PrivNet.AI: Privacy-Preserving Learning on Encrypted Graphs via Geometric Deep Learning and Post-Quantum Cryptography

Chiman Soltanian
HTL Wien West / Data Science Aspirant

May 20, 2025

Contents

1 Introduction

We propose **PrivNet.AI**, an open-source research framework for privacy-preserving machine learning on sensitive graph-structured data, using homomorphic encryption, zero-knowledge proofs, and post-quantum cryptographic primitives (notably isogeny-based and lattice-based cryptography). Leveraging recent advances in *Geometric Deep Learning* (GDL), we develop secure protocols for training *Graph Neural Networks* (GNNs) on encrypted data representations like genome interaction networks, social-financial graphs, and knowledge graphs.

1.1 Motivation

In modern applications (e.g., personalized medicine, financial compliance, scientific data sharing), the trade-off between **privacy** and **progress** remains unresolved. Institutions require collaborative machine learning but cannot reveal sensitive graph structures or features due to legal and ethical constraints. Our framework addresses this with:

- **Training deep models on encrypted data**
- **End-to-end post-quantum-secured communication and storage**
- **Structure-aware encrypted computation using GDL**

1.2 Related Work

- **Homomorphic Encryption + ML**: CryptoNets (Microsoft, 2016); recent CKKS schemes enabling approximate encrypted arithmetic.

- **Zero-Knowledge GNN Inference**: zkGNNs (2022), enabling provable GNN predictions without revealing inputs.

- **Post-Quantum Cryptography**: NIST PQC candidates (Kyber, Dilithium, SIKE), Ring-LWE-based schemes for encryption and signatures.
- **Geometric Deep Learning**: Pioneered in Bronstein et al. (2017), applied to graphs, manifolds, and meshes.

2 Background

2.1 The Modern Paradox: AI Performance vs. Privacy

In the last decade, deep learning has transformed multiple domains—from healthcare and finance to national security—by leveraging vast quantities of structured data. However, the more intelligent AI becomes, the more data it requires, including sensitive personal and organizational information.

Privacy, thus, becomes the bottleneck of modern AI.

- Health data must comply with HIPAA and GDPR.
- Financial institutions are legally prohibited from exposing customer transactions.
- Social platforms cannot ethically or legally process user-level graphs without consent.

This leads to the central dilemma: *Can we build intelligent systems without seeing the data?*

2.2 The Rise of Privacy-Preserving Computation

In parallel, the field of cryptography has matured to offer powerful tools that allow:

- Computation over encrypted data via **Fully Homomorphic Encryption (FHE)**
- Collaborative model training via **Secure Multi-party Computation (SMPC)**
- Trust-free verification via **Zero-Knowledge Proofs (ZKPs)**
- Quantum-safe security via **Post-Quantum Cryptography (PQC)**

Yet, these cryptographic primitives remain underutilized in deep learning pipelines due to their computational complexity and poor interoperability with modern models.

2.3 The Need for Geometric Learning on Structured Private Data

Much of the world's sensitive data is **non-Euclidean**:

- Transaction graphs
- Protein–protein interaction networks
- Knowledge graphs and ontologies
- Smart grid and IoT sensor meshes

Geometric Deep Learning (GDL)—especially Graph Neural Networks (GNNs)—is the state-of-the-art method for such data. But applying GDL models to encrypted graphs is currently infeasible with standard pipelines.

2.4 The Quantum Threat to Cryptographic Infrastructure

The rise of quantum computing threatens most classical cryptographic protocols (RSA, ECC). Post-Quantum Cryptography (PQC) is essential to ensure that AI systems handling confidential data remain future-proof.

This implies that any privacy-preserving AI project must be:

- Secure against quantum attacks (lattice-based, isogeny-based)
- Capable of training on encrypted and/or decentralized data
- Able to prove its actions without revealing its internal states (via ZKPs)

2.5 Research Hypothesis Behind PrivNet.AI

The PrivNet.AI project is founded on the following hypothesis:

> *"It is possible to build a unified geometric deep learning framework that trains over encrypted, non-Euclidean data structures using post-quantum primitives while ensuring provable correctness through zero-knowledge proofs."*

2.6 Key Research Questions

1. How can we design GNN architectures that operate over encrypted node and edge features?
2. What are the theoretical limitations of message passing in ciphertext domains?
3. Can we integrate Ring-LWE-based homomorphic operations into GDL libraries?
4. How can we efficiently verify (without revealing) that encrypted training follows protocol?
5. What is the class of aggregation and pooling functions compatible with PQ-safe encryption schemes?

