

# Privacy-Preserving Graph-Based Machine Learning with Fully Homomorphic Encryption for Collaborative Anti-Money Laundering

Fabrianne Effendi<sup>1</sup>[0009-0002-8595-4323] and Anupam Chattopadhyay<sup>1</sup>[0000-0002-8818-6983]

Nanyang Technological University, 50 Nanyang Ave, 639798, Singapore  
fabr0001@e.ntu.edu.sg, anupam@ntu.edu.sg

**Abstract.** Combating money laundering has become increasingly complex with the rise of cybercrime and digitalization of financial transactions. Graph-based machine learning techniques have emerged as promising tools for Anti-Money Laundering (AML) detection, capturing intricate relationships within money laundering networks. However, the effectiveness of AML solutions is hindered by data silos within financial institutions, limiting collaboration and overall efficacy. This research presents a novel privacy-preserving approach for collaborative AML machine learning, facilitating secure data sharing across institutions and borders while preserving privacy and regulatory compliance. Leveraging Fully Homomorphic Encryption (FHE), computations are directly performed on encrypted data, ensuring the confidentiality of financial data. Notably, FHE over the Torus (TFHE) was integrated with graph-based machine learning using Zama Concrete ML. The research contributes two key privacy-preserving pipelines. First, the development of a privacy-preserving Graph Neural Network (GNN) pipeline was explored. Optimization techniques like quantization and pruning were used to render the GNN FHE-compatible. Second, a privacy-preserving graph-based XGBoost pipeline leveraging Graph Feature Preprocessor (GFP) was successfully developed. Experiments demonstrated strong predictive performance, with the XGBoost model consistently achieving over 99% accuracy, F1-score, precision, and recall on the balanced AML dataset in both unencrypted and FHE-encrypted inference settings. On the imbalanced dataset, the incorporation of graph-based features improved the F1-score by 8%. The research highlights the need to balance the trade-off between privacy and computational efficiency.

**Keywords:** Fully Homomorphic Encryption · Privacy-Preserving · Graph-Based Machine Learning · Anti-Money Laundering.

## 1 Introduction

### 1.1 Background and Motivation

**The Money Laundering Problem.** Money laundering is the process of concealing the origins of illegally obtained funds to make them appear legitimate. It

poses severe consequences, fostering corruption, organized crime, terrorism, and environmental offenses. Beyond discouraging foreign investments and distorting international capital flows, money laundering poses a substantial threat to the stability of the financial system and the broader economy. The United Nations estimates that 2-5% of global GDP is laundered annually, with only 1% of these funds seized [1]. For example, in 2023, Singapore uncovered a money laundering case involving over S\$3 billion (US\$2.21 billion), making it one of the largest cases discovered globally [2].

### **Graph-Based Machine Learning in Anti-Money Laundering (AML).**

Money laundering activities inherently involve network structures, where their web of transactions can be modelled by graph-like patterns such as fan-out, fan-in, gather-scatter and so on [3]. Machine learning have gained considerable interest in uncovering hidden patterns in transaction data [4]. Particularly, graph-based machine learning [5] including graph neural networks (GNNs) show great promise in AML detection (section 2.1). They can effectively capture complex relationships between entities such as individuals, businesses and accounts, identifying money-laundering patterns, learning anomalies, and unveiling hidden connections that are often missed by traditional detection methods.

**The Challenge of AML Silos.** Despite the advancements in AML solutions, AML transaction monitoring and detection are often conducted in silos within each financial institution [6], hindering the effectiveness of AML solutions. Adherence to data privacy regulations, such as the EU’s General Data Protection Regulation (GDPR) [7], poses legal challenges to financial institutions in maintaining the privacy of transaction data when data sharing. It is not possible for financial institutions to directly share meta-information about transaction accounts or owners without explicit consent. Consequently, this prevents the comprehensive modeling of transaction networks and limits the ability of financial institutions to track transactions and customer relationships spanning across multiple institutions and borders, which is often exploited by criminals orchestrating intricate schemes across multiple institutions.

**Collaborative AML.** To effectively combat money laundering, a collaborative approach is required. Emerging solutions involve leveraging a Trusted Third Party (TTP) to aggregate and analyze data, while preserving privacy. For example, the Monetary Authority of Singapore (MAS) developed COSMIC in Singapore [8], a platform for collaborative sharing of Money Laundering/Terrorism Financing (ML/TF) information among banks using rule-based transaction monitoring [9]. However, many institutions still lack legally-compliant infrastructure for secure data sharing. According to a FATF report, regulatory challenges and data privacy concerns were the two most frequently cited challenges in developing and implementing new AML technologies, with nearly 70% and 60% of respondents highlighting these issues respectively [10].

**Privacy-Preserving Technologies.** Privacy-preserving technologies, such as Homomorphic Encryption (HE), offer a promising solution. HE allows computations to be performed directly on encrypted data without decryption, ensuring data confidentiality. The increasing adoption and community support of HE, such as Fully Homomorphic Encryption over the Torus (TFHE) (section 3.2), motivates research in applying it to AML to enable secure collaborative efforts.

## 1.2 Objective and Contributions

This paper presents a novel privacy-preserving AML approach using TFHE to enable secure data sharing and collaboration between financial institutions while protecting data privacy. To the best of my knowledge, while there are recent research works that have conducted separate explorations of graph-based machine learning for AML (section 2.1), Gradient-Boosting Tree (GBT) for AML (section 2.2), privacy-preserving technologies (PPTs) for collaborative AML and privacy-preserving machine learning respectively (section 2.3), none have combined these areas of study to explore the use of GNN or GBT with FHE in AML, all the less so with the TFHE scheme. Hence, this paper makes the following contributions:

1. A proposed solution architecture for collaborative privacy-preserving machine learning for AML using FHE. This is detailed in section 3.5.
2. Exploration of the feasibility of developing a privacy-preserving GNN pipeline integrating TFHE using Concrete ML [11] for money laundering detection. This exploration is detailed in section 4.
3. Development of a privacy-preserving graph-based XGBoost pipeline leveraging the Graph Feature Preprocessor (GFP) [12] and integrating TFHE using Concrete ML [11] to detect money laundering. This is detailed in section 5.
4. A set of experiments with incrementally GFP-enriched graph features using XGBoost, comparing the model performance for both unencrypted and TFHE-encrypted inference, and against a basic XGBoost baseline. The results evaluated the trade-offs of building a privacy-preserving pipeline between privacy and model performance. This is detailed in section 6.

We have also released the project code repository on GitHub<sup>1</sup>.

## 2 Related Work

### 2.1 Graph-Based Machine Learning for AML

The application of Graph Neural Networks (GNNs) to the domain of Anti-Money Laundering (AML) has been an active area of research. Weber et al. conducted early experiments on synthetic AML datasets, finding that variants like Fast Graph Convolutional Networks (FastGCN) offered faster computation [13]. They later explored GNNs and traditional machine learning methods on the real-world

<sup>1</sup> <https://github.com/fabecode/GraphML-FHE>

Elliptic dataset, with GCN outperforming simpler models but being surpassed by random forest [14]. Recognizing the importance of temporal information, Alarab et al. [15] and Pareja et al. [16] developed novel GNN architectures that captured evolving financial transaction patterns over time on the Elliptic dataset. Cardoso et al. [17] introduced LaundroGraph, a self-supervised graph representation learning approach tailored for AML, showcasing the potential of graph-based techniques in extracting meaningful representations from financial interaction networks. Building on these foundational works, Johannessen et al. explored heterogeneous graph neural networks for real-world AML scenarios at DNB Bank [18], while Altman et al. [3] and Egressy et al. [19] made significant contributions in developing and evaluating GNN models on the comprehensive synthetic AMLworld dataset. This body of research consistently highlights the effectiveness of GNNs in capturing the complex, graph-structured nature of money laundering activities, paving the way for continued advancements in this field.

