A Deep Dynamic Graph Generative Framework for Blockchain Phishing Detection

Siyi Xiao, Lejun Zhang, Xinwei Zhang, *Graduate Student Member, IEEE,* Sen Zhang, Shen Su, Jing Qiu, Ran Guo, Haibo Hu, *Senior Member, IEEE*

***Abstract*—Blockchain phishing scams cause billions in annual losses, yet extreme data imbalance severely hampers existing detection algorithms. Current dynamic graph generation methods fragment structures and generate erroneous connections, failing to capture local dynamic patterns vital for node classification. This raises critical questions: Can models minimize isolated subgraph generation? How can they learn and replicate structured, recurring interaction patterns?**

To answer these questions, we introduce GraphFlowGen, an end-to-end deep generative framework. To minimize isolated subgraph generation, GraphFlowGen employs a novel preprocessing module that jointly extracts structural and temporal contexts from transaction data, preventing fragmentation and information loss. To learn and replicate structured interaction patterns, it incorporates a Transformer encoder with Graph Attention Networks (GAT) to capture node connection dynamics and temporal evolution. To ensure high fidelity while reducing erroneous links, a reinforcement learning (RL) mechanism iteratively refines generated graph structures. Empirical validation on three real-world datasets demonstrates the effectiveness of our algorithm in local dynamic graph generation and its utility for downstream phishing detection tasks.

***Index Terms*—Dynamic graph generation, Blockchain, Phishing detection, Data augmentation**

I. Introduction

ETHEREUM [1], as one of the most prominent blockchain platforms, leverages decentralized ledger technology to enable secure peer-to-peer transactions without relying on trusted intermediaries. Built upon foundational principles such as distributed consensus, immutability, and cryptographic security, Ethereum further supports smart contracts—self-executing programs that have driven transformative innovations in domains including finance, supply chain management, and decentralized applications (dApps) [2].

However, its rapid adoption has also attracted a growing number of malicious actors [3], leading to a surge in blockchain-based phishing scams. These attacks commonly deceive users through tactics such as counterfeit wallet interfaces, fraudulent token airdrops, and impersonated smart contracts [4], resulting in billions of dollars in annual losses and significantly eroding trust in the broader ecosystem.

To address this challenge, researchers have developed graph learning–based detection approaches that model Ethereum transactions as dynamic networks. For instance, Jung et al. [5] employed data mining techniques for fraud detection, while Chen et al. [6] proposed a graph embedding method to identify phishing accounts based on transaction topology. Although these methods show promising results, their effectiveness is fundamentally constrained by severe class imbalance: in real-world datasets, phishing nodes constitute only a very small fraction of all addresses [7], [8]. Consequently, models tend to overfit to benign patterns and exhibit limited sensitivity to rare malicious instances.

A promising direction to mitigate this issue is synthetic data augmentation via dynamic graph generation. Recent frameworks such as TagGen [9] and TIGGER [10] generate temporal graphs using random walks or point processes, primarily optimizing for global statistical properties (e.g., degree distribution). Nevertheless, these approaches are not well suited for security-sensitive tasks. Phishing detection relies less on macro-level graph characteristics and more on fine-grained, local behavioral signatures—such as star-shaped outflows, sudden bursts in transaction activity, or persistent interaction patterns among small groups of nodes. Moreover, existing generators often fragment node trajectories across time or construct disconnected subgraphs by stitching them together with heuristic edge-formation rules, which can lead to temporally inconsistent neighborhood structures and spurious connections that distort the very behavioral cues essential for accurate detection.

As a result, current graph generation techniques fall short of meeting the specific demands of security-focused synthesis. We contend that effective generation for imbalanced phishing detection must fulfill three essential criteria:

- **Temporal Coherence:** Maintain consistent evolution of node neighborhoods over time, preventing disjointed trajectories that arise from processing snapshots independently or relying on biased sampling strategies.

This work is sponsored by the National Natural Science Foundation of China No. U24B20148, 62472111, W2511073, U2436208, U2468204, U24A20336, 62302114 and No. 62372129. Innovation Fund Program of the Engineering Research Center for Integration and Application of Digital Learning Technology of Ministry of Education No.1331007. Natural Science Foundation of Guangdong Province No. 2024A1515010177. The Project of Guangdong Key Laboratory of Industrial Control System Security No. 2024B1212020010. Guangdong S&T Program under Grant 2024B0101010002. Yangzhou University International Academic Exchange Fund No.YZUF2024106. Research Grants Council (Grant No: C2004-21GF and C2003-23Y), Hong Kong SAR, China.

Siyi Xiao is with the College of Information Engineering, Yangzhou University, Yangzhou 225127, China.

Lejun Zhang, Shen Su, Ran Guo and Jing Qiu are with the Cyberspace Institute Advanced Technology, Guangzhou University, Guangzhou 510006, China.

Xinwei Zhang, Sen Zhang, and Haibo Hu are with the Department of Electrical and Electronic Engineering, The Hong Kong Polytechnic University, Hong Kong SAR, China.

Corresponding author: Lejun Zhang. Email: zhanglejun@gzhu.edu.cn

- **Topological Fidelity:** Preserve realistic local connectivity around known phishing nodes, avoiding the introduction of artificial edges or the disruption of authentic relationships through arbitrary graph modifications.
- **Behavior-Aware Dynamics:** Accurately replicate fine-grained node-level activity patterns, including bursts in transaction frequency, stability in functional roles, and diversity in interaction partners, all of which are characteristic of actual phishing operations.

Considering these criteria, we introduce *GraphFlowGen*, an end-to-end deep generative framework designed specifically to synthesize high-fidelity temporal transaction graphs that embody the structural and behavioral traits of phishing behavior. GraphFlowGen consists of three primary components: a preprocessing module for joint structural-temporal context extraction, a hybrid encoder combining Transformer and Graph Attention Network (GAT) architectures, and a reinforcement learning–based optimization mechanism.