2.7 Intended Contributions of PrivNet.AI

- A research platform combining cryptographic algebra with deep learning geometry
- Open-source libraries for encrypted graph learning
- Whitepapers on theoretical limits of homomorphic geometric models
- Deployment-ready privacy-preserving AI solutions for financial, medical, and governmental domains

2.8 Broader Impact

The PrivNet.AI project envisions a future where:

- AI respects user and institutional privacy by design
- Model decisions are verifiable without revealing sensitive input
- Privacy-preserving AI can be a first-class citizen in post-quantum cybersecurity

It is not merely a project—it is a paradigm shift.

3 Geometric Deep Learning(GDLs)

Geometric Deep Learning is a modern field of machine learning that generalizes deep neural networks to *non-Euclidean* domains such as:

- Graphs (social networks, molecular structures)
- Manifolds (surfaces, 3D meshes)
- Symmetric spaces (used in physics)

Classical deep learning works well on data like images or text that lie in regular grids. But many real-world datasets are **irregular**, e.g.,:

- Social networks (nodes = people, edges = relationships)
- Protein–protein interaction networks
- Knowledge graphs

GDL aims to bring the power of deep learning to such domains by respecting their underlying geometry.

3.0.1 Graphs: The Language of Irregular Structure

A graph $G = (V, E)$ consists of:

- A set of **nodes** (or vertices) $V = \{v_1, \ldots, v_n\}$
- A set of **edges** $E \subseteq V \times V$, representing connections
- (Optionally) a feature matrix $X \in \mathbb{R}^{n\times d}$, where each row x_i represents node features
- An adjacency matrix $A \in \mathbb{R}^{n\times n}$, where $A_{ij} = 1$ if there is an edge from v_i to v_j

Goal: Predict labels for nodes, edges, or the entire graph using deep learning.

3.0.2 Message Passing Neural Networks (MPNNs)

Graph Neural Networks work based on the *message passing* principle:

> Each node updates its feature by aggregating information ("messages") from its neighbors.

3.0.3 One Message Passing Layer

At each layer l, each node v_i updates its feature as follows:

$$h_i^{(l+1)} = \sigma\left(\sum_{j\in\mathcal{N}(i)} W^{(l)} h_j^{(l)} + b^{(l)}\right)$$

Where:

- $h_i^{(l)}$ is the feature vector of node i at layer l
- $\mathcal{N}(i)$ is the set of neighbors of node i
- $W^{(l)}$ is a learnable weight matrix
- σ is an activation function (e.g., ReLU)

3.0.4 Graph Convolutional Networks (GCNs)

One of the most popular GNN architectures is the Graph Convolutional Network (GCN) by Kipf and Welling (2017).

GCN Layer Equation

The update rule is:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(l)}W^{(l)}\right)$$

Where:

- $\tilde{A} = A + I$ is the adjacency matrix with self-loops
- $\tilde{D}$ is the degree matrix of $\tilde{A}$
- $H^{(0)} = X$ is the initial node feature matrix
- $W^{(l)}$ is a learnable weight matrix
- σ is an activation function

Intuition

Each node takes a *weighted average* of its neighbors' features, then applies a transformation and non-linearity.

3.0.5 Example: Node Classification

Given:

- A citation network where each node is a paper
- Edges represent citations
- Node features represent word vectors from the abstract
- Labels are subject areas (e.g., Math, Biology)

Task: Use a GCN to predict the subject of each paper.

It should be mensioned that, some applications of GNNs are:

- **Finance:** Fraud detection on transaction graphs
- **Cryptography/Privacy:** Training models on structured data with privacy guarantees

3.1 Mathematical Preliminaries

3.1.1 Manifolds and Metric Spaces

Let $\mathcal{M}$ be a smooth d-dimensional Riemannian manifold with metric g.

- The **Laplace–Beltrami operator** $\Delta_{\mathcal{M}}$ is the generalization of the Laplacian to manifolds.
- Eigenfunctions of $\Delta_{\mathcal{M}}$ provide a frequency basis for functions on $\mathcal{M}$ (spectral methods).

3.1.2 Group Theory and Symmetry

Let G be a group acting on a space X.

Equivariance: A function $f : X \to Y$ is equivariant if:

$$f(g \cdot x) = g \cdot f(x), \quad \forall g \in G$$

This generalizes convolution: translation equivariance in CNNs becomes group equivariance in GDL.