## 2.2 Gradient-Boosted Trees (GBT) for AML

The application of gradient-boosted tree (GBT) models, such as XGBoost and LightGBM, has also gained significant traction, with several studies including Tertychnyi et al. [20], Jullum et al. [21] and Vassallo et al. [22], demonstrating the effectiveness of these ensemble learning techniques in AML detection. More recently, researchers have explored integrating graph-based feature extraction techniques with GBT models to further enhance their performance in AML tasks. Eddin et al. [23] showcased the benefits of incorporating graph-based features, such as degree and GuiltyWalker, into LightGBM models. Building on this, Altman et al. [3] and Blanuša et al. [12] conducted comprehensive comparisons of GNN baselines and GBT models enhanced with their Graph Feature Preprocessor (GFP) on the AMLworld dataset, finding that the GBT-GFP combination, particularly XGBoost-GFP, outperformed the GNN approaches. These advancements highlight the potential of leveraging the synergies between graph-based representations and gradient-boosted tree models for effective AML detection.

## 2.3 Privacy-Preserving Technologies (PETs)

**PETs in Financial Crimes.** Privacy-Preserving Technologies (PETs) have emerged as a promising approach to protect sensitive information and enable secure data computation and analysis while preserving privacy in various domains, including finance applications [24]. The integration of PETs with AML efforts is a relatively new and active area of research.

One notable industry-led proof-of-concept is Project Aurora by the Bank for International Settlements [25], which explored the use of different PET combinations, along with machine learning and network analysis, to detect money laundering across siloed, national, and cross-border payment data. Their findings suggest that a centralized cross-border approach leveraging HE and Local Differential Privacy (LDP) demonstrates the best performance in AML. Meanwhile, Mastercard showcased the potential of FHE in facilitating secure cross-border

sharing of financial crime intelligence among Singapore, the United States, India, and the United Kingdom [26].

Recently, research-focused advancements have also emerged in this space. Zhang et al. [27] proposed a framework that utilizes hybrid Federated Learning (FL) to enable collaborative financial crime detection among multiple institutions, while Egmond et al. [28] developed a secure risk propagation algorithm using secure multi-party computation (MPC) and additive HE for confidential risk score updates across a collaborative inter-bank network.

**PETs for Privacy-Preserving Machine Learning.** Numerous studies have also explored the integration of FHE with machine learning models, primarily focusing on Convolutional Neural Networks (CNNs), including CryptoNets by Dowlin et al. [29], Homomorphic CNN (HCNN) by Badawi et al. [30], Low-Latency CryptoNets (LoLa) by Brutzkus et al. [31], and ResNet-20 with RNS-CKKS FHE and bootstrapping by Lee et al. [32]. However, these approaches often prove ineffective for Graph Neural Networks (GNNs) due to the differences in computational patterns between the two. While FHE-based CNN inference often focuses on optimizing 2D convolutions, GNNs like Graph Convolutional Networks (GCNs) introduce different computational patterns. Specifically, GCNs rely heavily on consecutive matrix multiplications for feature and node aggregation, creating a bottleneck that is not present in CNN layers, where multi-channel 2D convolutions dominate the computations [33].

Notably, in the area of privacy-preserving GNNs, Ran et al. [34] introduced CryptoGCN, a HE-based framework for Graph Convolutional Network (GCN) inference, leveraging the sparsity of matrix operations to minimize encrypted computational overhead. These advancements in PET-based approaches underscore the growing importance and potential of privacy-preserving techniques in the fight against financial crimes, setting the stage for the proposed privacy-preserving AML solution in this research.

### 3 Homomorphic Encryption (HE)

#### 3.1 HE Overview

Homomorphic Encryption (HE) is a cryptographic technique that allows computations to be directly performed on encrypted data without the need for decryption. Unlike traditional encryption, which necessitates the decryption of data before meaningful computation, HE empowers secure computation while maintaining the confidentiality of data throughout the computation process.

#### 3.2 Fully Homomorphic Encryption (FHE)

Fully Homomorphic Encryption (FHE) was imagined by Rivest et al. [35] in 1978, and the first scheme was developed by Craig Gentry [36] in 2009. FHE supports an unlimited number of computations on encrypted data without revealing

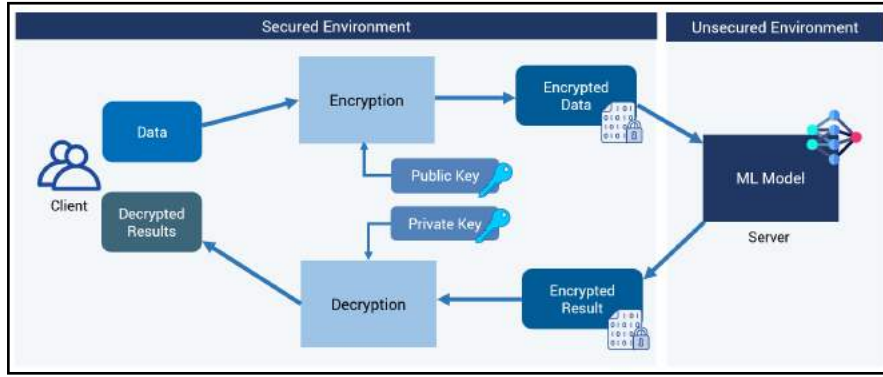
the underlying messages. Mathematically, if a user has an arbitrary function  $f$  and aims to derive  $f(m_1, \dots, m_n)$  for some inputs  $m_1, \dots, m_n$ , the plaintexts  $p_1, \dots, p_n$  are first encrypted using the public key  $pk$ , i.e.:

$$\text{Encrypt}(pk, p_1, \dots, p_n) = c_1, \dots, c_n \quad (1)$$

Computations are then performed directly on the encrypted ciphertexts  $c_1, \dots, c_n$ . Decryption of the output using secret key  $sk$  gives the result  $f(m_1, \dots, m_n)$ , i.e.:

$$\text{Decrypt}(sk, \text{Eval}(pk, f, c_1, \dots, c_n)) = f(m_1, \dots, m_n) \quad (2)$$

**Machine Learning using FHE.** In the context of machine learning, a party can encrypt input data using the public key, and the model will be able to process the data without seeing the original data. The final result is also encrypted, and the data can only be decrypted by the client who has the private key. This ensures confidentiality of the data from the machine learning model.



**Fig. 1.** Fully Homomorphic Encryption Process in Machine Learning

**Fully Homomorphic Encryption over the Torus (TFHE).** TFHE is a specialized FHE scheme designed by Chillotti et al. [37] for efficient computation of Boolean circuits on encrypted data. It enables very fast gate bootstrapping as well as circuit bootstrapping and operations over Boolean gates, reducing bootstrapping time to  $13ms$ .

**TFHE-Concrete.** TFHE-Concrete, an extended version of TFHE in Chillotti et al. [38], enhances the versatility and performance of TFHE in practical applications. It pushes the frontiers of bootstrapping by being the first to implement programmable bootstrapping (PBS), allowing for the simultaneous evaluation of arbitrary univariate function during bootstrapping. This is achieved by replacing the plaintext bits with a function of them in a lookup table. Presently, this is the most powerful technique for efficiently evaluating homomorphic non-linear functions, including activation functions in deep neural networks [39]. By combining the benefits of fast bootstrapping with programmable bootstrapping functionality, TFHE-Concrete enables secure and efficient computations on encrypted data, making it a valuable tool for privacy-preserving data analysis.