To ensure **temporal coherence**, the preprocessing module constructs continuous node trajectories by aligning transaction events across consecutive time intervals instead of treating each interval in isolation. Nodes receive appearance-order embeddings, and the Transformer processes historical sequences using positional encodings to capture both long-term trends and short-term deviations. An autoregressive strategy conditions the generation of each new snapshot on previously generated outputs, thereby enforcing smooth neighborhood evolution.

To ensure **topological fidelity**, the GAT component learns context-sensitive representations by attending to local adjacency structures, preserving genuine dependencies among transacting entities. Edge creation during synthesis is driven by learned attention weights rather than heuristic rules, reflecting authentic structural priors. The reinforcement learning module further enhances fidelity by penalizing topological anomalie, through a reward function that prioritizes realistic connectivity around minority-class seed nodes.

To support **behavior-aware dynamics**, the Transformer-GAT architecture jointly models node attributes, together with interaction patterns within local neighborhoods. The reinforcement learning agent iteratively refines the generation process, with rewards derived from behavioral similarity measures. This ensures that synthetic subgraphs exhibit the nuanced characteristics necessary for training effective phishing detectors.

With these designs, *GraphFlowGen* can generate realistic dynamic graphs without requiring labels as explicit input features, yet effectively guided by minority-class samples, making it especially suitable for augmenting limited phishing samples in highly imbalanced blockchain environments. While motivated by blockchain phishing detection, our framework is general-purpose and applicable to any dynamic graph domain requiring high-fidelity local structure synthesis.

In summary, our key contributions are:

- We provide the first systematic analysis of temporal graph generation requirements for imbalanced anomaly detection, identifying temporal coherence, topological fidelity, and behavior-aware dynamics as three fundamental dimensions.
- We propose *GraphFlowGen*, an end-to-end deep generative framework tailored for synthesizing high-quality minority-class temporal subgraphs. It uniquely integrates joint structural-temporal encoding with RL-guided optimization to preserve local dynamic patterns across time.
- We validate GraphFlowGen on multiple real-world dynamic graph datasets—including Ethereum (blockchain), Stack Overflow (user collaboration), and Wikipedia (editing interactions)—demonstrating its superior ability to generate realistic local structures (e.g., star motifs, wedges) and significantly boost downstream detection performance. Our results show consistent improvements over state-of-the-art baselines across diverse network types, confirming the generalizability and effectiveness of our approach.

The remainder of this paper is organized as follows. Section II reviews related work. Section III revisits problem definition and challenges. Section IV details the design of GraphFlowGen. Section V introduces the experimental setup and the evaluation results, and Section VI provides the conclusion.

II. Related Work

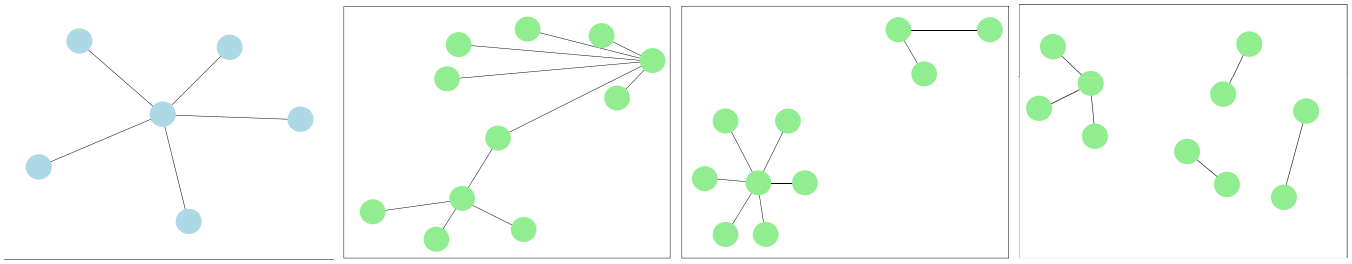
A. Phishing Scam Detection on Blockchain

Phishing scams on blockchain platforms, particularly Ethereum, have garnered significant attention due to their severe economic impact. Early approaches focused on rule-based and data mining techniques to identify fraudulent patterns in transaction networks. For instance, Jung *et al*. [5] employed data mining methods for Ethereum fraud detection, while Chen *et al*. [11] developed heuristics for detecting Ponzi schemes by analyzing transaction flows.

More recent works have leveraged graph-based representations to model Ethereum transactions as networks, enabling the capture of complex interactions. Chen *et al*. [6] proposed a graph embedding approach for phishing detection in Ethereum transaction networks, utilizing techniques inspired by DeepWalk [12] and node2vec [13] to embed accounts and detect anomalous clusters. Similarly, Wu *et al*. [14] utilized network embeddings to identify phishers through community analysis, incorporating inductive learning methods like GraphSAGE [15] for scalable representation. Advanced deep learning models have further improved detection accuracy; Other studies, such as those by Mighan *et al*. [16], explore forking-based attacks in Ethereum's proof-of-work consensus, highlighting vulnerabilities that phishing scams can exploit. Despite these advancements, a common challenge across these methods is the severe class imbalance in blockchain datasets, where fraudulent transactions are rare, leading to biased models prone to overfitting and reduced generalization [17]. Consequently, there is a growing need for data augmentation strategies, such as synthetic graph generation, to balance datasets and enhance detection performance in real-world.

B. Dynamic Graph Generation

Dynamic graph generation techniques aim to synthesize evolving networks that capture temporal dependencies, with



(a) (b) (c) (d)

Fig. 1: Comparison of graph snapshots: (a) true original graph; (b)TagGen [9], (c)TIGGER [10], and (d)DGGEN [27]: snapshots generated by three distinct algorithms.

applications in social sciences, finance, and blockchain security. These methods overcome limitations of static graph models by simulating structural changes over time. Surveys by Kazemi *et al*. [18] and Ju *et al*. [19] provide overviews of representation learning for dynamic graphs, highlighting temporal embeddings and recent advances through 2024. We categorize methods based on their mechanisms, noting strengths and areas for improvement in local and temporal modeling.