3.2 Generalized Convolutions on Non-Euclidean Domains

3.2.1 Spectral Convolution on Graphs

For a graph $G = (V, E)$ with Laplacian $L = D - A$, let $L = U \Lambda U^T$ be the eigendecomposition.

Graph Fourier Transform:

$$\hat{f} = U^T f$$

Spectral convolution:

$$f *_G g = U(U^T f \odot U^T g)$$

3.2.2 Spatial Convolution via Message Passing

Instead of using Laplacian eigenbasis, spatial methods define local neighborhoods:

$$h_v^{(l+1)} = \sigma \left(\sum_{u \in \mathcal{N}(v)} W^{(l)} h_u^{(l)} \right)$$

This is the foundation of GNNs and generalizes to mesh networks and point clouds.

3.3 Core GDL Algorithmic Patterns

3.3.1 Algorithm 1: Spectral Convolution on Manifold/Graph

Algorithm 1 Spectral Convolution

```
Require: Signal $f$ on vertices, Laplacian $L$
Compute eigendecomposition $L = U \Lambda U^T$
Compute Fourier coefficients: $\hat{f} \leftarrow U^T f$
Multiply by filter: $\hat{f}_{\text{filtered}} \leftarrow \hat{f} \odot \hat{g}$
Inverse transform: $f_{\text{out}} \leftarrow U \hat{f}_{\text{filtered}}$
return $f_{\text{out}}$
```

3.3.2 Algorithm 2: Message Passing Neural Network (MPNN)

Algorithm 2 Message Passing on Graph

```
Require: Graph $G = (V, E)$, feature matrix $X$
for each layer $l = 0$ to $L - 1$ do
    for each node $v \in V$ do
        Aggregate messages: $m_v^{(l)} \leftarrow \mathrm{AGG}(\{h_u^{(l)} : u \in \mathcal{N}(v)\})$
        Update state: $h_v^{(l+1)} \leftarrow \mathrm{UPD}(h_v^{(l)}, m_v^{(l)})$
    end for
end for
return Final node features $\{h_v^{(L)}\}$
```

4 Graph Neural Networks (GNNs)

4.1 Mathematical Preliminaries

4.1.1 Graph Basics

Let $G = (V, E)$ be an undirected graph:

- $V = \{v_1, v_2, \dots, v_n\}$: nodes (vertices)
- $E \subseteq V \times V$: edges
- $A \in \mathbb{R}^{n \times n}$: adjacency matrix, where $A_{ij} = 1$ if $(v_i, v_j) \in E$
- $X \in \mathbb{R}^{n \times d}$: feature matrix, x_i is the feature vector of node v_i
- $D = \text{diag}(d_1, \dots, d_n)$: degree matrix where $d_i = \sum_j A_{ij}$

We also define the normalized adjacency matrix:

$$\hat{A} = D^{-1/2}(A + I)D^{-1/2}$$

4.1.2 Graph Laplacian

The (combinatorial) graph Laplacian is:

$$L = D - A$$

The normalized Laplacian is:

$$\mathcal{L} = I - D^{-1/2}AD^{-1/2}$$

4.2 Message Passing Framework

All GNNs can be seen under the umbrella of **Message Passing Neural Networks (MPNNs)**.

At layer l, each node updates its state by aggregating messages from neighbors:

$$h_v^{(l+1)} = \text{UPDATE}^{(l)}\left(h_v^{(l)}, \text{AGGREGATE}^{(l)}\left(\{h_u^{(l)} : u \in \mathcal{N}(v)\}\right)\right)$$

This includes three steps:

1. **Message computation**: $m_{uv} = M(h_u, h_v, e_{uv})$
2. **Aggregation**: $m_v = \text{AGGREGATE}(\{m_{uv}\}_{u \in \mathcal{N}(v)})$
3. **Update**: $h_v^{(l+1)} = U(h_v^{(l)}, m_v)$

4.3 Graph Convolutional Networks (GCNs)

GCNs simplify message passing into a single propagation rule:

4.3.1 GCN Layer Formula (Kipf & Welling, 2017)

$$H^{(l+1)} = \sigma\left(\hat{A}H^{(l)}W^{(l)}\right)$$

Where:

- $H^{(0)} = X$ is the input node feature matrix
- $\hat{A} = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$, with $\tilde{A} = A + I$
- $W^{(l)}$: trainable weight matrix at layer l
- σ: non-linearity (e.g., ReLU)

