Fully Homomorphic Encryption

DSC 4110/6111: Modern Cryptography and AI/ML Security

Dr. Laltu Sardar

School of Data Science,
Indian Institute of Science Education and Research Thiruvananthapuram (IISER TVM)

August 14, 2026

Table of Content

Paillier Encryption Scheme

Key Generation:
Choose two large primes p, q and compute

$$n = pq.$$

Set

$$\lambda = \text{lcm}(p-1, q-1).$$

Choose

$$g = n + 1.$$

The public key is

$$(n, g),$$

and the private key is

$$(\lambda).$$

Encryption

For a message

$$m \in \mathbb{Z}_n,$$

choose a random

$$r \in \mathbb{Z}_n^*.$$

The ciphertext is

$$\boxed{c = g^m r^n \pmod{n^2}}$$

Since we choose $g = n + 1$,

$$\boxed{c = (n+1)^m r^n \pmod{n^2}.}$$

Decryption

Define

$$L(u) = \frac{u-1}{n}.$$

The decrypted message is

$$\boxed{m = L(c^{\lambda} \bmod n^2)\left[L(g^{\lambda} \bmod n^2)\right]^{-1} \pmod{n}.}$$

For $g = n+1$, we will show that this recovers the original message m.

Step 1: Raise the Ciphertext to λ

Starting from

$$c = (n+1)^m r^n,$$

raise both sides to λ:

$$c^\lambda = (n+1)^{m\lambda} r^{n\lambda} \pmod{n^2}.$$

We consider the two terms separately.

Step 2: Simplify $(n+1)^{m\lambda}$

Using the binomial theorem,

$$(n+1)^{m\lambda} = 1 + m\lambda n + \binom{m\lambda}{2} n^2 + \cdots$$

Modulo n^2, all terms containing n^2 or higher powers vanish.
Therefore,

$$\boxed{(n+1)^{m\lambda} \equiv 1 + m\lambda n \pmod{n^2}.}$$

Step 3: Simplify $r^{n\lambda}$

Since

$$\lambda = \text{lcm}(p-1, q-1),$$

Carmichael's theorem gives

$$r^{\lambda} \equiv 1 \pmod{n}.$$

Therefore,

$$r^{n\lambda} = (r^{\lambda})^{n} \equiv 1^{n} \pmod{n^2}.$$

Hence,

$$\boxed{r^{n\lambda} \equiv 1 \pmod{n^2}.}$$

Step 4: Combine the Results

We have

$$c^{\lambda} = (n+1)^{m\lambda} r^{n\lambda}.$$

Using the previous results,

$$c^{\lambda} \equiv (1 + m\lambda n)(1) \pmod{n^2}.$$

Thus,

$$\boxed{c^{\lambda} \equiv 1 + m\lambda n \pmod{n^2}.}$$

Step 5: Apply the L Function

Recall

$$L(u) = \frac{u-1}{n}.$$

Therefore,

$$L(c^\lambda) = \frac{c^\lambda - 1}{n}.$$

Since

$$c^\lambda \equiv 1 + m\lambda n \pmod{n^2},$$

we obtain

$$\boxed{L(c^\lambda) \equiv m\lambda \pmod{n}.}$$

Step 6: Evaluate $L(g^\lambda)$

Since

$$g = n + 1,$$

we have

$$g^\lambda = (n+1)^\lambda.$$

Again, by the binomial theorem,

$$(n+1)^\lambda \equiv 1 + \lambda n \pmod{n^2}.$$

Therefore,

$$L(g^\lambda) = \frac{g^\lambda - 1}{n} \equiv \boxed{\lambda \pmod{n}}.$$

Step 7: Decryption

The decryption formula is

$$D(c) = L(c^{\lambda})L(g^{\lambda})^{-1} \pmod{n}.$$

Substituting the results,

$$D(c) = (m\lambda)\lambda^{-1} \pmod{n}.$$

Since λ is invertible modulo n,

$$\boxed{D(c) = m \pmod{n}.}$$

Correctness Result

Starting with

$$c = (n+1)^m r^n,$$

we obtain

$$L(c^\lambda) \equiv m\lambda \pmod{n}$$

and

$$L(g^\lambda) \equiv \lambda \pmod{n}.$$

Therefore,

$$\begin{aligned} D(c) &= L(c^\lambda)L(g^\lambda)^{-1} \\ &\equiv m\lambda\lambda^{-1} \\ &\equiv m \pmod{n}. \end{aligned}$$

$$\boxed{\therefore\ D(E(m,r)) = m \pmod{n}}$$

Thus, Paillier decryption is correct.

