### **Final Overview with Blockchain, Hybrid Encryption, and MPC**

1. **Local Model Training on Nodes**
   * Each of the 1000 nodes trains a machine learning model on its local dataset, resulting in a local update (e.g., weights or gradients).
   * Instead of sending these updates directly, nodes apply hybrid encryption and MPC to protect their individual updates.
2. **Hybrid Encryption: Combining MPC with Homomorphic Encryption**
   * **MPC Step**: Each node splits its local update into random “shares” using an MPC technique like *Shamir’s Secret Sharing*. These shares are distributed among a subset of nodes and stored on the blockchain.
   * **Homomorphic Encryption Step**: Nodes encrypt their shares using homomorphic encryption (e.g., Paillier encryption). This allows computations (like summing values) on the encrypted shares without needing decryption.
   * **Blockchain’s Role**: Each encrypted share is recorded as a transaction on the blockchain, creating an immutable and decentralized record of each node’s contribution.
3. **Blockchain for Key and Share Management**
   * Nodes use blockchain to manage cryptographic keys and share references securely. Private keys or decryption keys are distributed in pieces across nodes (or stored in a secure, decentralized way on the blockchain).
   * Each node retrieves and verifies the necessary keys or shares from the blockchain, ensuring authenticity without a central authority.
4. **Secure Aggregation Using Blockchain and Smart Contracts**
   * Smart contracts on the blockchain manage the secure aggregation process. The contract verifies that each node has submitted its encrypted update and all necessary shares before aggregating.
   * Once all shares are collected, the smart contract initiates the aggregation of encrypted updates. Because of homomorphic encryption, these encrypted shares can be summed directly on the blockchain without decryption.
5. **Master Node Retrieves Final Aggregated Result**
   * After aggregation, the master node retrieves the final result from the blockchain. Since the aggregated data is still encrypted, the master node can use a private key to decrypt it and obtain the final, combined model update.
   * The master node updates the global model with this aggregated result and can distribute it back to the nodes for the next training round.
6. **Blockchain as an Audit and Compliance Trail**
   * Every transaction (e.g., encrypted share submission, aggregation results) is recorded on the blockchain, creating a tamper-proof audit trail.
   * The blockchain ensures transparency and accountability, as each step is recorded and verifiable, and regulatory bodies can audit this ledger if required.

### 

### 

### **How Quantum Computing Could Enhance This Setup**

Quantum computing could be beneficial in several ways, particularly by speeding up encryption/decryption and improving the efficiency of MPC. Here’s where it could make a difference:

1. **Faster Homomorphic Encryption/Decryption**
   * Homomorphic encryption is computationally intensive. Quantum computing could accelerate this process by quickly factoring large numbers and handling complex computations on encrypted data, which would significantly speed up homomorphic encryption tasks.
2. **Quantum Key Distribution (QKD) for Enhanced Security**
   * Quantum key distribution provides an unbreakable method for secure key exchange using quantum mechanics. By incorporating QKD, nodes could securely exchange encryption keys with guaranteed confidentiality, even against potential quantum attacks.
3. **Quantum-Based MPC**
   * Quantum algorithms can optimize MPC protocols by performing operations on shared data more efficiently. Quantum MPC could enhance the security and speed of the sharing and aggregation process, especially for large-scale networks with thousands of nodes.
4. **Post-Quantum Cryptography**
   * Quantum computers pose a risk to classical encryption methods. Using post-quantum cryptography for all blockchain and encryption protocols can future-proof the system against potential quantum attacks, ensuring that the data remains secure even as quantum technology advances.