4.3.2 Algorithm: Forward Pass of a 2-layer GCN

Algorithm 3 2-Layer GCN Forward Pass

```
Require: Feature matrix $X$, adjacency matrix $A$, weights $W^{(0)}, W^{(1)}$
$\tilde{A} \leftarrow A + I$
$\tilde{D}_{ii} \leftarrow \sum_j \tilde{A}_{ij}$
$\hat{A} \leftarrow \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$
$H^{(1)} \leftarrow \text{ReLU}(\hat{A}XW^{(0)})$
$Z \leftarrow \text{softmax}(\hat{A}H^{(1)}W^{(1)})$
return $Z$ (predicted labels or embeddings)
```

4.4 Graph Attention Networks (GATs)

GATs use attention weights to decide how much attention to pay to each neighbor.

4.4.1 Layer Formula

$$h_i^{(l+1)} = \sigma\left(\sum_{j\in\mathcal{N}(i)} \alpha_{ij} W h_j^{(l)}\right)$$

Where:

$$\alpha_{ij} = \text{softmax}_j\left(\text{LeakyReLU}(a^T[Wh_i \| Wh_j])\right)$$

- W: shared linear transformation
- a: attention vector
- $\|$: concatenation

4.5 Training GNNs

4.5.1 Loss Functions

For node classification:

$$\mathcal{L} = -\sum_{i\in\mathcal{V}_{\text{labeled}}} \sum_{k=1}^{C} y_{ik} \log \hat{y}_{ik}$$

Where:

- y_{ik}: ground truth label (one-hot)
- $\hat{y}_{ik}$: predicted softmax output

4.5.2 Backpropagation on Graphs

Gradient flow is similar to standard neural networks, except gradients are computed w.r.t. shared weights $W^{(l)}$ across nodes, and matrix products involve sparse $\hat{A}$.

4.6 Theoretical Expressivity

4.6.1 Weisfeiler–Lehman (WL) Test

The power of a GNN is often compared to the WL test for graph isomorphism. A GNN is as powerful as WL if:

$$h_v^{(l+1)} = \text{HASH}\left(h_v^{(l)}, \text{MULTISET}\{h_u^{(l)} : u \in \mathcal{N}(v)\}\right)$$

4.6.2 Universal Approximation

Under certain conditions, GNNs can approximate any function on graphs that is permutation invariant:

$$f(\mathcal{G}) = f(\pi(\mathcal{G}))$$

5 Examples of GDLs and GNNs

Graph Neural Networks (GNNs) are models that learn from graph-structured data using message passing and neighborhood aggregation. Geometric Deep Learning (GDL) generalizes neural networks to non-Euclidean domains, including graphs, manifolds, and groups.

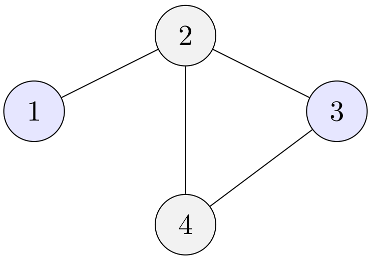
This document provides basic step-by-step examples of how GNNs work on simple graphs.

5.1 Example 1: Node Classification with GNN

Consider a simple graph of 4 nodes:

- Each node has a feature value x_i
- Nodes 1 and 3 are labeled, nodes 2 and 4 are unlabeled
- The goal is to classify nodes 2 and 4

Graph Diagram



Node Features

- $x_1 = [1]$ (label: class 0)
- $x_2 = [0.5]$ (unlabeled)
- $x_3 = [0]$ (label: class 1)
- $x_4 = [0.3]$ (unlabeled)

GNN Layer Update

Use simple aggregation: mean of neighbors' features, followed by a linear transformation and ReLU.

Let:

$$W = [2], \quad \sigma(x) = \mathrm{ReLU}(x) = \max(0, x)$$

Step-by-step:

- Node 2's neighbors: 1, 3, 4

$$h_2^{(1)} = \sigma\left(W \cdot \frac{x_1 + x_3 + x_4}{3}\right) = \sigma\left(2 \cdot \frac{1 + 0 + 0.3}{3}\right) = \sigma(0.867) = 0.867$$

- Node 4's neighbors: 2 and 3

$$h_4^{(1)} = \sigma\left(W \cdot \frac{x_2 + x_3}{2}\right) = \sigma\left(2 \cdot \frac{0.5 + 0}{2}\right) = \sigma(0.5) = 0.5$$

