{aligned}$$

Homomorphic over multiplication operation.



Continued...

2. GM Cryptosystem

Security Assumption – Hardness of Quadratic Residuosity Problem

KeyGen : $n = pq$, x - a quadratic non-residue modulo n

Public key = (x, n) ; Private key = (p, q)

Encryption : message is encrypted bit by bit, y_i is a quadratic non-residue value

$$c_i = E(m_i) = y_i^2 x^{m_i} \pmod{n}, \quad \forall m_i = \{0, 1\},$$

Decryption: To decrypt the ciphertext c_i , one decides whether c_i is a quadratic residue modulo n or not; if so, m_i returns 0, or else m_i returns 1.

Homomorphic Property:

$$\begin{aligned} E(m_1) * E(m_2) &= (y_1^2 x^{m_1} \pmod{n}) * (y_2^2 x^{m_2} \pmod{n}) \\ &= (y_1 * y_2)^2 x^{m_1+m_2} \pmod{n} = E(m_1 + m_2). \end{aligned}$$

Continued...

3. Paillier Cryptosystem

Security Assumption : Composite Residuosity Problem

KeyGen : public key = (n,g) private key = (p,q)

Encryption: For each m choose a random r,

$$c = E(m) = g^m r^n \pmod{n^2}.$$

Decryption :

$$D(c) = \frac{L(c^\lambda \pmod{n^2})}{L(g^\lambda \pmod{n^2})} \pmod{n} = m,$$

Homomorphic Property:

$$\begin{aligned} E(m_1) * E(m_2) &= (g^{m_1} r_1^n \pmod{n^2}) * (g^{m_2} r_2^n \pmod{n^2}) \\ &= g^{m_1+m_2} (r_1 * r_2)^n \pmod{n^2} = E(m_1 + m_2). \end{aligned}$$

Homomorphic over addition operation.

Somewhat Homomorphic Encryption Schemes

BGN Cryptosystem

Security Assumption: Subgroup Decision Problem

KeyGen: public key = (n, G, G_1, e, g, h) , here $n = q_1 * q_2$ and $e : G \times G \rightarrow G_1$

Private key = (q_1)

Encryption: choose a random r

$$c = E(m) = g^m h^r \mod n.$$

Decryption: first compute $c' = c^{q_1} = (g^m h^r)^{q_1}$ and let $g' = g^{q_1}$

$$m = D(c) = \log_{g'} c'.$$

Homomorphic Property:

Addition - Homomorphic addition of plaintexts m_1 and m_2 using ciphertexts $E(m_1) = c_1$ and $E(m_2) = c_2$

$$c = c_1 c_2 h^r = (g^{m_1} h^{r_1})(g^{m_2} h^{r_2}) h^r = g^{m_1+m_2} h^{r'}$$

where $r = r_1 + r_2 + r$ and it can be seen that $m_1 + m_2$ can be easily recovered from the resulting ciphertext c .

Multiplication - The homomorphic multiplication of messages m_1 and m_2 using the ciphertexts $c_1 = E(m_1)$ and $c_2 = E(m_2)$ are computed as follows:

$$\begin{aligned} c &= e(c_1, c_2) h_1^r = e(g^{m_1} h^{r_1}, g^{m_2} h^{r_2}) h_1^r \\ &= g_1^{m_1 m_2} h_1^{m_1 r_2 + r_2 m_1 + \alpha q_2 r_1 r_2 + r} = g_1^{m_1 m_2} h_1^{r'} \end{aligned}$$

It is seen that r is uniformly distributed like r and so $m_1 m_2$ can be correctly recovered from resulting ciphertext c .

Limitations

Computational complexity

Performance is often a disadvantage.

Do not provide verifiable computing.

Number of homomorphic functions
and the number of times they can be
applied is limited.

Thank You!