### 3.3 Zama’s Concrete Framework

Zama’s Concrete framework is an open-source tool that empowers developers to integrate HE into their applications without the need for in-depth cryptography knowledge [38]. TFHE-Concrete, as an integral part of the Concrete framework, provides comprehensive support across various categories, including leveled operations, bootstrapped operations, and PBS. It supports the approximate or exact evaluation of arbitrary functions, and supports both Boolean and integer operations, making it a versatile and integrated FHE solution.

**Table 1.** Comparison of TFHE-Concrete with Other FHE Schemes [40]

FHE Schemes	Operations			Non-linear		Data Types		
	Level	Bootstrapped	PBS	Exact	Approx	Bool	Int	Real
<b>BGV</b>	✓	×	×	×	✓	×	✓	×
<b>BFV</b>	✓	×	×	×	✓	×	✓	×
<b>CKKS</b>	✓	×	×	×	✓	×	×	✓
<b>TFHE-Lib</b>	×	✓	×	✓	×	✓	×	×
<b>TFHE-Concrete</b>	✓	✓	✓	✓	✓	✓	✓	×

**Concrete ML.** Concrete ML is a privacy-preserving machine learning Python toolkit built on top of the Concrete framework [11]. Building on Concrete’s TFHE and PBS implementation, it empowers data scientists to develop privacy-preserving models on encrypted data using the TFHE framework for secure machine learning inference.

The toolkit incorporates ready-to-use FHE-friendly models with an interface equivalent to scikit-learn. Additionally, it also provides support for custom models, including deep neural networks built with PyTorch or Keras/Tensorflow. For custom models, it is necessary to implement quantization before compiling to FHE, utilizing third-party libraries like Brevitas for PyTorch. The model is subsequently converted and imported into Concrete ML through the Open Neural Network Exchange (ONNX), an open-source standard facilitating interoperability across various deep learning frameworks and hardware platforms.

### 3.4 FHE Implementation using Concrete ML

To implement FHE in our pipelines, Concrete ML was leveraged as the FHE implementation library. Concrete ML was chosen for its suitability in realizing privacy-preserving FHE-based machine learning models and its user-friendly interface for data scientists to develop privacy-preserving models on encrypted data. Training was first done on unencrypted data, producing a model that was then converted to an FHE equivalent that can perform encrypted inference.

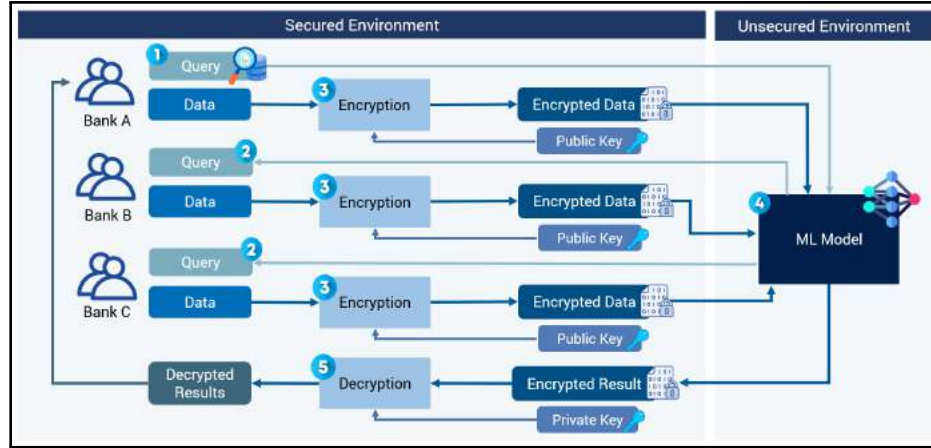
In the project, Concrete ML is used in 2 forms:

1. For XGBoost, Concrete ML’s built-in scikit-learn-like interface is leveraged to build an FHE-compatible XGBoost that performs encrypted inference.

2. For GNN, a custom model architecture is created using PyTorch Geometric, while leveraging Concrete ML for compilation, encryption, and inference.

### 3.5 Collaborative AML with FHE Solution Architecture

Below is a high-level overview of the project’s collaborative AML with FHE solution architecture.



**Fig. 2.** Collaborative FHE Solution Architecture

1. **Client Initialization:** A financial institution (FI) initiates the AML computation process with a query.
2. **Query Forwarding:** The centralised server forwards the query to participating FIs.
3. **Input Encryption:** Participating FIs, including the inquiry FI, encrypt relevant data using FHE with the inquiry FI’s public key. This key is securely shared once, allowing for multiple transactions to be encrypted without repeated exchanges.
4. **Secure Computation:** The consolidated encrypted data from multiple parties undergoes machine learning inference on the server, producing results in encrypted form while maintaining input confidentiality.
5. **Result Decryption:** The final encrypted results are decrypted by the inquiry FI using its private key, revealing only the output of the computation while keeping the individual inputs from participating FIs confidential.

The above process aligns with confidentiality regulations, such as those in Singapore, where the Banking Act prohibits the disclosure of Customer Information (CI), or the existence of a non-public relationship between the customer and the bank [26].



## 4 TFHE-Compatible GNN Pipeline



**Fig. 3.** Privacy-Preserving GNN Pipeline

The GNN pipeline focused on exploring the feasibility of making a Graph Neural Network (GNN) compatible with Fully Homomorphic Encryption (FHE), a challenge that has not been extensively addressed due to the inherent complexity of GNNs. This aimed to address the lack of prior work in integrating GNNs with FHE, which is crucial for enabling privacy-preserving collaborative machine learning in domains such as anti-money laundering.

The pipeline involved: selecting a GNN baseline model, quantizing and pruning it for FHE compatibility, training on Concrete ML, compiling the GNN to its FHE equivalent, and finally performing FHE inference.

### 4.1 Baseline GNN model

The Graph Isomorphism Network (GIN) was selected as the baseline GNN model. GIN is a message passing GNN that uses an iterative aggregation mechanism inspired by the Weisfeiler-Lehman graph isomorphism test [44]. The baseline GIN model is implemented using PyTorch Geometric and adapted from previous AML studies [3,19,45].

### 4.2 Quantization

Quantization makes models TFHE-compatible by transforming floating-points to integer representation. It is the process of constraining an input from a continuous or otherwise large set of values to a discrete set. While primarily used for enhancing the efficiency and compression of neural networks, quantization also helps address specific limitations associated with FHE [46]. Firstly, quantization effectively substitutes floating-point value multiplications with integer multiplications, a feasible operation in FHE. Moreover, quantization enables the reduction of values to small integers, offering a strategy to navigate the challenges posed by limited precision in programmable bootstrapping.

**Quantization Targets.** To refine the efficiency and security of Graph Neural Networks (GNNs), the following quantization targets strategies are employed to facilitate the quantization process. Each strategy focuses on specific aspects of the GNN architecture, encompassing both node and edge features, as well as the quantization of weights and activations within GNN layers.

1. **Node Feature Quantization:** Node feature quantization reduces the precision of input node features. To achieve this, the chosen methodology involves the application of quantization to node features, effectively transforming their representation into lower-precision numerical values. In practice, this quantization is implemented through the adoption of quantized linear layers, such as QuantLinear in Brevitas, strategically incorporated into both input and hidden layers.
2. **Edge Feature Quantization:** Quantizing edge features becomes imperative, particularly in scenarios involving edge convolutional layers. The methodology extends the quantization process to edge features, dependent on their relevance in the GNN architecture. Implementation-wise, the integration of quantized linear layers (QuantLinear) or alternative quantized operations is recommended to effectively process edge features.
3. **Weights and Activations Quantization:** In the realm of GNN layers, the overarching objective is to quantize both weights and activations. This is achieved through substituting the traditional linear layers with quantized linear layers (QuantLinear), and activation function with quantized activation functions (QuantReLU) within the GNN model.