Interpretation

- Node 2's updated embedding is closer to node 1 (class 0) ⇒ predict class 0
- Node 4's updated embedding is between node 2 and node 3 ⇒ ambiguous, but with more layers this can be clarified

5.2 Example 2: Graph Convolution on Small Adjacency Matrix

Adjacency Matrix

Let:

$$A = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \quad \text{(with self-loops)}$$

Feature matrix:

$$X = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}$$

Degree matrix:

$$D = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

Normalized adjacency:

$$\hat{A} = D^{-1/2} A D^{-1/2}$$

Apply GCN layer:

$$H^{(1)} = \sigma(\hat{A} X W)$$

With:

$$W = 1, \quad \sigma = \text{identity}$$

Compute:

$$H^{(1)} \approx \hat{A} X = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{6} & 0 \\ 1/\sqrt{2} & 1/\sqrt{6} & 1/\sqrt{6} \\ 0 & 1/\sqrt{6} & 1/\sqrt{2} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix} \approx \begin{bmatrix} 0.707 \\ 1.152 \\ 1.414 \end{bmatrix}$$

Interpretation

The GCN aggregates neighbor information by blending node values through normalized adjacency, improving classification.

Conclusion

These examples show how GNNs update node embeddings by aggregating neighbor features and applying transformations. The flexibility of GDL allows it to handle various graph types, making it suitable for tasks like classification, regression, and representation learning on structured data.

Future examples can include:

- Edge classification
- Graph-level prediction (e.g., molecular property prediction)
- Message-passing on dynamic graphs

6 Training Deep Models on Encrypted Data Using GDL

6.1 Introduction

In many applications (genomics, healthcare, finance), data is sensitive and must remain private. Yet we want to use this data to train powerful deep learning models — especially on structured data like graphs and manifolds.

Goal: Train deep models like GNNs on *encrypted* data using **Geometric Deep Learning (GDL)**, while preserving data confidentiality.

6.2 Problem Setting

We assume:

- Sensitive structured data X (e.g., node features in a graph)
- A known graph structure $G = (V, E)$
- A client encrypts X before sending it to the server
- The server trains a GNN on the encrypted data without ever seeing raw values

6.3 Cryptographic Tools

6.3.1 Homomorphic Encryption (HE)

Definition: Homomorphic Encryption allows computations directly on encrypted data.

- Encryption: $\mathcal{E}(x)$
- Evaluation: $f(\mathcal{E}(x)) = \mathcal{E}(f(x))$

Types:

- Partial HE (PHE): supports either addition or multiplication
- Somewhat HE (SHE): supports limited depth of operations
- Fully HE (FHE): supports arbitrary functions over ciphertexts

Popular libraries: Microsoft SEAL, HElib, PALISADE

6.3.2 Secure Aggregation (Optional)

In federated or multi-party settings, aggregation of encrypted gradients across clients is performed securely without revealing local updates.

6.4 Geometric Deep Learning Recap

GDL generalizes deep learning to non-Euclidean domains, especially:

- Graphs: use GNNs (Graph Neural Networks)
- Meshes and manifolds: use spectral convolutions

GNN Layer (Unencrypted)

$$H^{(l+1)} = \sigma\left(\hat{A}H^{(l)}W^{(l)}\right)$$

where:

- $\hat{A}$ is normalized adjacency
- $H^{(0)} = X$ is the feature matrix
- $W^{(l)}$ is a trainable weight matrix

6.5 Encrypted GNN Computation

Key idea: Replace X with encrypted data $\mathcal{E}(X)$ and train the model directly on it.

Step-by-step Encrypted Forward Pass

1. **Input encryption:** client sends $\mathcal{E}(X)$
2. **Matrix multiplication:** compute $\mathcal{E}(X)W^{(0)}$ using homomorphic ops
3. **Aggregation:** encrypted summation over neighbors:

$$\mathcal{E}(h_i^{(1)}) = \sigma\left(\sum_{j\in\mathcal{N}(i)} \mathcal{E}(x_j)W^{(0)}\right)$$

4. **Non-linearities:** Use *polynomial approximations* of ReLU (e.g., square functions)
5. **Loss computation:** compute encrypted loss $\mathcal{E}(L)$ and send back to client for decryption

Homomorphic-friendly Functions

Since encryption schemes support basic arithmetic:

- Use low-degree polynomials for activations (e.g., $\sigma(x) = x^2$, $\tanh(x) \approx x - x^3/3$)
- Avoid softmax unless approximated

7 GDL and GNN in Post-Quantum Cryptography

7.1 Introduction

The emergence of quantum computing threatens traditional cryptographic primitives (RSA, ECC, DSA) based on factoring and discrete logarithm problems. This has led to the development of **Post-Quantum Cryptography (PQC)** — cryptographic algorithms believed to be secure against quantum attacks.

