**what is the flow of working of holomorphic encryption**

### Flow of Working of Homomorphic Encryption (HE)

Homomorphic Encryption enables computations on encrypted data while keeping the data itself secure and confidential. Here’s the typical workflow:

### ****1. Key Generation****

* **Keys Created**:
  + **Public Key**: Used to encrypt the data.
  + **Private Key**: Used to decrypt the result after computation.
* A key generation algorithm generates these keys securely.

### ****2. Data Encryption****

* **Input Data** (Plaintext): Sensitive data to be processed (e.g., numbers, text).
* **Encryption**: Using the public key, the plaintext is encrypted into **ciphertext**, which looks like random data and is unintelligible.
  + Example: Encrypt(Plaintext, Public Key) → Ciphertext.

### ****3. Encrypted Computation****

* **Processing on Ciphertext**: Without decrypting the ciphertext, computations (e.g., addition, multiplication) are performed directly on the encrypted data.
  + Example:
    - Ciphertext1 = E(50,000)
    - Ciphertext2 = E(60,000)
    - Computation: E(50,000) + E(60,000) = E(110,000)
* **Key Advantage**: The result of the computation is still encrypted and secure.

### ****4. Decryption****

* **Decryption**: The computed ciphertext is decrypted using the private key to obtain the plaintext result.
  + Example: Decrypt(Ciphertext, Private Key) → Plaintext Result.
  + If the computation result was E(110,000), decryption gives 110,000.

### ****Example Workflow****

#### Scenario: Compute Encrypted Sum of Two Salaries

1. **Key Generation**: Create public/private keys.
2. **Encryption**:
   * Salary A = 50,000 → E(50,000).
   * Salary B = 60,000 → E(60,000).
3. **Computation** (on encrypted data):
   * Sum = E(50,000) + E(60,000) = E(110,000).
4. **Decryption**:
   * Decrypt(E(110,000)) → 110,000.

### ****Types of Operations Supported****

Homomorphic encryption schemes can support:

1. **Partially Homomorphic Encryption (PHE)**: Supports either addition or multiplication (but not both).
   * Example: RSA supports multiplication; Paillier supports addition.
2. **Fully Homomorphic Encryption (FHE)**: Supports both addition and multiplication, enabling arbitrary computations on encrypted data.

### ****Applications****

* **Healthcare**: Privacy-preserving analytics on medical records.
* **Finance**: Secure data processing for fraud detection.
* **Cloud Computing**: Perform computations on encrypted data stored in untrusted environments.

Would you like more details on specific encryption schemes or their applications?

4o

Yes, that's precisely what homomorphic encryption achieves. It allows you to effectively "access" the results of computations on data without ever decrypting the data itself during the computation process.

Here's why this is significant:

* **Traditional Encryption:** With traditional encryption, if you want to perform any operation on encrypted data, you must first decrypt it. This creates a vulnerability because the data is exposed during the decryption and computation phase.
* Homomorphic Encryption: Homomorphic encryption overcomes this limitation. It enables computations to be carried out directly on the encrypted data. The result of these computations is also encrypted. Only the party with the decryption key can decrypt the final result to see the outcome of the operations.

**Key Points:**

* **No Decryption by the Computing Party:** The party performing the computations (e.g., a cloud server) never has access to the decryption key and, therefore, cannot decrypt the data.
* Meaningful Computations on Encrypted Data: The homomorphic properties of the encryption scheme ensure that the computations performed on the encrypted data produce a meaningful result that corresponds to the same computations performed on the original data.
* **Decryption by the Data Owner:** Only the data owner, who possesses the decryption key, can decrypt the final result and access the outcome of the computations.

**In simpler terms:**

Homomorphic encryption allows you to send your data to someone else for processing without ever revealing the actual data to them. They can perform computations on it, and you can get the result back, all while your data remains confidential.

This is a powerful concept with significant implications for privacy and security in various applications, such as cloud computing, secure data sharing, and privacy-preserving machine learning.

Sources and related content

then how can ai access the data without decrypting it can you explain



You're right to focus on this seemingly paradoxical aspect. It's the core of how homomorphic encryption works. The AI doesn't "access" the data in the traditional sense of reading or understanding its value. Instead, it manipulates the encrypted representation of the data.

Here's a more detailed explanation:

**Data Transformation:** The encryption process transforms the original data (plaintext) into a completely different form (ciphertext). This transformation is not arbitrary; it's designed with specific mathematical properties.

**Structure Preservation:** The crucial property of homomorphic encryption is that certain mathematical operations performed on the ciphertext correspond to operations performed on the plaintext. This means that if you perform an addition on two encrypted numbers, the result, when decrypted, will be the same as if you had added the original numbers before encrypting them.

**Blind Manipulation:** The AI model operates on this ciphertext, performing mathematical operations according to its algorithm. It's essentially manipulating encoded representations of the data without ever "seeing" the actual values.

**Mathematical Analogy:** Imagine you have two numbers, 5 and 3. You apply a function to them, say, "multiply by 2 and add 1."

* 1. f(5) = 11
  2. f(3) = 7

Now, if you add the results: 11 + 7 = 18.

If you first add the original numbers: 5 + 3 = 8, and then apply the function: f(8) = 17.

In this simple example, the function f is not homomorphic for addition because f(5) + f(3) ≠ f(5 + 3).

Homomorphic encryption uses much more complex functions where this property does hold for certain operations (addition, multiplication).

**Result Decryption:** The result of the AI's computations is also in encrypted form. Only when this encrypted result is decrypted using the appropriate key does the actual outcome of the computation become visible.

**Key Idea:**

The AI is not working with the data itself but with a transformed version that retains certain mathematical relationships. These relationships allow computations to be performed "blindly" on the encrypted data, with the results being meaningful once decrypted.

This is what makes homomorphic encryption so powerful: it enables computation without decryption, preserving privacy while allowing data to be processed.