Overall, the quantization strategies encompass a reduction in weight bit width for layer configurations, and a decrease in accumulator bit width for activation. These quantization techniques contribute significantly to the optimization and security enhancement of Graph Neural Networks.

**Quantization-Aware Training (QAT).** Quantization-Aware Training (QAT) was employed to ensure the GNN model’s compatibility with the reduced precision requirements of Fully Homomorphic Encryption (FHE). In QAT, the neural network undergoes training with an awareness of the quantization process, enabling the model to learn and adapt to the intricacies associated with lower bit-width weights and activations [47,48]. QAT was implemented using Brevitas for PyTorch. Each PyTorch layer was mapped to its quantized version in Brevitas, and the corresponding weight and bit width were configured accordingly.

### 4.3 Pruning

Pruning was employed to optimize the GNN model size and computational complexity, mitigating the risk of accumulator overflow during Fully Homomorphic Encryption (FHE) computations in Concrete ML [47]. Pruning involves setting certain weights in the neural network to zero, reducing the model’s size and computational demands.

Some key benefits of pruning in the context of FHE include: controlling the number of active neurons to reduce computational complexity, managing the accumulator bit width to ensure compatibility with FHE’s limited precision, and enhancing resource efficiency by reducing storage and computation requirements.

In the GNN implementation, edge pruning was performed on less informative edges to streamline graph connectivity while ensuring the integrity of the financial transactions graph and preserving critical components.

#### 4.4 Compilation of GNN model to FHE Equivalent

Final conversion of the GNN model is performed by Concrete ML, which uses the Concrete Compiler to translate the Multi-Level Intermediate Representation (MLIR) representation of the model into an FHE program, generating machine code that executes the model on encrypted data.

The conversion process starts with importing the ONNX model, followed by a sequence of transformations: initially into a NumpyModule, then into a QuantizedModule, and ultimately into an FHE circuit [49].

While efforts were made to develop a privacy-preserving GNN pipeline using Concrete ML, challenges in integrating PyTorch Geometric with the Concrete ML framework limited the successful completion of this component. The absence of support for the ScatterElements ONNX operator in Concrete ML posed a significant challenge during the final compilation of Graph Neural Networks (GNNs). While a custom implementation was developed to fill this gap, ongoing challenges remain in the conversion of NumpyModule to Quantized module. Nevertheless, the progress made in implementing techniques to render the GNN TFHE-compatible provides a good foundation for future research.

### 5 TFHE-Compatible XGBoost with GFP Pipeline



**Fig. 4.** Privacy-Preserving XGBoost Pipeline

The privacy-preserving graph-based XGBoost pipeline was successfully developed. This novel approach combines the predictive power of XGBoost with the security and confidentiality guarantees provided by TFHE, while leveraging Snap ML Graph Feature Preprocessor (GFP) to enrich the model with informative graph-based features.

The XGBoost pipeline encompasses the following steps: splitting the data, enriching graph features using GFP, performing hyperparameter tuning using Bayesian optimization, training the XGBoost model and performing FHE-encrypted inference using Concrete ML.

#### 5.1 Data Splitting

Data splitting was performed in a temporal manner. The transactions were ordered in ascending order of timestamps and split into train and test sets. Generally, transactions occurring before timestamp  $T_1$  are included in the training set,

while transactions occurring after timestamp  $T_1$  are included in the testing set. Transactions from the same day were placed in the same set too. This method was employed to prevent data leakage and ensured that the model generalizes well to unseen data by maintaining the temporal order of transactions and avoiding information leakage from future to past data. Additionally, grouping same-day transactions enhanced the integrity of the dataset and facilitated more effective model training. The distribution ensures that the training and testing sets are representative of the overall dataset while maintaining a balanced illicit ratio for effective model training and evaluation.

## 5.2 Snap ML Graph Feature Preprocessor (GFP) Setup

To enrich the model, the Snap ML Graph Feature Preprocessor (GFP) [50] was used to generate graph-based features from the dataset. Developed by Blanuša et al. [12], GFP facilitates efficient and real-time feature extraction from graph-structured data. It is compatible with scikit-learn, and simplifies the creation and maintenance of in-memory graphs while extracting relevant features. To search for graph patterns, the preprocessor analyzes each edge in the input edge list to identify patterns it participates in. For every edge and pattern type, the preprocessor computes a pattern histogram, which records the count of patterns of a given size. In the AML context, these additional features can augment the predictive capability of models like XGBoost, especially in detecting suspicious transactions patterns within financial transaction graphs.

The graph-based features were extracted from the AML datasets using a time window of 86400 seconds, or 1 day. Graph-based features include fan-in, fan-out, degree-in, degree-out, scatter-gather, simple cycle and temporal cycle patterns. For the vertex-statistics-based features, the "Amount" field of the basic transaction features were used to generate the vertex-statistics-based features. Vertex-statistics-based features selected were sum, variance and skewness.

An incremental approach was adopted to assess the impact of adding various graph-based features on the performance of the XGBoost model. These features were grouped into distinct categories: (i) basic features, (ii) fan-in/fan-out features, (iii) multi-hop pattern features, encompassing scatter-gather, simple cycle, and temporal cycle patterns, (iv) vertex-statistic-based features.

## 5.3 Hyperparameter Tuning via Bayesian Optimization

Bayesian optimization was employed to optimize the hyperparameters of the XGBoost model. This method systematically explores the hyperparameter space by iteratively assessing the model's performance with various parameter combinations, guided by a probabilistic model of the objective function. By selecting hyperparameters for evaluation based on the model's predictions, Bayesian optimization efficiently navigates the search space and identifies the hyperparameters values that maximize the model's performance. This approach enables the identification of optimal hyperparameters with fewer evaluations compared to exhaustive grid search or random search methods.

#### 5.4 Training and Inference using Concrete ML

After Bayesian optimization, the optimized XGBoost model undergoes the Concrete ML pipeline for training and FHE-encrypted inference. This process involves several key steps. First, the model is trained on plaintext, non-encrypted training data. Next, the floating-point values in the model are transformed into integers through a quantization step. The quantized model is then executed in a simulation environment to assess its accuracy under FHE and identify any necessary modifications to ensure full compatibility. Following this, the quantized model is compiled into an equivalent FHE circuit, which can be executed on encrypted data. Finally, once the appropriate cryptographic keys are generated, the quantized XGBoost model is executed on the encrypted data, enabling privacy-preserving inference using the Concrete ML framework.

## 6 Experiments and Results

### 6.1 Dataset

**Dataset Selection.** To train the machine learning models, a synthetic AML dataset was used to simulate money laundering scenarios, as strict privacy regulations surrounding banking data make real-world financial crime data not easily accessible. After evaluating several publicly available datasets [41,42,3], the AMLworld HI-Small dataset [3,43] was selected for its comprehensive representation of real-world money laundering activities.

The AMLworld HI-Small dataset models diverse patterns, including placement, layering, and integration, across multiple banks and currencies. Data exploration revealed 370 groups of money laundering patterns, comprising various topologies such as fan-in, fan-out, gather-scatter, cycles, and random patterns.

**Dataset Modification.** Due to computational constraints posed by the slower operations inherent in Fully Homomorphic Encryption (FHE), the original AMLworld HI-Small dataset was downsized. Two modified datasets were created using undersampling and random sampling techniques respectively:

1. Modified AML Dataset 1: A balanced dataset with 15,230 accounts, 10,354 transactions, and a 50% illicit ratio.
2. Modified AML Dataset 2: An imbalanced dataset with 9,070 accounts, 5,491 transactions, and a 5.72% illicit ratio.