### **Summary of the Workflow**

1. **Nodes Train Locally**: Each node trains locally and prepares its updates using hybrid encryption.
2. **Apply Hybrid Encryption**:
   * **MPC**: Split data into shares and distribute them among nodes.
   * **Homomorphic Encryption**: Encrypt each share.
   * Record encrypted shares on the blockchain for immutability.
3. **Blockchain for Key/Share Management and Verification**: Nodes use blockchain for secure, decentralized key management.
4. **Secure Aggregation with Blockchain**:
   * Use smart contracts to verify and aggregate encrypted data directly.
   * Only aggregated data is accessible to the master node, preserving node privacy.
5. **Master Node Decrypts Aggregated Result**: The master node decrypts the aggregated result, updates the global model, and sends it back to nodes.
6. **Blockchain for Auditing**: Every step is recorded on the blockchain for transparency and compliance.
7. The qPoW system combines cryptographic hash functions with quantum circuits, creating a secure and verifiable system.

### 

### **Benefits and Security Guarantees**

* **Enhanced Security**: Hybrid encryption with MPC and blockchain ensures each update is protected and only aggregated data is accessible.
* **Transparency and Integrity**: Blockchain’s immutability and smart contracts create a transparent, auditable system.
* **Scalability**: This setup can handle a large number of nodes securely.
* **Future-Proofing with Quantum Computing**: Quantum computing can accelerate encryption and decryption while QKD and post-quantum cryptography prepare the system against future quantum threats.

Task:

Θεώρησε ότι έχεις ένα απλό pytorch table matrix (θα είναι τα weights του trained model) Wi σε κάθε υπολογιστή 'i' μέσα σε ένα δίκτυο Δ.

Κάθε υπολογιστής μέσα στο δίκτυο θα στέλνει μέσω encrypted pathways τον πίνακα Wi σε έναν κεντρικό υπολογιστή "Global Model". Το global model θα κάνει την εξής λειτουργία:

Θα δημιουργεί τον πίνακα Gi = ΣWi/N, για i από το 1 έως το Ν.

Στην συνέχεια αυτός ο πίνακας Gi θα μοιράζεται εκ νέου στους υπολογιστές μέσω του encrypted pathway.

Άρα στην αρχή της 2ης επανάληψης θα ισχύει ότι Α'i=Gi=ΣAi/N

Το Τάσκ καλό είναι να υλοποιηθεί τόσο εντός google colab notebook όσο και με simulation τοπικά στο pc.

Το simulation είναι το ζητούμενο.

\*\* Στο αρχικό implementation θα έχει σημασία μόνο το πρώτο feed forward και feed back καθώς δεν έχουμε ακόμα το update weight from model training.

## **Strengths of This Approach**

### **1. High Security**

* **Hybrid Encryption (MPC + Homomorphic):** Ensures that individual node updates remain private while still allowing secure aggregation.
* **Qhash Integration:** Guarantees integrity and immutability, making the system robust against tampering.
* **Quantum Key Distribution (QKD):** Prepares the system for a post-quantum world, ensuring encryption keys are safe against future quantum attacks.
* **Blockchain Immutability:** Provides a tamper-proof ledger, ensuring data authenticity and accountability.

### **2. Transparency and Auditability**

* Every step is recorded on the blockchain, making the system auditable and compliant with regulations (e.g., GDPR, HIPAA).

### **3. Scalability**

* The use of blockchain and distributed ledger technology ensures that the system can scale across a large number of nodes.
* With optimizations like **batch processing** and **parallel encryption**, it can handle 1000+ nodes efficiently.

### **4. Federated Learning Enhancements**

* By combining federated learning with these advanced security protocols, you effectively address:
  + **Data Privacy:** Local data never leaves the nodes.
  + **Update Privacy:** Individual updates remain secure even during aggregation.
  + **Data Ownership:** Nodes retain full control over their data.

## **Challenges and Considerations**

### **1. Computational Overhead**

* **Homomorphic Encryption:** While it allows operations on encrypted data, it is computationally expensive. Optimizations or lighter schemes (e.g., CKKS for approximate computations) may be needed.
* **Shamir's Secret Sharing (MPC):** Generating and distributing shares for large matrices (especially with 1000 nodes) can introduce latency.

### **2. Blockchain Latency and Gas Costs**

* Public blockchains (e.g., Ethereum) can be slow and expensive for frequent transactions.
* **Solution:**
  + Use Layer 2 solutions like **Polygon** for faster, cheaper blockchain interactions.
  + Consider a **private blockchain** for lower costs and faster processing.