Key Idea

The correctness proof relies on three facts:

$$\boxed{(n+1)^k \equiv 1 + kn \pmod{n^2}}$$

$$\boxed{r^\lambda \equiv 1 \pmod{n}}$$

and

$$\boxed{L(u) = \frac{u-1}{n}.}$$

Together, they remove the random factor r^n and recover the message m.

Paillier Encryption Scheme

an Additive Homomorphic Encryption

Paillier Encryption Scheme

Paillier Encryption is a public-key encryption scheme with **additive homomorphism**.
The plaintext space is

$$\mathbb{Z}_n$$

and the ciphertext space is

$$\mathbb{Z}_{n^2}^*.$$

Choose two large primes

$$p,\ q$$

and set

$$\boxed{n = pq}.$$

The key idea is to encode the message m in the exponent: g^m, while adding a random factor to hide m.

$$\boxed{c = g^m r^n \bmod n^2}$$

where

$$r \xleftarrow{\$} \mathbb{Z}_n^*.$$

Main Property

Paillier supports homomorphic **addition** of plaintexts through multiplication of ciphertexts.

Paillier: Key Generation

Choose two large primes $p,\ q$ and compute $n = pq$.
Let

$$\lambda = \mathrm{lcm}(p-1, q-1).$$

Choose

$$g \in \mathbb{Z}^*_{n^2}$$

such that the required decryption condition holds.
Define

$$L(u) = \frac{u-1}{n}.$$

Compute

$$\mu = \left(L(g^\lambda \bmod n^2)\right)^{-1} \bmod n.$$

The keys are

$$\boxed{pk = (n, g)}$$

and

$$\boxed{sk = (\lambda, \mu).}$$

Paillier: Encryption and Decryption

Encryption

For

$$m \in \mathbb{Z}_n,$$

choose random

$$r \xleftarrow{\$} \mathbb{Z}_n^*.$$

Compute

$$\boxed{c = g^m r^n \bmod n^2}$$

Decryption

Given c, compute

$$u = c^\lambda \bmod n^2.$$

Then

$$\boxed{m = L(u)\,\mu \bmod n}$$

where

$$L(u) = \frac{u-1}{n}.$$

Thus,

$$\boxed{\mathrm{Dec}(\mathrm{Enc}(m)) = m}$$

Paillier: Additive Homomorphism

Let

$$c_1 = \mathrm{Enc}(m_1) = g^{m_1} r_1^n \bmod n^2$$

and

$$c_2 = \mathrm{Enc}(m_2) = g^{m_2} r_2^n \bmod n^2.$$

Multiply the ciphertexts:

$$\begin{aligned} c_1 c_2 &= g^{m_1} r_1^n g^{m_2} r_2^n \bmod n^2 \\ &= g^{m_1+m_2} (r_1 r_2)^n \bmod n^2. \end{aligned}$$

This is exactly an encryption of

$$m_1 + m_2.$$

Therefore,

$$\boxed{\mathrm{Enc}(m_1)\,\mathrm{Enc}(m_2) = \mathrm{Enc}(m_1 + m_2)}$$

Key Property

Paillier is **additively homomorphic**.

Paillier: Multiplication by a Known Constant

Paillier also allows multiplication of a plaintext by a **known constant**.
Suppose

$$c = \mathrm{Enc}(m).$$

For a known integer a,

$$c^a = (g^m r^n)^a = g^{am}(r^a)^n \bmod n^2.$$

Therefore,

$$\boxed{c^a = \mathrm{Enc}(am)}$$

This gives

$$\boxed{\mathrm{Enc}(m)^a = \mathrm{Enc}(am)}$$

Thus Paillier can evaluate expressions such as

$$a_1 m_1 + a_2 m_2 + \cdots + a_k m_k$$

when the coefficients a_i are known.

Important

Paillier does **not** provide general multiplication of two encrypted values.

Paillier: Small Numerical Example

For illustration, choose $p = 3, \quad q = 5.$

Then $n = pq = 15, \quad n^2 = 225.$

Choose

$$g = n + 1 = 16.$$

We have

$$\lambda = \text{lcm}(2, 4) = 4$$

and

$$\mu = 4.$$

Let the plaintexts be

$$m_1 = 7, \quad m_2 = 4.$$

Choose random values

$$r_1 = 2, \quad r_2 = 7.$$

Encryption gives

$$c_1 = 16^7 2^{15} \bmod 225 = 83,$$
$$c_2 = 16^4 7^{15} \bmod 225 = 223.$$

The Cloud multiplies the ciphertexts:

$$\boxed{c_1 c_2 \bmod 225 = 59}$$

No decryption is performed by the Cloud.