These modified datasets provide a robust foundation for developing and evaluating machine learning pipelines to combat financial crimes, while accounting for the computational limitations of FHE.

## 6.2 Environment Setup

The experiments were conducted on a Linux system with an Intel Xeon CPU E5-1630 v4 @ 3.70GHz with 8 CPU cores, using Concrete ML version 1.4.1 to implement TFHE. As Concrete ML currently does not offer GPU support, CPU resources were utilized for the experiments.

## 6.3 Experimental Overview

Experiments were conducted on the privacy-preserving XGBoost pipeline using the 2 modified AML datasets, with inference conducted on both unencrypted (clear) and FHE-encrypted data. XGBoost served as the baseline model, and graph features were incrementally introduced in 3 main categories: (i) single-hop patterns, (ii) multi-hop patterns and (iii) vertex statistics patterns. These graph features were added onto the base dataset features as inputs for the training and testing pipelines.

**Table 2.** Pattern categories of Graph-Based Features

Pattern Category	Graph-Based Features
Single-Hop	Fan-in, fan-out, degree-in, degree-out
Multi-Hop	Scatter-gather, simple cycle, temporal cycle patterns
Vertex Statistics	Sum, variance, skewness of transaction amount

In the experiments, the Bayesian hyperparameter tuning process iteratively samples various hyperparameter combinations across 50 iterations. A 3-fold cross-validation technique is employed in each iteration, where the dataset is divided into 3 equal folds for training and evaluation, with average performance guiding subsequent iterations of the optimization process. Once optimal hyperparameters are determined, the resulting model is integrated into the graph-based gradient boosting pipeline for FHE-based inference.

## 6.4 Results and Discussion on Balanced AML Dataset 1

The following analysis will examine the performance metrics from Table 3 to assess the effectiveness of the Bayesian optimized XGBoost model in identifying illicit activities in the balanced AML dataset. Additionally, the discussion will explore the inference time overhead linked with FHE as depicted in Table 4.

**High Performance with Basic Features.** The XGBoost model demonstrates impressive performance even with just the basic features, achieving over 99% accuracy, F1 score, precision, and recall. This suggests that the model has effectively learned from the dataset and can make accurate predictions without the need for additional graph-based features.

**Little Effect of Graph Features.** Despite the presence of various graph patterns in the dataset, the inclusion of graph-based features did not significantly improve the model’s performance. This indicates that on a balanced dataset,

**Table 3.** Performance metrics of Bayesian optimized XGBoost on Dataset 1. It highlights the effect of FHE encryption and adding graph-based features on XGBoost performance. Clear denotes inference on unencrypted data, while FHE denotes inference on FHE-encrypted data.

Input Features	Accuracy		F1		Precision		Recall	
	Clear	FHE	Clear	FHE	Clear	FHE	Clear	FHE
Basic features	0.9972	0.9972	0.9978	0.9978	0.9981	0.9981	0.9975	0.9975
+ Single-hop	0.9972	0.9972	0.9978	0.9978	0.9981	0.9981	0.9975	0.9975
+ Multi-hop	0.9972	0.9972	0.9978	0.9978	0.9981	0.9981	0.9975	0.9975
+ Vertex statistics	0.9964	0.9964	0.9972	0.9972	0.9969	0.9969	0.9975	0.9975

**Table 4.** Inference time metrics of Bayesian optimized XGBoost on Dataset 1. It highlights the effect of FHE encryption and adding graph-based features on inference time. Time Ratio compares total inference time on FHE-encrypted data to unencrypted.

Input Features	Inference Time (s)				Time Ratio (FHE / Clear)
	Average (Batch)		Total		
	Clear	FHE	Clear	FHE	
Basic features	0.008414	1009.0963	0.1683	20181.9266	119926.12x
+ Single-hop	0.005632	806.4183	0.1183	16128.3664	136363.27x
+ Multi-hop	0.007108	818.8802	0.1422	16377.6037	115200.84x
+ Vertex statistics	0.005781	959.1578	0.1156	19183.1552	165917.40x

XGBoost may already capture the relevant features adequately without the need for additional complexity introduced by graph features.

**FHE Encryption Overhead.** While the model’s performance remained strong, the inference time for FHE-encrypted data was significantly higher than unencrypted data (Table 4), exceeding 100,000 times the unencrypted inference time. This highlights the need to explore strategies that can reduce the FHE computational overhead without compromising model performance.

Future work could focus on balancing model metrics and FHE inference time, potentially by adjusting parameters or feature sets to maintain high accuracy, precision, and recall while minimizing the encryption overhead.

## 6.5 Results and Discussion on Imbalanced AML Dataset 2

The following analysis evaluates the performance metrics (Table 5) and inference time (Table 6) for the imbalanced AML dataset. Here, the minority F1-score is a better measure of performance than accuracy, as the latter can be misleading by favoring the majority class. In contrast, the F1-score considers both precision and recall, offering a balanced assessment and highlighting the detection of illicit money laundering transactions.

**Lower Performance on Imbalanced Dataset.** For the imbalanced dataset, the model faced challenges in accurately identifying illicit activities, evident from the drop in F1-score and recall compared to the balanced dataset. The imbalanced nature negatively impacted the F1-score, a key metric for such scenarios.

**Table 5.** Performance metrics of Bayesian optimized XGBoost on Dataset 2. It highlights the effect of FHE encryption and adding graph-based features on XGBoost performance. Clear denotes inference on unencrypted data, while FHE denotes inference on FHE-encrypted data.

Input Features	Accuracy		F1		Precision		Recall	
	Clear	FHE	Clear	FHE	Clear	FHE	Clear	FHE
Basic features	0.9016	0.9016	0.3056	0.3056	0.7857	0.7857	0.1897	0.1897
+ Single-hop	0.9094	0.9094	0.3867	0.3867	0.8529	0.8529	0.2500	0.2500
+ Multi-hop	0.9075	0.9075	0.3733	0.3733	0.8235	0.8235	0.2414	0.2414
+ Vertex statistics	0.9035	0.9035	0.3467	0.3467	0.7647	0.7647	0.2241	0.2241

**Improved Performance with Graph-Based Features.** Despite the challenges posed by the imbalanced dataset, incorporating graph-based features improved the model’s predictive capabilities. The addition of single-hop features led to the highest accuracy, F1-score, precision, and recall among all feature sets, outperforming the basic features. However, adding more advanced multi-hop and vertex statistics features resulted in a slight performance decrease compared to the single-hop features, though they still outperformed the basic features. Further investigation is required to understand the reasons behind this performance change when incorporating the additional graph-based feature categories.

**Consistent Performance in Encrypted Inference.** Moreover, the consistency in performance between unencrypted and FHE-encrypted inference highlights the robustness of Concrete-TFHE in maintaining model accuracy, F1-score, precision and recall while preserving privacy. This underscores the potential of Concrete ML in preserving privacy without sacrificing predictive performance, a crucial aspect in sensitive domains like financial transactions.

**Table 6.** Inference time metrics of Bayesian optimized XGBoost on Dataset 2. It highlights the effect of FHE encryption and adding graph-based features on inference time. Time Ratio compares total inference time on FHE-encrypted data to unencrypted.