### **3. Complexity**

* Integrating QKD, blockchain, and federated learning introduces significant complexity in implementation, debugging, and maintenance.
* **Solution:** Start with simpler federated learning setups and progressively integrate advanced features like blockchain and QKD.

### **4. Quantum Computing Hardware**

* While QKD is highly secure, it requires specialized hardware for real-world deployment.
* **Solution:** Simulate QKD for now and plan to integrate real QKD hardware as it becomes more accessible.

## **Improvements for Federated Learning**

### **1. Reduce Blockchain Dependency**

Instead of recording every share on the blockchain:

* **Batch Transactions:** Combine updates from multiple nodes into a single transaction.
* **Store Hashes Only:** Store Qhashes instead of the full encrypted shares to reduce storage requirements.

### **2. Optimize Encryption**

* Use **lighter encryption schemes** (e.g., CKKS or BFV) for approximate computations if exact results aren't critical.
* Parallelize encryption and decryption processes across multiple nodes or GPUs.

### **3. Use Adaptive Federated Learning**

* Allow nodes with high-quality data or faster processing capabilities to contribute more to the global model (weighted aggregation).

### **4. Improve Aggregation**

* Move aggregation logic directly onto the blockchain using **smart contracts** to reduce the reliance on the master node.

## **Comparison with Traditional Federated Learning**

| **Feature** | **Traditional FL** | **This Approach** |
| --- | --- | --- |
| **Security** | Relies on encryption for communication | MPC, Homomorphic Encryption, Blockchain, QKD |
| **Privacy** | Data privacy via local training | Data + Update privacy with encryption and MPC |
| **Transparency** | Limited | Fully auditable via blockchain |
| **Quantum Resistance** | Vulnerable to quantum attacks | Qhash, QKD, and Post-Quantum Cryptography |
| **Scalability** | Scales with optimizations | Scales with optimizations and blockchain layers |
| **Complexity** | Relatively simple | High complexity due to hybrid methods |

## **Conclusion**

### **Is This a Good Approach?**

* **Yes**, if:
  + **Security** and **future-proofing** are critical for your application.
  + You need a **transparent and auditable system**.
  + You are building a **research-oriented or enterprise-level system**.
* **No**, if:
  + Performance and simplicity are higher priorities.
  + Cost (e.g., blockchain gas fees) is a major concern.
  + You don’t need advanced quantum resistance yet.

### **Final Recommendation**

1. Start with a simpler federated learning system to establish the base.
2. Gradually integrate blockchain for auditing and Qhash for integrity.
3. Add homomorphic encryption and QKD as needed for enhanced security.

Let me know if you want to refine or proceed with a specific implementation!

## **Step 1: Local Model Training**

### **What Happens:**

1. Each of the 1000 nodes trains a machine learning model on its local dataset.
2. The training produces updates (weight matrices Wi), which represent the changes needed for the global model.

### **Why It’s Important:**

* **Privacy:** Data never leaves the local nodes, so sensitive information remains secure.
* **Decentralization:** Each node works independently, reflecting real-world data scenarios where data is distributed.

## **Step 2: Hybrid Encryption**

This step secures the weight matrices before they are shared with the global system.

### **How It Works:**

1. **Shamir's Secret Sharing (MPC):**
   * Each node splits its weight matrix into random pieces (called shares).
   * Only a subset of shares is needed to reconstruct the original data.
   * These shares are distributed among other nodes, ensuring that no single node holds the complete data.
2. **CKKS Homomorphic Encryption:**
   * Each share is encrypted using CKKS (a homomorphic encryption scheme).
   * CKKS allows mathematical operations (e.g., summation) directly on encrypted data without the need for decryption.

### **Why It’s Important:**

* **Privacy:** Even if shares are intercepted, they cannot reveal the original data.
* **Security:** Homomorphic encryption ensures that sensitive computations can be performed securely.