Paillier: Example of Homomorphic Addition

From the previous example, $m_1 = 7$, $m_2 = 4$ and the Cloud obtained $c = 59$.
The Cloud does not know the plaintext sum.

The User decrypts:

$$u = 59^4 \bmod 225.$$

Then

$$L(u) = \frac{u-1}{15}.$$

Using $\mu = 4$, the decryption gives

$$\boxed{m = L(u)\mu \bmod 15 = 11.}$$

But

$$7 + 4 = 11.$$

Therefore,

$$\boxed{\mathrm{Dec}(c_1 c_2) = m_1 + m_2 = 11.}$$

What Happened?

The Cloud performed only

$$c_1 c_2 \bmod n^2,$$

yet the User obtained the sum of the hidden plaintexts.

Other Additively Homomorphic Encryption Schemes

Paillier is one of the most well-known additively homomorphic encryption schemes, but it is not the only one.

- **Paillier (1999):** Based on composite residuosity; widely used as a standard example of additive HE.
- **Damgård–Jurik (2001):** Generalization of Paillier that supports a larger plaintext space.
- **Benaloh (1994):** An early probabilistic encryption scheme with additive homomorphic properties.
- **Goldwasser–Micali (1984):** An early probabilistic homomorphic encryption scheme supporting **XOR** of plaintext bits.
- **DGK:** A somewhat homomorphic scheme particularly efficient for comparison and small-integer computations.

Additive HE $\longrightarrow$ Addition without decrypting

Important

These schemes differ in their plaintext spaces, security assumptions, efficiency, and exact homomorphic capabilities.

Is Additive + Multiplicative HE Sufficient?

We Have Seen Additive and Multiplicative HE

Are They Sufficient for Outsourcing Linear Regression?

$$\boxed{y = mx + c}$$

Multiplication: $m \times x$ Addition: $mx + c$

Can we combine the two?

Are Additive and Multiplicative HE Sufficient?

Answer: Not necessarily.
For

$$\boxed{y = mx + c},$$

we need both

$$m \times x \qquad \text{and} \qquad (mx) + c.$$

We have seen schemes supporting these operations separately.

The problem is composition.
An additive HE scheme and a multiplicative HE scheme generally use different ciphertext spaces and different algebraic structures.
Thus, we cannot simply take

$$\mathrm{Enc}_{\mathrm{Mult}}(mx)$$

and feed it into an additive HE operation:

$$\mathrm{Enc}_{\mathrm{Mult}}(mx) \not\longrightarrow \mathrm{Enc}_{\mathrm{Add}}(mx + c).$$

Key Requirement

We need **one encryption scheme** in which addition and multiplication can be performed and composed on the same ciphertexts.

From PHE to Somewhat Homomorphic Encryption

The schemes we have seen so far are examples of **Partially Homomorphic Encryption (PHE)**.
They support one algebraic operation:

$$\boxed{\text{Additive PHE} \longrightarrow +}$$

or

$$\boxed{\text{Multiplicative PHE} \longrightarrow \times}$$

A natural next step is to construct a single scheme supporting **both** operations:

$$\boxed{\text{Enc}(m_1) + \text{Enc}(m_2) \longrightarrow \text{Enc}(m_1 + m_2)}$$

and

$$\boxed{\text{Enc}(m_1) \times \text{Enc}(m_2) \longrightarrow \text{Enc}(m_1 m_2)}$$

We can get it from

$$\boxed{\text{Somewhat Homomorphic Encryption (SHE)}}.$$

BGN Encryption Scheme

a Somewhat Homomorphic Encryption

BGN Homomorphic Encryption

Boneh–Goh–Nissim (BGN) is a pairing-based homomorphic encryption scheme.
Let

$$N = pq$$

where p, q are large primes.
Let G and G_T be groups of order N, with a bilinear pairing

$$\boxed{e : G \times G \to G_T}$$

Choose $g \in G$ and an element h from a subgroup of G of order q.
The public key contains

$$\boxed{(N, G, G_T, e, g, h)}$$

and the secret key is

$$\boxed{sk = q}.$$

The plaintext is chosen from a **small message space**.

BGN: Encryption and Decryption

Encryption

For a small plaintext

$$m \in \{0, 1, \ldots, T\},$$

choose random

$$r \xleftarrow{\$} \mathbb{Z}_N.$$

Compute

$$\boxed{C = g^m h^r}$$

The random factor h^r hides the message.