Input Features	Inference Time (s)				Time Ratio (FHE / Clear)
	Average (Batch)		Total		
	Clear	FHE	Clear	FHE	
Basic features	0.003041	270.1013	0.02433	2160.8107	88822.54x
+ Single-hop	0.003055	64.9026	0.02444	519.2208	21246.95x
+ Multi-hop	0.002270	34.1322	0.01816	273.0572	15038.47x
+ Vertex statistics	0.003607	121.0173	0.02886	968.1386	33546.76x

**Effect of Feature Complexity on Inference Time.** Interestingly, as graph-based feature complexity increased, inference time decreased for both clear and FHE-encrypted data, suggesting that the overhead of advanced features may be offset by their ability to streamline the inference process, resulting in more efficient model predictions. Further analysis is needed to validate this observation. Other potential factors contributing to this trend could include the optimization



of feature representations, the inherent parallelism in graph-based computations, or the effectiveness of the Bayesian optimization process.

Overall, the integration of graph-based features showed promise in enhancing the model’s ability to detect illicit activities. Future research could focus on optimizing model performance for imbalanced scenarios.

## 7 Conclusion

This paper proposed a novel privacy-preserving approach for collaborative AML detection using FHE. Two key pipelines were developed:

1. A privacy-preserving GNN pipeline that explored the integration of GIN with TFHE, where optimization techniques like quantization and pruning were used in attempts to render the GNN FHE-compatible.
2. A privacy-preserving graph-based XGBoost pipeline that leveraged the Graph Feature Preprocessor (GFP) to enhance the model’s predictive performance on the AML datasets. Experiments demonstrated the XGBoost model’s ability to achieve over 99% accuracy, F1-score, precision, and recall on the balanced dataset, with the incorporation of graph-based features improving the F1-score by 8% on the imbalanced dataset.

A key strength of the proposed approach was the consistent performance of the XGBoost model in both unencrypted and FHE-encrypted inference settings, highlighting the robustness of the Concrete-TFHE framework in preserving privacy without compromising predictive capabilities. However, it also underscored the need to balance the trade-off between privacy preservation and computational efficiency, as FHE-encrypted inference incurred significant overhead.

## 8 Future Work

This work lays the foundation for innovative privacy-preserving approaches to AML detection. Future research could focus on:

1. Improving FHE compatibility and performance of models.
2. Investigating alternative privacy-preserving methods, such as differential privacy or multi-party computation, to enhance privacy-preserving capabilities.
3. Assessing scalability of solutions to accommodate larger datasets.

By addressing these future directions, the privacy-preserving machine learning approaches developed in this paper can be further advanced, contributing to the ongoing efforts to safeguard financial systems against illicit activities while preserving data privacy. Additionally, the insights and techniques explored in this paper may have the potential for broader applications in other domains that require privacy-preserving collaborative machine learning.

**Acknowledgments.** The authors would like to thank Alka Luqman for her contributions during the debugging of the GNN pipeline.

**Disclosure of Interests.** The authors have no competing interests to declare that are relevant to the content of this article.

## References

1. United Nations Office on Drugs and Crime. Money Laundering. <https://www.unodc.org/unodc/en/money-laundering/overview.html>. 2023.
2. Megan Cheah. S\$3 billion anti-money laundering bust wraps up with 10th and final individual jailed. *The Business Times*, Jun. 2024.
3. Erik Altman, Jovan Blanuša, Luc von Niederhäusern, Béni Egressy, Andreea Anghel, Kubilay Atasü. Realistic Synthetic Financial Transactions for Anti-Money Laundering Models. 2023. arXiv: 2306.16424 [cs.AI].
4. McKinsey & Company. The fight against money laundering: Machine learning is a game changer. Oct. 2022.
5. E. Kurshan and H. Shen. Graph Computing for Financial Crime and Fraud Detection: Trends, Challenges and Outlook. 2021. arXiv: 2103.03227 [cs.CR].
6. PwC. Breaking down the silos to combat financial crimes. [www.pwc.com/ca/en/services/deals/breaking-down-the-silos-to-financial-crimes.html](http://www.pwc.com/ca/en/services/deals/breaking-down-the-silos-to-financial-crimes.html).
7. Sanne Wass. Banks need steer on data protection vs money-laundering rules: industry bodies. en-us, In: S&P Global Market Intelligence. Feb. 2020.
8. Monetary Authority of Singapore. COSMIC. en, Dec. 2023. <https://www.mas.gov.sg/regulation/anti-money-laundering/cosmic>.
9. Monetary Authority of Singapore. Consultation Paper on the Regulations Relating to FI-FI Information Sharing for AML/CFT. Dec. 2023.
10. Financial Action Task Force (FATF). Opportunities and Challenges of New Technologies for AML/CFT. FATF, Paris, France, 2021.
11. Zama. What is Concrete ML? | Concrete ML Documentation. en, Feb. 2024. <https://docs.zama.ai/concrete-ml/>.
12. Jovan Blanuša, Maximo Cravero Baraja, Andreea Anghel, Luc von Niederhäusern, Erik Altman, Haris Pozidis, Kubilay Atasü. Graph Feature Preprocessor: Real-time Subgraph-based Feature Extraction for Financial Crime Detection. 2024. arXiv: 2402.08593 [cs.LG].
13. Mark Weber, Jie Chen, Toyotaro Suzumura, Aldo Pareja, Tengfei Ma, Hiroki Kanezashi, Tim Kaler, Charles E. Leiserson, Tao B. Schardl. Scalable Graph Learning for Anti-Money Laundering: A First Look. 2018. arXiv: 1812.00076 [cs.SI].
14. Mark Weber, Giacomo Domeniconi, Jie Chen, Daniel Karl I. Weidele, Claudio Bellei, Tom Robinson, Charles E. Leiserson. Anti-Money Laundering in Bitcoin: Experimenting with Graph Convolutional Networks for Financial Forensics. 2019. arXiv: 1908.02591 [cs.SI].
15. Ismail Alarab and Simant Prakoornwit. “Graph-Based LSTM for Anti-money Laundering: Experimenting Temporal Graph Convolutional Network with Bitcoin Data”. *Neural Processing Letters* 55.1 (Feb. 2023), pp. 689–707.
16. Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao B. Schardl, Charles E. Leiserson. EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs. 2019. arXiv: 1902.10191 [cs.LG].
17. Mário Cardoso, Pedro Saleiro, and Pedro Bizarro. "LaundroGraph: Self-Supervised Graph Representation Learning for Anti-Money Laundering." *ICAIF '22: Proceedings of the Third ACM International Conference on AI in Finance*. 2022. arXiv: 2210.14360 [cs.LG].