Autoregressive models, such as GraphRNN [20], sequentially predict nodes and edges, offering scalability but often relying on biased random walks [21], which fragment structures and inadequately capture node evolutions like degree or clustering changes. TagGen [9] generates graphs via temporal random walks and local perturbations. GAN-based approaches, including GraphGAN [22], DyngraphGAN [23], and GCN-GAN [24], leverage adversarial training for embeddings and link prediction. However, subgraph splicing can disrupt topologies and introduce spurious connections, reducing fidelity in sparse networks.

Diffusion-based models, like those in [25] and motif-aware MoDiff [26], provide probabilistic synthesis via continuous-time processes, improving noise handling over variational autoencoders. Yet, they involve high training costs, limiting real-time use. Hosseini *et al*. [27] offer an encoder-decoder framework for continuous-time generation. Temporal-specific models address evolutions through memory mechanisms (e.g., TGN [28], DyRep [29]), continuous embeddings [21], inductive walks [30], [31], and attention (GAT [32]). TIGGER [10] integrates point processes for scalable interaction graphs. These enhance reasoning but often neglect fine-grained node behaviors, such as evolving adjacency lists and neighborhood shifts. A persistent challenge is incomplete node behavior modeling, yielding graphs that fall short of real dynamics [33]. This prompts inquiries into reducing independent subgraph generation and learning recurrent patterns-core to our research.

GraphFlowGen mitigates these via a joint sampling strategy, Transformer-GAT encoding, and RL optimization, avoiding fragmentation and overheads for improved blockchain phishing detection.

III. Problem Definition and Challenges

In this section, we formally define the problem of phishing scam detection on blockchain platforms under data imbalance and introduce dynamic graph generation as a key technique to address it. We motivate the need for synthetic augmentation and outline how our GraphFlowGen framework leverages temporal graph modeling to generate realistic subgraphs for balancing datasets.

A. Problem Definition

We model the Ethereum transaction network as a discrete-time dynamic graph $G = \{G_t\}_{t=1}^{T}$, where each snapshot $G_t = (V_t, E_t, X_t)$ consists of a node set V_t representing unique addresses active during time window t, an edge set $E_t \subseteq V_t \times V_t$ denoting directed transactions, and a node feature matrix $X_t \in \mathbb{R}^{|V_t| \times d}$, where each row encodes address-level statistics such as in/out degree, transaction frequency, and balance change computed from raw ledger data. Each node $v \in V_t$ is associated with a binary label $y_v \in \{0, 1\}$, where $y_v = 1$ indicates a phishing account.

Given a historical sequence of snapshots $\{G_1, \cdots, G_{t-1}\}$ and a set of known minority-class phishing seed nodes $S \subseteq \bigcup_{i=1}^{t-1} V_i$, our goal is to generate a synthetic snapshot $\hat{G}_t$ that preserves realistic local structural patterns around phishing seeds, maintains temporal continuity with prior snapshots, and contains no ground-truth labels, ensuring compatibility with downstream training.

The augmented dataset $\mathcal{D}_{\text{aug}} = \{(G_t, Y_t)\} \cup \{(\hat{G}_t, \hat{Y}_t)\}$ is then used to train phishing detectors with improved recall on rare events.

Unlike generic graph generation tasks that prioritize global statistics, data augmentation for fraud detection demands high fidelity in local behavioral dynamics. Specifically, a useful generator should capture recurrent interaction patterns as well as the continuous evolution of node neighborhoods. Critically, the generation process must avoid producing isolated subgraphs disconnected from the broader network context and must faithfully simulate how local neighborhoods change over time. Failure to maintain contextual connectivity or to replicate realistic neighborhood dynamics results in synthetic graphs that lack the structural signatures of real phishing behaviors, ultimately limiting their effectiveness in improving detection performance.

B. Challenges

However, prevailing dynamic graph generation methods struggle to preserve such spatio-temporal regularities, leading

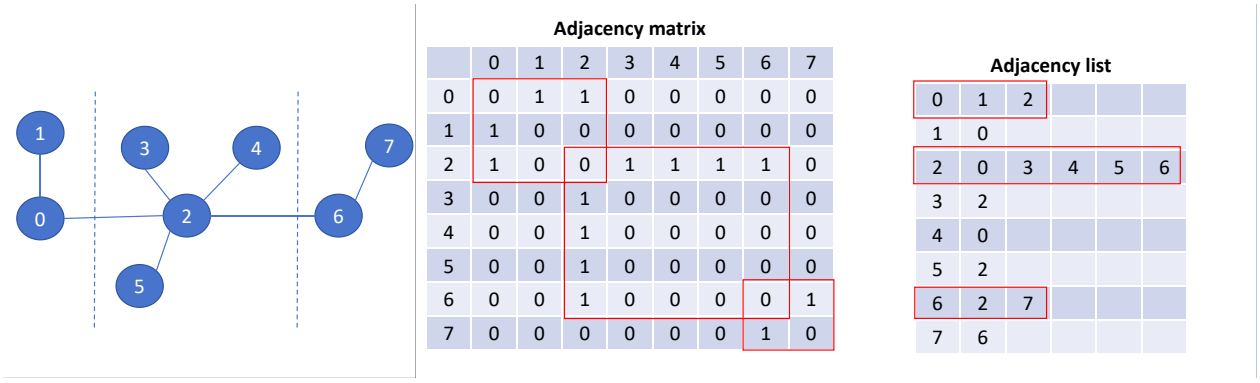


Fig. 2: (a)The connected dynamic graph sequence; (b) The concatenated adjacency matrix; (c) The adjacency list.