At the same time, **Geometric Deep Learning (GDL)** — especially **Graph Neural Networks (GNNs)** — has emerged as a powerful tool for learning over structured, non-Euclidean data such as lattices, graphs, and isogeny structures.

This note explores the novel intersection of these two fields, detailing how GDL methods contribute to PQC research, implementation, cryptanalysis, and even the design of future cryptographic systems.

7.2 Background: Post-Quantum Cryptography

7.2.1 Motivation

Quantum computers can break:

- RSA using Shor's algorithm (for integer factorization)
- ECC using Shor's algorithm (for discrete logs over elliptic curves)
- Symmetric encryption via Grover's algorithm (quadratic speedup)

7.2.2 Main PQC Families

- **Lattice-based cryptography** (e.g., NTRU, Kyber, FrodoKEM)
- **Code-based cryptography** (e.g., McEliece)
- **Multivariate polynomial cryptography** (e.g., Rainbow)
- **Hash-based signatures** (e.g., SPHINCS+)
- **Isogeny-based cryptography** (e.g., SIDH/SIKE, CSIDH)

NIST selected Kyber (lattice-based) and SPHINCS+ (hash-based) for standardization in 2022.

7.3 Geometric Deep Learning Primer

GDL is a unification of deep learning methods over non-Euclidean spaces. In the context of PQC, we focus primarily on **Graph Neural Networks (GNNs)**.

7.3.1 Why Graphs?

Many cryptographic structures can be naturally represented as graphs:

- **Lattices:** basis reduction graphs, shortest/closest vector transitions
- **Code-based:** Tanner graphs of codes
- **Isogeny-based:** graphs of isogenies between elliptic curves

7.3.2 GNN Framework

A typical GNN layer:

$$h_v^{(l+1)} = \sigma \left(\sum_{u \in \mathcal{N}(v)} W^{(l)} h_u^{(l)} \right)$$

This enables local feature aggregation across a cryptographic structure.

7.4 Recent Advances: GDL in PQC

7.4.1 1. GNNs in Cryptanalysis of Lattice-based Schemes

- GNNs have been used to **learn patterns in lattice instances**, including shortest vector problem (SVP) graphs.
- Recent studies use GNNs to predict the **hardness of lattice problems**, helping tune security parameters.
- Feature engineering includes Gram-Schmidt lengths, angle histograms, and reduction trajectories.

7.4.2 2. Graph Learning for Isogeny-Based Cryptography

- The isogeny graph of elliptic curves (as used in SIDH, CSIDH) has a rich graph structure.
- GNNs can be trained to **predict paths or vulnerabilities** in the isogeny walk.
- Spectral GNNs or message passing models have been tested on supersingular isogeny graphs.

7.4.3 3. Adversarial Learning and Attack Prediction

- GDL models can classify PQ keypairs or ciphertexts as "vulnerable" based on structural graph metrics.
- Graph attention networks (GATs) can identify **high-risk edges or substructures** in cryptographic graphs.
- This may lead to new side-channel detection tools.

7.4.4 4. Optimization of Cryptographic Parameters

- Cryptographic schemes often depend on complex parameter sets (e.g., NTRU: q, n, noise distribution).
- GDL models can help in **parameter selection** by training over graphs of failure probabilities and performance metrics.

7.5 5. Learning over Group and Ring Structures

- Many PQ schemes operate over algebraic structures (rings, modules, abelian groups).
- GDL models can exploit **group-equivariant architectures** (e.g., SE(3)-GNNs, spherical CNNs) to learn over such data.

8 Examples: The Role of GDL and GNNs in PQC

Introduction

Post-Quantum Cryptography (PQC) aims to develop cryptographic algorithms that are secure against attacks from quantum computers.