18. Fredrik Johannessen and Martin Jullum. Finding Money Launderers Using Heterogeneous Graph Neural Networks. 2023. arXiv: 2307.13499 [cs.LG].
19. Béni Egressy, Luc von Niederhäusern, Jovan Blanusa, Erik Altman, Roger Wattenhofer, and Kubilay Atasü. Provably Powerful Graph Neural Networks for Directed Multigraphs. 2024. arXiv: 2306.11586 [cs.LG].
20. Pavlo Tertychnyi, Ivan Slobozhan, Madis Ollikainen, Marlon Dumas. “Scalable and Imbalance-Resistant Machine Learning Models for Anti-money Laundering: A Two-Layered Approach”. Nov. 2020, pp. 43–58. ISBN: 978-3-030-64465-9.
21. Martin Jullum, Anders Løland, Ragnar Bang Huseby, Geir Ånonsen, Johannes Lorentzen. “Detecting money laundering transactions with machine learning”. *Journal of Money Laundering Control* (Jan. 2020). DOI: 10.1108/JMLC-07-2019-0055.
22. Dylan Vassallo, Vincent Vella, and Joshua Ellul. “Application of Gradient Boosting Algorithms for Anti-money Laundering in Cryptocurrencies”. *SN Computer Science* 2 (May 2021). DOI: 10.1007/s42979-021-00558-z.
23. Ahmad Naser Eddin, Jacopo Bono, David Aparício, David Polido, João Tiago Ascensão, Pedro Bizarro, Pedro Ribeiro. Anti-Money Laundering Alert Optimization Using Machine Learning with Graphs. 2022. arXiv: 2112.07508 [cs.LG].
24. Carsten Baum, James Hsin-yu Chiang, Bernardo David, Tore Kasper Frederiksen. SoK: Privacy-Enhancing Technologies in Finance. Cryptology ePrint Archive, Paper 2023/122. 2023. <https://eprint.iacr.org/2023/122>.
25. Bank for International Settlements (BIS) Innovation Hub. “Project Aurora: the power of data, technology and collaboration to combat money laundering across institutions and borders”. en, Mar 2024.
26. Infocomm Media Development Authority. Preventing Financial Fraud Across Different Jurisdictions with Secure Data Collaborations - IMDA PET Sandbox - Mastercard Case Study. Nov. 2023.
27. Haobo Zhang, Junyuan Hong, Fan Dong, Steve Drew, Liangjie Xue, Jiayu Zhou. A Privacy-Preserving Hybrid Federated Learning Framework for Financial Crime Detection. 2023. arXiv: 2302.03654 [cs.LG].
28. Marie Beth van Egmond, Vincent Dunning, Stefan van den Berg, Thomas Rooijackers, Alex Sangers, Ton Poppe, Jan Veldsink. Privacy-preserving Anti-Money Laundering using Secure Multi-Party Computation. Cryptology ePrint Archive, Paper 2024/065. 2024.
29. Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. “CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy”. In *Proceedings of The 33rd International Conference on Machine Learning*, ed. by Maria Florina Balcan and Kilian Q. Weinberger, vol. 48, pp. 201–210. New York, USA: PMLR, 20–22 Jun 2016.
30. Ahmad Al Badawi, Jin Chao, Jie Lin, Chan Fook Mun, Jun Jie Sim, Benjamin Hong Meng Tan, Xiao Nan, Khin Mi Mi Aung, Vijay Ramaseshan Chandrasekhar. Towards the AlexNet Moment for Homomorphic Encryption: HCNN, the First Homomorphic CNN on Encrypted Data with GPUs. Cryptology ePrint Archive, Paper 2018/1056. 2018. DOI: 10.1109/TETC.2020.3014636.
31. Alon Brutzkus, Oren Elisha, and Ran Gilad-Bachrach. Low Latency Privacy Preserving Inference. 2019. arXiv: 1812.10659 [cs.LG].
32. Joon-Woo Lee, HyungChul Kang, Yongwoo Lee, Woosuk Choi, Jieun Eom, Maxim Deryabin, Eunsang Lee, Junhyun Lee, Donghoon Yoo, Young-Sik Kim, Jong-Seon No. Privacy-Preserving Machine Learning with Fully Homomorphic Encryption for Deep Neural Network. 2021. arXiv: 2106.07229 [cs.LG].

33. Ran Ran, Nuo Xu, Tao Liu, Wei Wang, Gang Quan, and Wujie Wen. Penguin: Parallel-Packed Homomorphic Encryption for Fast Graph Convolutional Network Inference. 2023. In *Proceedings of the Thirty-seventh Conference on Neural Information Processing Systems*.
34. Ran Ran, Nuo Xu, Wei Wang, Gang Quan, Jieming Yin, Wujie Wen. CryptoGCN: Fast and Scalable Homomorphically Encrypted Graph Convolutional Network Inference. 2022. arXiv: 2209.11904 [cs.CR].
35. Ronald L. Rivest, Len Adleman, Michael L. Dertouzos, and others. “On Data Banks and Privacy Homomorphisms”. *Foundations of Secure Computation* 4.11 (1978), pp. 169–180.
36. Craig Gentry. “A fully homomorphic encryption scheme”. PhD thesis. Stanford, CA, USA, 2009. ISBN: 9781109444506. Type: AAI3382729.
37. Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. TFHE: Fast Fully Homomorphic Encryption over the Torus. Cryptology ePrint Archive, Paper 2018/421. 2018. <https://eprint.iacr.org/2018/421>.
38. Ilaria Chillotti, Marc Joye, Damien Ligier, Jean-Baptiste Orfila, Samuel Tap. “CONCRETE: Concrete Operates oN Ciphertexts Rapidly by Extending TfhE”. In *8th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, p. 7. ACM. Virtual Event: ACM, New York, NY, USA, Dec. 2020. DOI: 10.25835/0072999.
39. Marc Joye. Homomorphic Encryption 101. en, Dec. 2021. <https://www.zama.ai/post/homomorphic-encryption-101>.
40. Zama. Introducing the Concrete Framework. en, July 2022. <https://www.zama.ai/post/introducing-the-concrete-framework>.
41. Maryam Mahootiha. Money Laundering Data. 2020. <https://www.kaggle.com/datasets/maryam1212/money-laundering-data>.
42. Toyotaro Suzumura and Hiroki Kanezashi. Anti-Money Laundering Datasets. 2021. <https://github.com/IBM/AMLSim>.
43. IBM Research. IBM Transactions for Anti Money Laundering (AML). 2023. <https://www.kaggle.com/datasets/ealtman2019/ibm-transactions-for-anti-money-laundering-aml>.
44. Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? 2019. arXiv: 1810.00826 [cs.LG].
45. Erik Altman, Jovan Blanuša, Luc von Niederhäusern, Béni Egressy, Andreea Anghel, Kubilay Atasü. Multi-GNN architectures for Anti-Money Laundering. 2023. <https://github.com/IBM/Multi-GNN>.
46. Zama. Quantization of Neural Networks for Fully Homomorphic Encryption. en, Jan. 2022. <https://www.zama.ai/post/quantization-of-neural-networks-for-fully-homomorphic-encryption>.
47. Andrei Stoian, Jordan Frery, Roman Bredehoft, Luis Montero, Celia Kherfallah, and Benoit Chevallier-Mames. Deep Neural Networks for Encrypted Inference with TFHE. Cryptology ePrint Archive, Paper 2023/257. 2023. <https://eprint.iacr.org/2023/257>.
48. Shyam A. Tailor, Javier Fernandez-Marques, and Nicholas D. Lane. Degree-Quant: Quantization-Aware Training for Graph Neural Networks. 2021. arXiv: 2008.05000 [cs.LG].
49. Zama. Compilation | Concrete ML Documentation. en, Feb. 2024. <https://docs.zama.ai/concrete-ml/advanced-topics/compilation>.
50. IBM Research. Graph Feature Preprocessor — Snap ML 1.14.2 documentation. 2023. [https://snapml.readthedocs.io/en/latest/graph\\_preprocessor.html](https://snapml.readthedocs.io/en/latest/graph_preprocessor.html).

## A Key Considerations in Collaborative FHE Architecture

This appendix outlines some of the key considerations in the design of the collaborative FHE architecture.

### A.1 Advantages of Collaborative FHE over MPC

Below details the key advantages of our proposed collaborative FHE architecture in comparison to traditional Multi-Party Computation (MPC) methods in the context of AML applications.

In MPC, data fragmentation is an important requirement where multiple parties must divide and share their data to perform computations on distributed fragments. This process necessitates continuous coordination and communication among all participating FIs, resulting in significant overhead and complexity. Each party must remain online and actively participate in every computation step, making the process highly interactive and less scalable.

In contrast, our collaborative FHE architecture does not rely on data fragmentation and allows for non-interactive processing of encrypted data. Once an FI encrypts its data using a public key, computations can proceed independently without further interaction. This non-interactive approach not only simplifies data handling but also enhances efficiency by allowing FIs to contribute data without needing to be online throughout the entire computation process.