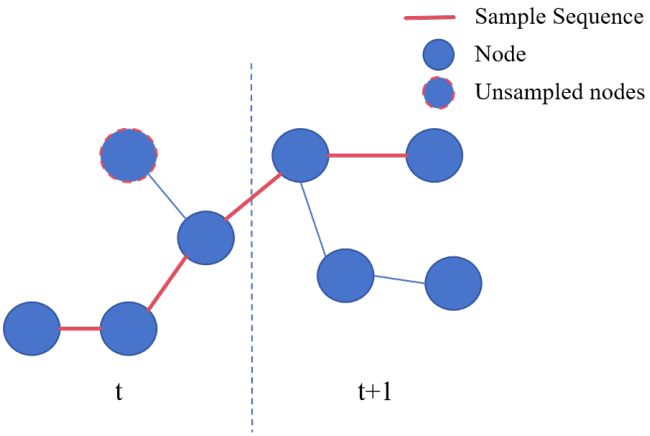


Fig. 3: Figure illustrates the structural evolution of a dynamic network across different snapshots, with the left portion representing the true graph structure of the previous snapshot $(t-1)$, and the right portion depicting the true graph structure of the subsequent snapshot (t). This visualization suggests that a random walk scheme spanning snapshots may lead to the fragmentation of temporal structural information. During training, when the algorithm predicts new nodes for the next snapshot, the model is limited to relying solely on the local dependencies from the previous snapshot, failing to fully capture the global temporal context.

to synthetic graphs that lack realism and utility for minority-class augmentation, as shown in Figure 3 and Figure 4. We identify three interrelated challenges:

- **Challenge 1: Limited temporal coherence in neighborhood representation.** Many approaches rely on biased random walks that implicitly prioritize transitions to nodes in recent snapshots. While this simplifies temporal modeling, it fragments node trajectories across time, undermining the ability to capture consistent, node-specific neighborhood evolution and yielding temporally inconsistent structural representations.
- **Challenge 2: Topological inconsistency in synthetic graph construction.** Existing methods often generate independent node sequences and assemble graphs via inter-sequence splicing or stitching. This decouples topology formation from temporal dynamics, frequently introducing artificial edges or breaking genuine connectivity, thereby compromising the topological fidelity of the generated graphs.
- **Challenge 3: Lack of behavior-aware node dynamics modeling.** Most frameworks treat nodes as passive entities and fail to model behavioral attributes—such as transaction frequency, interaction diversity, or role stability over time. Consequently, the synthesized graphs miss the nuanced local variations essential for augmenting minority classes in imbalanced fraud detection, where subtle behavioral cues drive discriminative performance.

Figure 2 illustrates three equivalent representations of a dynamic graph: (a) a sequence of temporally ordered snapshots, (b) a concatenated adjacency matrix across time, and (c) the corresponding adjacency list format. In the matrix view, each entry denotes the presence (1) or absence (0) of an edge at a specific time step, while the adjacency list explicitly enumerates neighbors per node per timestamp. As highlighted by the red boxes, core nodes—especially those involved in coordinated activities such as phishing—often exhibit persistent or gradually evolving local structural motifs across consecutive snapshots. These recurrent patterns are consistently captured in both representations and constitute critical behavioral signatures for anomaly detection.

To address these issues, we observe that the adjacency list representation naturally encodes the sequential evolution of node neighborhoods without requiring explicit graph transformation. Unlike the concatenated adjacency matrix, which becomes prohibitively large and sparse with increasing time steps, the adjacency list remains compact, scalable, and rich in fine-grained spatio-temporal detail. Therefore, we formulate dynamic graph generation as the autoregressive synthesis of adjacency lists conditioned on historical snapshots. This enables direct modeling of how neighborhoods expand, contract, or rewire over time, supporting high-fidelity simulation of local dynamics critical for downstream phishing detection.

Motivated by these insights, GRAPHFLOWGEN jointly learns spatio-temporal evolution in an end-to-end manner, avoiding intermediate graph transformations or adversarial

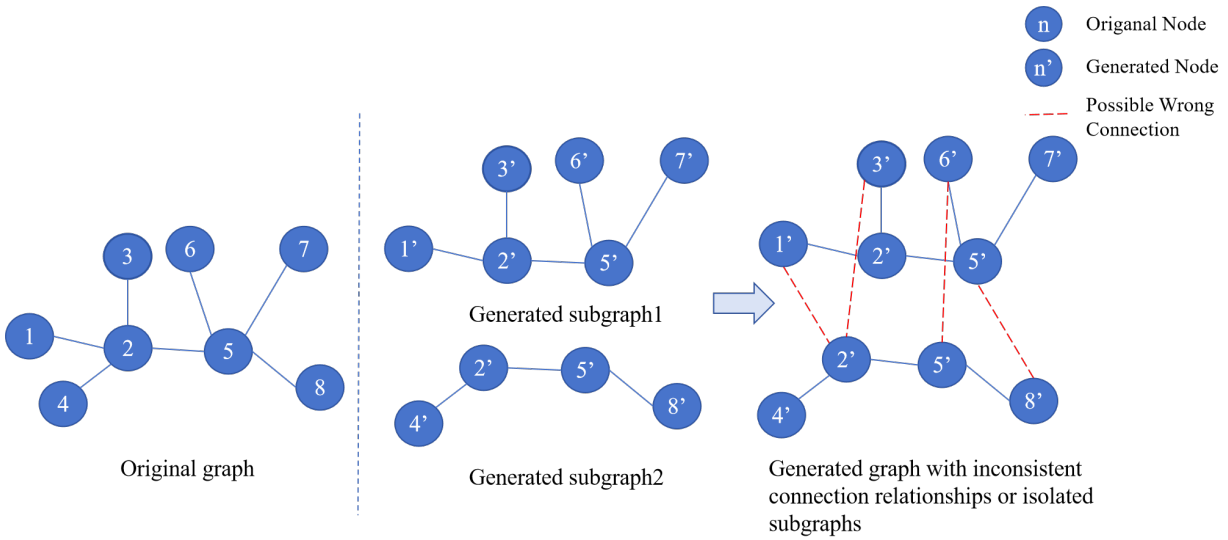


Fig. 4: Illustrations of the dynamic network's structural evolution, with the leftmost portion representing the original graph structure of the previous snapshot $(t-1)$; the middle section depicting a generated subgraph; and the right portion showing the predicted graph for the next snapshot (t). This visualization reveals two potential issues with the current generation approach: firstly, event-based graph generation methods have a notable probability of producing two independent subgraphs, as exemplified by the case where red dashed lines are removed; secondly, concatenation-based graph generation methods relying on conditional probabilities may result in an excessive number of erroneous links between subgraphs, such as red dashed lines connecting implausible node pairs.

training. The next section details our architecture.