Decryption

Using the secret key q,

$$C^q = (g^m h^r)^q.$$

Since h has order q,

$$h^q = 1.$$

Therefore, $\boxed{C^q = (g^q)^m}$

Let $\hat{g} = g^q$.

Recover m using

$$\boxed{m = \log_{\hat{g}}(C^q)}$$

Since m is small, this discrete logarithm can be computed efficiently.

BGN: Homomorphic Addition

Suppose

$$C_1 = g^{m_1} h^{r_1}$$

and

$$C_2 = g^{m_2} h^{r_2}.$$

Multiply the ciphertexts:

$$\begin{aligned} C_1 C_2 &= g^{m_1} h^{r_1} g^{m_2} h^{r_2} \\ &= g^{m_1+m_2} h^{r_1+r_2}. \end{aligned}$$

This is a valid encryption of

$$m_1 + m_2.$$

Therefore,

$$\boxed{C_1 C_2 = \mathrm{Enc}(m_1 + m_2)}$$

Hence BGN supports **unlimited homomorphic additions**.

BGN: One Homomorphic Multiplication

The special feature of BGN is the bilinear pairing:

$$e : G \times G \to G_T.$$

For

$$C_1 = g^{m_1} h^{r_1}, \qquad C_2 = g^{m_2} h^{r_2},$$

compute

$$e(C_1, C_2).$$

By bilinearity,

$$\begin{aligned} e(C_1, C_2) &= e(g^{m_1} h^{r_1}, g^{m_2} h^{r_2}) \\ &= e(g, g)^{m_1 m_2} \cdot \text{random factors.} \end{aligned}$$

Thus the resulting ciphertext contains

$$\boxed{m_1 m_2}.$$

After this multiplication, the ciphertext is in G_T.

BGN therefore provides

$$\boxed{\text{arbitrary additions} + \text{one multiplication}}$$

BGN: A Simple Example

Consider two encrypted messages

$$m_1 = 2, \qquad m_2 = 3.$$

The Cloud receives

$$C_1 = \mathrm{Enc}(2), \qquad C_2 = \mathrm{Enc}(3).$$

Addition:

$$C_1 C_2 = \mathrm{Enc}(2+3) = \boxed{\mathrm{Enc}(5)}.$$

The Cloud can continue adding:

$$C_1 C_2 C_3 = \mathrm{Enc}(2+3+m_3).$$

Multiplication:
Using the pairing,

$$e(C_1, C_2) = \boxed{\mathrm{Enc}(2 \times 3)} = \boxed{\mathrm{Enc}(6)}$$

in the target group G_T.
Thus the Cloud can evaluate

$$\boxed{2 \times 3 + 4 + 5}$$

by performing one multiplication followed by arbitrary additions.

BGN: Why Is It Not FHE?

BGN supports

$$\boxed{\text{unlimited additions} + \text{one multiplication.}}$$

For example,

$$m_1 m_2 + m_3 + m_4$$

can be evaluated.
But a computation requiring

$$m_1 m_2 m_3$$

needs two sequential multiplications.
After the first multiplication,

$$G \times G \xrightarrow{e} G_T.$$

The resulting ciphertext is in G_T, and the BGN construction does not provide another homomorphic multiplication on that ciphertext.
Therefore,

$$\boxed{\text{BGN} \neq \text{FHE}}$$

BGN was an important step toward FHE because it demonstrated that **addition and multiplication can coexist in a single homomorphic encryption system**.

Limitation

Revisit Linear Regression

solving Outsourcing with BGN scheme

Outsourcing Linear Regression with BGN

Recall the model

$$\boxed{y = mx + c}.$$

The Owner wants to hide the model parameters m, c from the Cloud.
Using BGN, the Owner computes

$$C_m = \mathrm{Enc}(m), \qquad C_c = \mathrm{Enc}(c).$$

and sends

$$\boxed{C_m,\ C_c}$$

to the Cloud.

The User sends the query x to the Cloud.
The Cloud sees

$$\boxed{x,\ C_m,\ C_c}$$

but does not know m or c.

Computing mx over the Encrypted Model

The Cloud knows the query x.
Suppose

$$C_m = \text{Enc}(m).$$

BGN allows multiplication of the plaintext by a **known constant** through exponentiation:

$$\boxed{C_m^x = \text{Enc}(m)^x = \text{Enc}(mx)}$$

Therefore, the Cloud computes

$$\boxed{C_{mx} = C_m^x}$$

without learning m.