Geometric Deep Learning (GDL) and Graph Neural Networks (GNNs) are powerful tools for learning patterns in structured data, such as graphs. These tools can be used to:

- Understand cryptographic structures (e.g., lattices, isogeny graphs)
- Predict vulnerabilities
- Optimize parameters
- Automate analysis and design of cryptographic systems

8.1 Clarification: Why Are Graphs Important in PQC?

Many PQ cryptographic schemes naturally produce graph-like structures:

- **Lattice-based cryptography:** basis vectors form a geometric lattice; reduction steps form a transition graph
- **Code-based cryptography:** codes can be represented as Tanner graphs
- **Isogeny-based cryptography:** elliptic curves connected by isogenies form a graph

These structures are hard to analyze by hand. GDL offers tools to learn from them automatically.

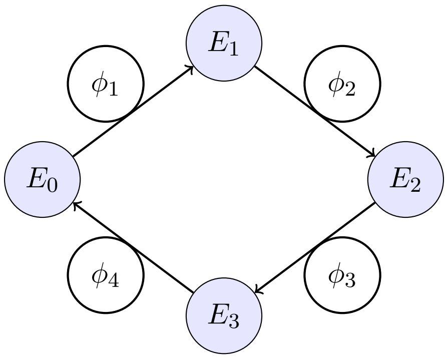
8.2 Clarification: What Do GNNs Do in This Context?

A Graph Neural Network can:

- Learn node embeddings from isogeny or lattice graphs
- Classify nodes or subgraphs (e.g., weak instances vs. secure ones)
- Predict optimal parameters for cryptographic constructions
- Simulate isogeny walks or reduction paths

8.3 Example: Isogeny Graph in Supersingular Curves

Let us visualize a small isogeny graph of supersingular elliptic curves.



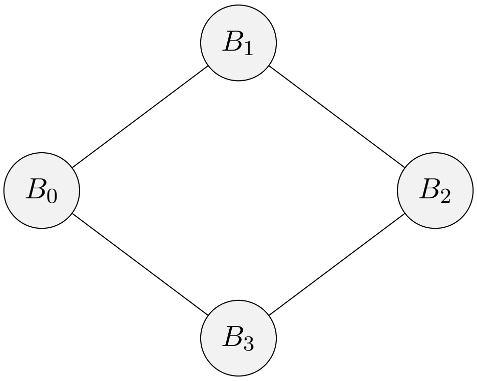
This graph shows elliptic curves E_i connected via isogenies ϕ_i. A GNN can be trained to:

- Identify paths that reveal trapdoor information
- Predict which nodes (curves) are vulnerable
- Suggest optimal jump sequences for hardening cryptographic keys

8.4 Example: Learning Lattice Weaknesses

In lattice-based cryptography, finding a short vector in a lattice is hard. However, some lattice instances may be easier than others.

Idea: represent basis transitions as a graph.



Nodes are basis vectors, edges are reduction steps.
A GNN can:

- Predict how "hard" a basis is for Shortest Vector Problem (SVP)
- Suggest better reduction paths
- Learn which reductions lead to weak lattice instances

8.5 Clarification: Post-Quantum Security of GDLs

A related concern is that machine learning itself must be secure against quantum threats. Therefore:

- All data and models must be encrypted using **post-quantum schemes** (e.g., Ring-LWE, SIDH)
- GNNs may operate on encrypted graphs
- Predictions must be verified without leaking information (e.g., using ZKPs)

8.6 What Is Post-Quantum Cryptography?

- **Goal**: Design cryptosystems secure even if large quantum computers exist.
- **Families**: Lattice-based (e.g. Learning With Errors), code-based (McEliece), isogeny-based (SIDH/SIKE), hash-based, multivariate.
- **Challenge**: Hard mathematical problems must resist both classical and quantum algorithms.

8.6.1 What Are GDL and GNNs?

- **GDL**: Extends deep learning to non-Euclidean domains (graphs, manifolds).
- **GNNs**: A family of GDL models that perform *message passing*—each node updates its state by aggregating information from its neighbors.
- **Why graphs in PQC?**
  - *Lattices*: Can represent basis reduction as a graph of states.
  - *Isogeny graphs*: Vertices are elliptic curves; edges are isogenies.
  - *Code Tanner graphs*: Represent parity-check relationships.

8.6.2 How Can GNNs Help PQC?

(a) **Cryptanalysis**: Learn patterns in hard problems (e.g. identify "easy" lattice instances).

(b) **Parameter Selection**: Predict security/performance trade-offs for scheme parameters.

(c) **Attack Path Prediction**: On an isogeny graph, learn which paths yield weak keys.