IV. Our Method: GraphFlowGen

To address the challenges in modeling dynamic graphs for blockchain phishing detection, we propose GraphFlowGen, an end-to-end deep generative framework designed to capture both structural and temporal dynamics in transaction networks.

A. Overview

In this section, we provide an overview of our method as illustrated in Figure 5. GraphFlowGen is an end-to-end deep generative framework specifically designed to **address the severe class imbalance** in blockchain phishing detection by synthesizing realistic minority-class dynamic graphs. The model integrates three core components to jointly capture structural and temporal dynamics in Ethereum transaction networks, **directly tackling the fundamental challenges** of temporal coherence, topological fidelity, and behavior-aware dynamics.

- **Time Sequence Encoder**: To **ensure temporal coherence**, this module processes chronologically ordered global statistics from historical snapshots using a multi-layer Transformer architecture. By incorporating positional encodings and a hybrid mean-max pooling strategy, it effectively captures both long-term trends and transient bursts, **crucial for modeling the temporal signatures of anomalous phishing behaviors**. This prevents the generation of temporally inconsistent snapshots that could mislead downstream detectors.
- **Graph Structural Encoder**: To **preserve topological fidelity**, this component encodes the local topology of each graph snapshot by learning context-aware node representations through multi-head attention over adjacency structures. It **preserves fine-grained neighborhood evolution patterns**, enabling the model to reconstruct sparse yet meaningful motifs commonly associated with phishing accounts. This ensures that synthetic phishing subgraphs exhibit authentic structural characteristics, rather than being generated via heuristic rules that might miss critical topological anomalies.
- **Graph Generation Mechanism**: To **support behavior-aware dynamics**, this mechanism frames graph synthesis as a Markov Decision Process, autoregressively constructing adjacency lists conditioned on fused spatio-temporal embeddings. A reward function combining structural similarity and connectivity **encourages the generation of coherent, high-fidelity subgraphs** while minimizing isolated components or spurious edges. This **iterative RL-guided process** ensures that the synthetic graphs not only look realistic but also exhibit nuanced behavioral patterns essential for training effective phishing detectors.

By unifying temporal reasoning, structural attention, and policy-driven optimization, GraphFlowGen **generates synthetic dynamic graphs that faithfully replicate real-world phishing interaction patterns**, **thereby directly addressing the data scarcity issue** and enabling effective data augmentation for downstream anomaly detection tasks. This approach **overcomes the limitations of label-dependent or fixed-time-range methods**, making it particularly suitable for augmenting limited phishing samples in highly imbalanced blockchain environments.

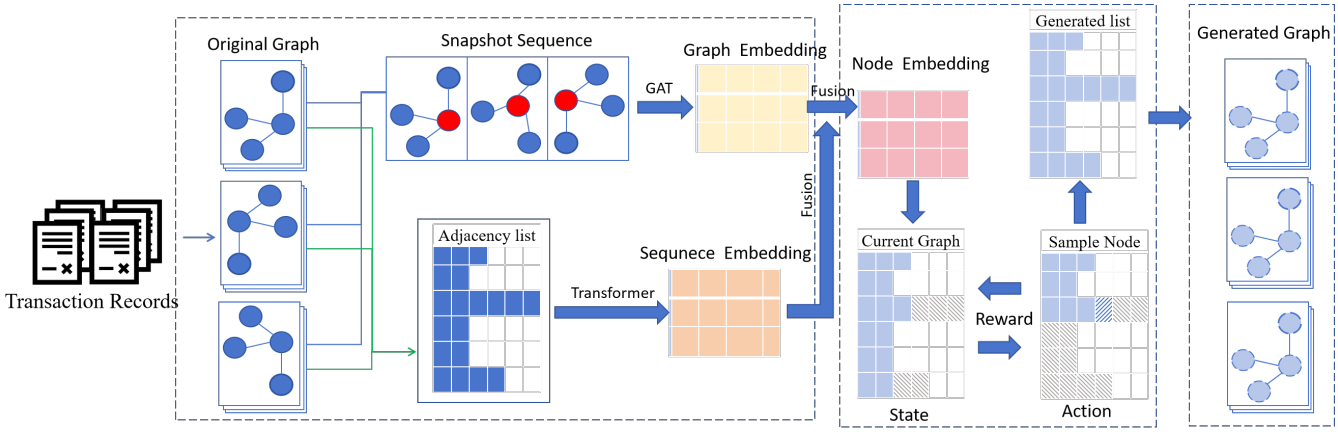


Fig. 5: Overview of GraphFlowGen. The framework integrates (i) a Transformer-based Time Sequence Encoder for temporal context, (ii) a GAT-based structural encoder for local neighborhood representation, and (iii) an RL-driven autoregressive generator that synthesizes realistic dynamic graphs via adjacency list prediction.

B. Time Sequence Encoder

To effectively capture long-term temporal dependencies in the evolution of dynamic networks, we design a Time Sequence Encoder based on the Transformer architecture. This module models the global statistical features of historical snapshots, learning complex contextual patterns in the time series to provide critical temporal prior information for subsequent graph structure generation. Specifically, the input to this module is a set of historical snapshot features arranged in chronological order, forming a time series matrix $\mathbf{X}_t \in \mathbb{R}^{L\times N}$, where L represents the sequence length and N denotes the feature dimension for each time step. These features, extracted and standardized during preprocessing, include normalized timestamp, number of nodes, number of transactions, and total transaction volume, ensuring training stability.