The plaintext operation

$$m \times x$$

has therefore been transformed into the ciphertext operation

$$\boxed{C_m^x}.$$

Adding the Encrypted Intercept

The Cloud already has

$$C_{mx} = \mathrm{Enc}(mx)$$

and

$$C_c = \mathrm{Enc}(c).$$

Using BGN's additive homomorphism,

$$\begin{aligned} C_y &= C_{mx} C_c \\ &= \mathrm{Enc}(mx)\,\mathrm{Enc}(c) \\ &= \boxed{\mathrm{Enc}(mx + c)}. \end{aligned}$$

Thus the Cloud has evaluated

$$\boxed{y = mx + c}$$

without learning m or c.

The Cloud sends

$$\boxed{C_y}$$

back to the User.

Complete BGN-Based Linear Regression

The complete computation is

$$\boxed{C_m = \mathrm{Enc}(m), \qquad C_c = \mathrm{Enc}(c)}$$

followed by

$$\boxed{C_m^x = \mathrm{Enc}(mx)}$$

and

$$\boxed{C_y = C_m^x C_c = \mathrm{Enc}(mx + c).}$$

Finally, the authorized party decrypts:

$$\boxed{\mathrm{Dec}(C_y) = mx + c.}$$

Hence,

$$\boxed{\text{Encrypted Model} + \text{Plaintext Query} \xrightarrow{\text{BGN}} \text{Encrypted Prediction}}$$

Observation

Revisit Linear Regression

solving Outsourcing with BGN scheme
Can we encrypt queries as well?

Can We Also Hide the User's Query?

The User's query x may also be sensitive.
Instead of sending x in plaintext, the User can compute

$$\boxed{C_x = \mathrm{Enc}(x)}$$

and send only C_x to the Cloud.
Now the Cloud has

$$\boxed{C_m = \mathrm{Enc}(m), \quad C_c = \mathrm{Enc}(c), \quad C_x = \mathrm{Enc}(x).}$$

The desired computation is

$$\boxed{\mathrm{Enc}(m) \times \mathrm{Enc}(x) \longrightarrow \mathrm{Enc}(mx)}$$

followed by

$$\boxed{\mathrm{Enc}(mx) + \mathrm{Enc}(c) \longrightarrow \mathrm{Enc}(mx + c).}$$

New Challenge

Now both m and x are encrypted. We need **ciphertext-ciphertext multiplication**.

Can BGN Compute the Encrypted Query?

BGN provides one ciphertext-ciphertext multiplication through the bilinear pairing.
Given

$$C_m = \mathrm{Enc}(m), \qquad C_x = \mathrm{Enc}(x),$$

the Cloud can compute

$$\boxed{e(C_m, C_x)}$$

and, by bilinearity, the resulting ciphertext contains a term corresponding to

$$\boxed{mx}.$$

Thus BGN can perform the required multiplication once.

However, the pairing maps

$$\boxed{G \times G \longrightarrow G_T.}$$

The resulting ciphertext is therefore in

$$G_T,$$

whereas the encrypted intercept

$$C_c = \mathrm{Enc}(c)$$

is in the original ciphertext group.

The Problem

BGN: Plaintext Query vs. Encrypted Query

Case 1: Plaintext query

$$C_m = \mathrm{Enc}(m), \quad C_c = \mathrm{Enc}(c), \quad x$$

The Cloud computes

$$C_m^x = \mathrm{Enc}(mx)$$

and then

$$C_m^x C_c = \mathrm{Enc}(mx + c).$$

BGN works naturally.

Case 2: Encrypted query

$$C_m, \quad C_c, \quad C_x = \mathrm{Enc}(x)$$

BGN can perform the multiplication

$$C_m, C_x \xrightarrow{e} \mathrm{Enc}(mx) \quad \text{in } G_T.$$

But the result cannot be directly combined with

$$C_c \in G.$$

References

Textbook:

1. Douglas R. Stinson and Maura Paterson. *Cryptography: Theory and Practice*. 4th Edition. Chapman & Hall/CRC Press, 2019. ISBN: 9780367148591.

Acknowledgment:

- The figures in this work were generated with the assistance of ChatGPT (OpenAI).

Dr. Laltu Sardar, Assistant Professor, IISER Thiruvananthapuram
laltu.sardar@iisertvm.ac.in, laltu.sardar.crypto@gmail.com
Course webpage: https://laltu-sardar.github.io/courses/26_crypto.html.