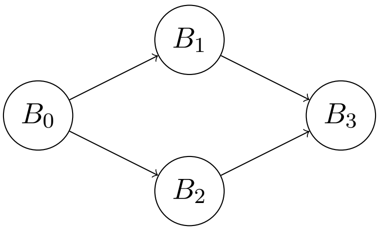
(d) **Designing New Primitives**: Use GNN-inspired architectures to craft cryptographic maps.

8.7 Example 1: Lattice-Based Cryptography and GNNs

8.7.1 Lattice Reduction as a Graph

A lattice basis reduction process (e.g. BKZ) can be viewed as a graph:

$$\text{Nodes } = \{\text{basis states}\}, \quad \text{Edges } = \{\text{basis transformation steps}\}.$$



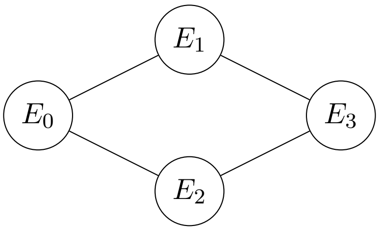
8.7.2 GNN Application

- *Features*: Basis quality measures (Gram–Schmidt lengths, orthogonality).
- *GNN task*: Predict which reduction path leads to a short vector fastest.
- *Benefit*: Guide parameter tuning (block size, reduction steps) for desired security/cost.

9 Example 2: Isogeny Graphs and Path Finding

9.0.1 Isogeny Graph Structure

Vertices are supersingular elliptic curves; edges are isogenies of fixed degree.



9.0.2 GNN Application

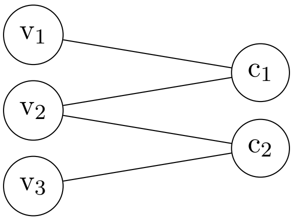
- *Features*: Curve invariants, endomorphism ring data.
- *GNN task*: Classify which paths (walks) lead to a known private key.
- *Benefit*: Identify vulnerable isogeny steps, inform protocol defenses.

9.1 Example 3: Code-Based Cryptography

9.1.1 Tanner Graph of a Code

Parity-check matrix H gives a bipartite graph:

$$\text{Variable nodes (bits)} \leftrightarrow \text{Check nodes (parity equations).}$$



9.1.2 GNN Application

- *Features*: Bit reliability metrics, syndrome values.
- *GNN task*: Enhance decoding by learning optimized message updates.
- *Benefit*: Faster or more accurate decoding, affecting security margins.

9.2 Summary of Benefits

- **Pattern Discovery**: GNNs learn complex structures in PQC graphs.
- **Parameter Tuning**: Predict security vs. performance trade-offs.
- **Vulnerability Detection**: Highlight weak graph regions (attack paths).
- **Hybrid Design**: Inspire new PQ crypto primitives via learned architectures.

9.3 Conclusion

By treating PQC problems as graph-structured learning tasks, GDL and GNNs provide a powerful, data-driven complement to traditional algebraic methods. These beginner-friendly examples illustrate how simple graph representations and GNN models can open new research directions in securing cryptography for the quantum era.

10 Conclusions and Future Work

10.0.1 A. GNNs as Cryptographic Tools

GNNs may not only analyze but define cryptographic transformations. For example:

- Learnable isogeny walks
- Adaptive noise shaping in LWE schemes
- Learned trapdoor permutations via geometric transformations

10.0.2 B. Quantum-Resistant Privacy with GDL

Combining:

- GNN-based models for structured encrypted data
- Fully Homomorphic Encryption (FHE) or Multi-party Computation (MPC)
- Post-quantum primitives (e.g., Ring-LWE, SIDH) for encryption layer

enables **training GDL models over encrypted, PQ-safe data**.

10.0.3 C. Explainable PQC via GNNs

GNNs provide a natural way to visualize and explain which parts of the cryptographic structure are:

- Most vulnerable (based on attention maps)
- Most critical for performance
- Algorithmically optimal (in sampling and key generation)

10.0.4 D. Privacy-Preserving GNN Inference for PQ Protocols

Scenario: a cryptographic protocol accepts structured input (graph, code, lattice) and returns a response. The GNN runs on the encrypted input:

$$f(\mathcal{E}(x)) = \mathcal{E}(f(x))$$

This builds PQ-secure AI inference pipelines.

10.1 Challenges

- **Lack of datasets:** PQ cryptography lacks large open graph datasets
- **Noise in encryption:** encrypted data is noisy and non-differentiable
- **Computational cost:** GDL and PQC are both expensive
- **Explainability:** Understanding GNN decisions in cryptographic contexts is hard