Before feeding the input into the Transformer layers, we augment it with positional encodings to inject information about the relative or absolute position of each time step in the sequence. This is essential because the self-attention mechanism itself is permutation-invariant and would otherwise be unable to distinguish the temporal order of snapshots. The positional encodings are added directly to the projected input embeddings:

$$\mathbf{H}^{(0)} = \mathbf{X}_t\mathbf{W}_e + \mathbf{b}_e + \mathbf{P}, \tag{1}$$

where $\mathbf{W}_e \in \mathbb{R}^{N\times D_h}$ and $\mathbf{b}_e$ are learnable parameters, D_h is the hidden dimension, and $\mathbf{P} \in \mathbb{R}^{L\times D_h}$ denotes the sinusoidal positional encoding matrix.

This representation is then processed by a multi-layer Transformer encoder. Each layer consists of a Multi-Head Self-Attention mechanism followed by a position-wise Feed-Forward Network, with residual connections and Layer Normalization applied throughout to stabilize training and mitigate gradient vanishing. The self-attention mechanism allows the model to dynamically attend to relevant time steps, regardless of their distance in the sequence, thereby capturing both short-term fluctuations and long-term trends.

The encoder outputs a contextualized temporal feature sequence $\mathbf{H}_t \in \mathbb{R}^{L\times D_h}$, where each time step's representation aggregates information from the entire input window. Rather than relying solely on average pooling-which may dilute critical transient signals like sudden surges associated with phishing activities-we adopt a hybrid aggregation strategy. Specifically, we compute the global temporal summary vector $\mathbf{h}_t \in \mathbb{R}^{D_h}$ by concatenating the mean-pooled and max-pooled representations:

$$\mathbf{h}_t = \mathrm{MLP}\left(\left[\frac{1}{L}\sum_{l=1}^{L}\mathbf{H}_t[l] \;\Big\|\; \max_{l=1,\dots,L}\mathbf{H}_t[l]\right]\right), \tag{2}$$

where $\|$ denotes concatenation and MLP is a lightweight two-layer perceptron that fuses the complementary statistics. This design preserves both the overall trend and salient peaks in the temporal dynamics, which is crucial for detecting rare but high-impact events such as phishing attacks.

This vector serves as a key input for subsequent feature fusion and prediction tasks. The choice of Transformer is motivated by its ability to model arbitrary-range temporal dependencies in parallel while adaptively focusing on informative moments in the sequence-properties particularly well-suited to the irregular and bursty nature of blockchain transaction data. Experimental results demonstrate that the Time Sequence Encoder significantly enhances the model's ability to predict network evolution trends, especially in scenarios involving abrupt structural changes or anomalous activity.

C. Graph Structural Encoder

To effectively model the topological structure and node interactions within each snapshot of a dynamic network, we employ a GAT as the graph structure encoder. This module leverages node features and connectivity patterns in the current snapshot to learn context-aware node representations, capturing local structural patterns and latent behavioral semantics. For a given snapshot t, the graph is defined by a node set $\mathcal{V}_t$, an edge set $\mathcal{E}_t \subseteq \mathcal{V}_t \times \mathcal{V}_t$, a node feature matrix $\mathbf{X}_g \in \mathbb{R}^{N\times D_n}$, and an adjacency matrix $\mathbf{A} \in \{0,1\}^{N\times N}$, where $N = |\mathcal{V}_t|$ denotes the number of nodes in the snapshot, and $D_n = 4$ represents the feature dimension per node. Initially, the raw

node features are transformed into a high-dimensional latent space via a linear embedding with a ReLU activation:

$$\mathbf{H}^{(0)} = \text{ReLU}(\mathbf{X}_g\mathbf{W}_g + \mathbf{b}_g), \tag{3}$$

where $\mathbf{W}_g \in \mathbb{R}^{D_n \times D_h}$ and $\mathbf{b}_g \in \mathbb{R}^{D_h}$ are learnable parameters, and $\mathbf{H}^{(0)} \in \mathbb{R}^{N \times D_h}$ represents the initial node embeddings, with D_h as the hidden dimension. Subsequently, a multi-layer, multi-head attention mechanism is employed for message passing. For the k-th attention head, the attention coefficient for node i with respect to its neighbor j is computed as:

$$\alpha_{ij}^{(k)} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^{(k)\top}[\mathbf{W}_k\mathbf{h}_i || \mathbf{W}_k\mathbf{h}_j]\right)\right)}{\sum_{j' \in \mathcal{N}(i)} \exp\left(\text{LeakyReLU}\left(\mathbf{a}^{(k)\top}[\mathbf{W}_k\mathbf{h}_i || \mathbf{W}_k\mathbf{h}_{j'}]\right)\right)} \tag{4}$$

where $\mathbf{W}_k \in \mathbb{R}^{D_h \times D'_h}$ is a head-specific linear transformation matrix, $\mathbf{a}^{(k)} \in \mathbb{R}^{2D'_h}$ is a learnable attention vector, $||$ denotes vector concatenation, and $\mathcal{N}(i)$ represents the neighborhood of node i. The output of each attention head is computed as:

$$\mathbf{h}_i'^{(k)} = \sum_{j \in \mathcal{N}(i)} \alpha_{ij}^{(k)} \mathbf{W}_k \mathbf{h}_j. \tag{5}$$