Moreover, our FHE architecture enables multiple FIs to encrypt and submit their data for secure computation while requiring only the inquiry FI to decrypt the final result. This significantly reduces communication overhead, resulting in improved scalability compared to MPC, which incurs considerable overhead due to constant data exchanges among participants.

### A.2 Public Key Sharing

Our FHE architecture effectively manages public key sharing and minimizes associated network overhead.

- **Initial Key Sharing:** The inquiry FI’s public key can be securely distributed at the start of a session or when a new FI initiates an inquiry. Once the key is shared, multiple transactions can be encrypted under the same key, reducing the need for repeated re-encryption for individual inquiries. This streamlines the processing of multiple data transactions within a single inquiry session.
- **Managing Changes in Inquiry FI:** If a new inquiry FI is introduced, its public key is shared efficiently with participating FIs, requiring minimal overhead as the key exchange only involves secure transmission of a small piece of data.
- **Session-Based Key Exchange:** The architecture supports session-based key exchanges, enabling each inquiry FI to generate a new session key or update its public key for each inquiry. This approach ensures that only new transactions require encryption with the new key, while existing encrypted data can continue to be processed without interruption.

## B Detailed characteristics of modified AML datasets

This appendix provides additional details on the modifications made to the original AMLworld HI-Small dataset.

### B.1 Modified AML Dataset 1

Undersampling was used to create a balanced dataset of 10,354 transactions and 15,230 accounts, preserving rows associated with the 370 money laundering pattern groups in the original dataset (Table 7). This reduced the dataset size for FHE computations while balancing the 50

**Table 7.** Money Laundering Pattern Groups in AMLworld HI-Small dataset

Pattern	Count
Fan-In	61
Fan-Out	80
Gather-Scatter	77
Cycle	82
Random	70
Total Patterns	370

**Table 8.** Characteristics of Modified AML Dataset 1

Characteristics	Value
Number of accounts	15,230
Number of transactions	10,354
Illicit money laundering ratio	50%

### B.2 Modified AML Dataset 2

A smaller, highly imbalanced dataset was created by randomly sampling 40 out of the 370 pattern groups to represent illicit transactions (Table 9). This resulted in 5,491 transactions across 9,070 accounts, with a 5.72% illicit ratio (Table 10).

**Table 9.** Money Laundering Pattern Groups in Modified AML Dataset 2

Pattern	Count
Fan-In	8
Fan-Out	7
Gather-Scatter	9
Cycle	8
Random	8
Total Patterns	40

**Table 10.** Characteristics of Modified AML Dataset 2

Characteristics	Value
Number of accounts	9070
Number of transactions	5491
Illicit money laundering ratio	5.72%

## C Train-Test Split Distribution

The datasets were split temporally, with the training set comprising 75-81% of the data (Tables 11 and 12).

**Table 11.** Train-Test Split Distribution for Modified AML Dataset 1

	Percentage of Dataset	Illicit Ratio
Train Sample	75.85%	45.47%
Test Sample	24.15%	64.16%

**Table 12.** Train-Test Split Distribution for Modified AML Dataset 2

	Percentage of Dataset	Illicit Ratio
Train Sample	81.50%	4.42%
Test Sample	18.50%	11.42%

## D Definition of Graph-Based Features

Below are an elaboration of the graph-based features.

- **Fan-in/fan-out** reveals the inbound and outbound connectivity of vertices, aiding in understanding recipient and initiator vertices within the graph.
- **Degree-in/degree-out** quantifies the total inbound and outbound edges for each vertex, providing a holistic view of vertex connectivity.
- **Scatter-gather patterns** detects dispersion and aggregation behaviors across the graph, offering insights into information propagation and aggregation.
- **Simple cycles** identifies closed paths within the graph, highlighting recurring patterns or loops indicative of repetitive processes.
- **Temporal cycles** uncovers cyclic patterns with temporal dimensions, offering insights into recurring activities over time.

Below are an elaboration of the vertex-statistics-based features.

- **Sum** provides insight into the overall volume of financial activity. High sums may indicate potentially suspicious behavior, such as large-scale transactions or money laundering schemes.

- **Variance** reflects the dispersion or spread of transaction amounts around the mean. High variance may suggest irregular or unpredictable patterns in financial transactions, indicative of fraudulent activities.
- **Skewness** measures the asymmetry of the distribution of transaction amounts. Positive skewness suggests a longer tail towards higher values, while negative skewness suggests a longer tail towards lower values. Extreme positive skewness may indicate unusual transaction patterns that require further investigation.

## E Hyperparameter Tuning Search Space

The Bayesian optimization search space for XGBoost hyperparameters is detailed in Table 13.

**Table 13.** Search Space for Bayesian Optimization of XGBoost Hyperparameters

Parameter	Search Space Range
n_estimators	(5, 30)
max_depth	(2, 12)
learning_rate	(0.003, 0.1)
colsample_bytree	(0.5, 1)



## F Additional Privacy-Preserving XGBoost Experiments

Experiments varying `n_estimators`, `max_depth`, and `n_bits` provided insights into balancing model performance and FHE inference time (Tables 14, 15, 16).

**Table 14.** Performance and time metrics of XGBoost models with varying `n_estimators`. Experiment was conducted on Dataset 1 at fixed `n_bits` = 3, `learning_rate` = 0.07, `colsample_bytree` = 0.98, `max_depth` = 3.

n_estimators	Accuracy		F1-Score		Inference Time (s)		Time Ratio
	Clear	FHE	Clear	FHE	Clear	FHE	
5	0.6428	0.6428	0.7822	0.7822	0.009405	1401.32	149000x
10	0.6428	0.6428	0.7822	0.7822	0.013848	2222.64	160504x
50	0.6432	0.6432	0.7824	0.7824	0.042257	26595.60	629379x
100	0.6483	0.6483	0.7846	0.7846	0.205946	55523.89	269604x
200	0.6483	0.6468	0.7846	0.7833	0.174605	65481.47	375027x

**Table 15.** Performance and time metrics of XGBoost models with varying `max_depth`. Experiment was conducted on Dataset 1 at fixed `n_bits` = 3, `learning_rate` = 0.07, `colsample_bytree` = 0.98, `n_estimators` = 20.

max_depth	Accuracy		F1-Score		Inference Time (s)		Time Ratio
	Clear	FHE	Clear	FHE	Clear	FHE	
1	0.3584	0.3584	0.0000	0.0000	0.019856	4014.30	202173.40x
4	0.5540	0.5540	0.6809	0.6809	0.121287	29193.94	240701.55x
7	0.5540	0.5540	0.6809	0.6809	1.873080	47420.48	25316.84x
10	0.5540	0.5540	0.6809	0.6809	0.228830	40407.25	176581.86x
13	0.5540	0.5540	0.6809	0.6809	0.184337	41289.46	223988.81x

**Table 16.** Performance and time metrics of XGBoost models with varying `n_bits`. Experiment was conducted on Dataset 1 at fixed `learning_rate` = 0.07, `colsample_bytree` = 0.98, `n_estimators` = 20, `max_depth` = 3.

n_bits	Accuracy		F1-Score		Inference Time (s)		Time Ratio
	Clear	FHE	Clear	FHE	Clear	FHE	
2	0.4540	0.4540	0.4553	0.4553	0.042825	7244.70	169168.00x
3	0.5540	0.5540	0.6809	0.6809	0.055623	11766.27	211534.87x
4	0.6432	0.6432	0.7824	0.7824	0.055750	30344.95	544307.47x
8	0.6428	0.6428	0.7821	0.7821	0.023308	32139.84	1378944.61x