## **Step 3: Qhash Integration**

### **How It Works:**

* Each encrypted share is hashed using a quantum-resistant hash function (**Qhash**).
* The hash acts as a unique identifier for the data.
* Only the Qhash is stored on the blockchain, reducing the need to store full encrypted data.

### **Why It’s Important:**

* **Integrity:** Qhash ensures that no one can tamper with the encrypted data.
* **Efficiency:** Storing hashes instead of full data significantly reduces storage requirements on the blockchain.

## **Step 4: Blockchain Integration**

### **How It Works:**

1. **Batched Transactions:**
   * Nodes group their Qhashes into batches.
   * Each batch is submitted to the blockchain as a single transaction.
2. **Smart Contracts:**
   * A smart contract verifies that all Qhashes in a batch are valid.
   * It manages the storage and retrieval of Qhashes for later aggregation.
3. **Decentralized Aggregation:**
   * The blockchain smart contract aggregates encrypted shares directly on-chain using homomorphic properties.

### **Why It’s Important:**

* **Transparency:** The blockchain provides an immutable ledger of all transactions.
* **Decentralization:** On-chain aggregation reduces reliance on a single master node.
* **Efficiency:** Batching reduces transaction costs and blockchain overhead.

## **Step 5: Adaptive Federated Learning**

### **How It Works:**

* Each node is assigned a weight factor based on:
  + **Data quality:** Nodes with better datasets contribute more to the global model.
  + **Compute power:** Nodes with higher computational capabilities can handle larger contributions.
* The global model is updated using a **weighted average** of the nodes’ updates.

### **Why It’s Important:**

* **Fairness:** Nodes with better data or resources have a larger influence.
* **Efficiency:** Ensures that contributions are optimized for the best model performance.

## **Step 6: Aggregation and Global Model Update**

### **How It Works:**

1. The encrypted updates are aggregated (summed) on the blockchain.
2. The master node retrieves the aggregated encrypted result from the blockchain.
3. The master node decrypts the aggregated result using the private key.
4. The decrypted result is used to update the global model.

### **Why It’s Important:**

* **Privacy:** The updates are never exposed in plaintext until after aggregation.
* **Security:** The private key is only held by the master node, preventing unauthorized access.

## **Step 7: Redistribution to Nodes**

### **How It Works:**

* The updated global model is sent back to all nodes securely.
* Nodes use the new global model to restart the next round of local training.

### **Why It’s Important:**

* **Feedback Loop:** Each node gets the benefit of learning from the combined contributions of all nodes.
* **Continuous Improvement:** The global model improves iteratively over multiple rounds.

## **Key Improvements in This Solution**

### **1. Reduced Blockchain Dependency**

* Instead of storing all encrypted shares, only Qhashes are stored.
* This reduces storage requirements and speeds up blockchain interactions.

### **2. Optimized Encryption**

* CKKS encryption is used instead of heavier schemes like Paillier.
* Encryption and decryption processes are parallelized to handle large-scale computations.

### **3. Adaptive Learning**

* Nodes contribute more if they have higher-quality data or greater computational resources.
* This ensures fairness and improves model performance.

### **4. On-Chain Aggregation**

* Aggregation logic is moved to the blockchain using smart contracts.
* This reduces reliance on the master node and improves decentralization.

## **Why This Solution Works for Federated Learning**

1. **Scalability:**
   * Efficient batching and parallelized encryption make it suitable for 1000+ nodes.
2. **Privacy and Security:**
   * Data never leaves the nodes in plaintext.
   * Encryption, MPC, and blockchain ensure that updates remain private and tamper-proof.
3. **Future-Proofing:**
   * Qhash and CKKS encryption are resistant to quantum attacks.
   * This makes the system secure against emerging threats.
4. **Fairness and Performance:**
   * Adaptive learning ensures that nodes contribute proportionally to their capability.
   * Weighted updates improve the overall quality of the global model.

## **Conclusion**

This solution provides a highly secure, scalable, and efficient framework for federated learning. It combines the strengths of blockchain, Qhash, and advanced encryption to address modern challenges in data privacy and model performance.