The outputs from all K attention heads are combined, either by concatenation (for intermediate layers) or averaging (for the final layer), to produce the updated node representation:

$$\mathbf{h}_i' = \begin{cases} \|_{k=1}^{K} \mathbf{h}_i'^{(k)}, & \text{if not last layer,} \\ \frac{1}{K}\sum_{k=1}^{K} \mathbf{h}_i'^{(k)}, & \text{if last layer,} \end{cases} \tag{6}$$

where K is the number of attention heads. After L_g layers of GAT encoding, the final output is a context-enhanced graph representation $\mathbf{H}_g \in \mathbb{R}^{N \times D_h}$, which integrates both the node's own features and structural information from its multi-hop neighborhood. This representation is well-suited for subsequent feature fusion and dynamic evolution prediction tasks. This design leverages the expressive power of attention mechanisms to model heterogeneous interactions within the graph structure, enabling the model to automatically focus on critical neighboring nodes. It demonstrates strong representational capabilities for tasks such as phishing account identification and anomalous fund flow detection. Additionally, the module supports variable-length node inputs, accommodating changes in network scale across different snapshots, thus ensuring excellent scalability.

D. Graph Generation Mechanism

To refine the graph generation process and ensure high-fidelity outputs that preserve both structural and temporal properties, we integrate a reinforcement learning (RL) module into GraphFlowGen. This module models the generation of the next snapshot $\hat{G}_t$ as a Markov Decision Process (MDP), where the agent iteratively builds an adjacency list representation of the graph. Unlike traditional approaches that rely on random walks or subgraph splicing, our RL-based mechanism autoregressively constructs the graph conditioned on the fused spatio-temporal representation H'_f, addressing key limitations in capturing local dynamics and temporal continuity.

The MDP is defined as follows:

- **State** s_k: The current partial adjacency list of $\hat{G}_t$, represented as a dictionary where keys are node IDs and values are lists of connected neighbors. Additionally, the state includes the fused embedding H'_f concatenated with a GNN-encoded summary of the partial graph using a lightweight GAT layer for efficiency, updated after each action to capture dependencies.
- **Action** a_k: A tuple $(\text{expand\_node}, \text{is\_new}, \text{neighbor\_id})$, where expand_node $\in \{0, \ldots, N_{\text{current}} - 1\}$ selects an existing node from the partial graph to expand, is_new $\in \{0, 1\}$ indicates whether to add a new node or connect to an existing one, and neighbor_id $\in \{0, \ldots, N_{\text{current}} - 1\}$ specifies the neighbor if is_new $= 0$. If is_new $= 1$, neighbor_id is ignored and a new ID is auto-assigned.
- **Transition**: Deterministic; the partial adjacency list is updated by adding the edge expand_node $\rightarrow$ neighbor_id or introducing a new node with that edge.
- **Reward** r_k: A dense, multi-component signal to guide learning and mitigate sparse feedback:

$$R_{\text{sim}} = \cos(f(\hat{G}_t), f(G_{t-1})), \tag{7}$$

$$R_{\text{conn}} = \log\left(1 + \frac{|\text{largest\_component}(\hat{G}_t)|}{N_{\text{target}}}\right), \tag{8}$$

where R_{sim} measures structural similarity between $\hat{G}_t$ and G_{t-1}, it employs Cosine Similarity, measuring directional alignment rather than Euclidean distance. This effectively allows the model to generate significant scale bursts without penalty, provided the underlying topological pattern remains consistent, thus preventing over-smoothing. R_{conn} provides a connectivity bonus. $f(\cdot)$ denotes a graph representation learner used to map a graph snapshot to a latent embedding vector. Specifically, $f(G)$ is implemented as a lightweight GAT layer followed by a global mean pooling operation:

$$f(G) = \text{MeanPooling}(\text{GAT}(G)) \tag{9}$$

This allows R_{sim} to quantitatively measure the cosine similarity between the global structural representations of the generated graph $\hat{G}_t$ and the target predecessor G_{t-1}. The reward function is given by:

$$r_k = \lambda_{\text{sim}} R_{\text{sim}} + \lambda_{\text{conn}} R_{\text{conn}} + \delta, \tag{10}$$

where $\lambda_{\text{sim}} = 0.7$, $\lambda_{\text{conn}} = 0.3$, and δ is a small intermediate penalty for overly long trajectories. These hyperparameters were determined empirically based on validation set performance. We assign a dominant weight to λ_{sim} (0.7) to prioritize temporal coherence, while setting λ_{conn} (0.3) as an auxiliary regularizer to ensure graph connectivity.

The policy $\pi_\theta(a_k|s_k)$ is parameterized by a neural network that takes the state embedding and outputs action probabilities (softmax over the discrete space). To address the high variance inherent in vanilla REINFORCE due to episode-end rewards, we employ REINFORCE with a learned baseline: an input-dependent value function $V_\phi(s_k)$ subtracts from the return

$G_k = \sum_{i=k}^{K} \gamma^{i-k} r_i$ to form advantages $A_k = G_k - V_\phi(s_k)$. This technique stabilizes training by centering updates around zero-mean advantages.

Training optimizes the policy via policy gradients:

$$L_{\text{RL}} = -\,\mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_k A_k \log \pi_\theta(a_k|s_k)\right] + \beta\mathbb{E}\left[(G_k - V_\phi(s_k))^2\right], \tag{11}$$

where the second term represents the value function loss, and $\beta = 0.5$ balances policy and value learning. The full fine-tuning loss combines this with supervised terms:

$$L_{\text{finetune}} = L_{\text{RL}} + \lambda_{\text{node}} L_{\text{node}} + \lambda_{\text{sum}} L_{\text{node\_sum}}, \tag{12}$$

where $L_{\text{node}} = \text{MSE}(N_{\text{target}}, |V_t|)$ predicts target nodes, and $L_{\text{node\_sum}}$ aggregates cumulative counts. While R_{sim} ensures structural fidelity, the supervised loss L_{node} drives dynamic evolution. By strictly enforcing the target graph size predicted by the Time Sequence Encoder, L_{node} explicitly overrides the inertia from the previous snapshot to simulate transient bursts.

Graph generation starts from an empty adjacency list with a single core node sampled from high-degree nodes in G_{t-1} to preserve continuity. During inference, greedy temperature-scaled sampling is used for diversity.

In summary, GraphFlowGen provides a comprehensive end-to-end solution for generating realistic dynamic graphs in blockchain transaction networks, addressing key challenges in temporal coherence, topological consistency, and behavior-aware dynamics through its integrated components: a Transformer-based Time Sequence Encoder for capturing long-term dependencies, a GAT-based Graph Structural Encoder for modeling local topologies, and an RL-driven autoregressive generation mechanism for high-fidelity synthesis. This framework not only mitigates data imbalance in phishing detection but also ensures scalability and fidelity in downstream tasks. For a concise algorithmic overview, we present the pseudocode of GraphFlowGen in Algorithm 1.

Algorithm 1 GraphFlowGen

Require: Historical snapshots $\{G_1, \ldots, G_{t-1}\}$, phishing seed nodes S, time-series features X_t, node features X_g
Ensure: Synthetic snapshot $\hat{G}_t = (\hat{V}_t, \hat{E}_t, \hat{X}_t)$

```
// Step 1: Temporal Encoding
Extract global temporal features from {G_i}_{i=1}^{t-1} into X_t
Compute temporal context vector h_t using the Time Sequence Encoder (Eq. 2)
// Step 2: Structural Encoding
Encode local topology of G_{t-1} using GAT to obtain node embeddings H_g (Eq. 3–6)
// Step 3: Feature Fusion
Fuse h_t and H_g into joint spatio-temporal embedding H_f
// Step 4: RL-based Autoregressive Generation
Initialize partial graph Ĝ_t with a core phishing seed node from S
for each generation step until convergence do
    Sample next action (add node or edge) via policy π_θ(· | s)
    Update state s (current adjacency list)
    Compute reward r = λ_sim R_sim + λ_conn R_conn (Eq. 7–10)
    Update policy parameters θ using REINFORCE with baseline (Eq. 11)
end for
return Ĝ_t
```

E. Complexity Analysis

The computational complexity corresponds to the three primary modules. The Transformer-based Time Sequence Encoder has a complexity of $O(L^2 \cdot d)$, where L is the sequence length and d is the hidden dimension. Given that L represents the number of historical snapshots (typically small), this cost is low. The GAT-based Structural Encoder scales as $O(|V|+|E|)$, which is linear with respect to the graph size, making it suitable for sparse blockchain networks. The RL generation process involves autoregressive decision-making steps proportional to the size of the synthesized subgraph, ensuring the overall framework remains computationally feasible.

V. Experiment Results and Analysis

A. Experimental Setup

We evaluate GraphFlowGen on three real-world dynamic graph datasets spanning distinct domains, comparing it against representative state-of-the-art baselines. All experiments are implemented in Python with PyTorch and run on a machine equipped with an NVIDIA RTX 3060 Laptop (6GB VRAM), Intel Core i7-11800H CPU, and 16GB RAM.

The datasets are:

- **Ethereum** [14]: A blockchain transaction subgraph for phishing detection, containing 1,660 phishing and 14,940 benign accounts (10% minority class). Used to assess synthetic data augmentation for anomaly detection.
- **Stack Overflow (SO)** [34]: A sparse, power-law user collaboration network from Q&A interactions, ideal for evaluating local motif preservation.
- **Wikipedia** [35]: A denser editing network with bursty, hierarchical interaction patterns, testing the model's ability to capture evolving collaborative structures.

To construct the discrete-time dynamic graph sequence, continuous interaction data is discretized into snapshots using a fixed time window. In our experiments, we set the time interval Δt to 24 hours, aggregating all transactions within each day into a single snapshot G_t. Each dataset is chronologically split into 70% training, 10% validation, and 20% test sets. For Ethereum, GraphFlowGen is trained only on snapshots containing at least one phishing node, and synthetic graphs are generated to alleviate class imbalance.

We compare against four baselines:

- **TAGEN** [9]: Generates graphs via temporal random walks and local perturbations.
- **TIGGER-I/T** [10]: Inductive (I) and transductive (T) variants modeling continuous-time node embeddings.
- **DGGEN** [27]: A probabilistic point-process model that jointly generates node pairs, timestamps, and edges.

TABLE I: Static structural statistics on Ethereum, SO, and Wikipedia datasets.

Dataset	Method	Mean degree	Claw count	Wedge count	Power law	LCC
Ethereum	Original	1.5453	102.7929	21.7107	2.1453	6.2036
	GraphFlowGen	**1.5894**	**106.42**	**22.83**	**2.1098**	**7.2036**
	TAGEN	1.7841	220.3	38.6	2.4873	68.5
	TIGGER-I	1.1495	11.56	7.37	2.9253	3.5475
	TIGGER-T	1.1595	11.45	7.40	3.4765	3.5200
	DGGEN	1.0589	0.38	6.50	2.6554	2.0000
SO	Original	1.9004	2328.17	260.61	2.1969	140.5062
	GraphFlowGen	**1.8631**	**2184.3**	**241.8**	**2.2234**	**163.1553**
	TAGEN	2.6842	4872.1	612.4	2.6871	1250.3
	TIGGER-I	1.5123	892.6	104.3	2.7345	80.3478
	TIGGER-T	1.4891	874.2	98.7	2.8123	79.7249
	DGGEN	1.1087	124.5	18.2	3.1568	45.2983
Wikipedia	Original	1.7645	1260.55	146.22	2.1969	76.0748
	GraphFlowGen	**1.2407**	**1250.96**	**140.15**	**2.1972**	**88.3378**
	TAGEN	2.3218	3421.8	489.7	2.9124	782.6
	TIGGER-I	1.4231	487.2	62.8	2.6451	43.5030
	TIGGER-T	1.3987	472.9	59.4	2.7893	43.1658
	DGGEN	0.9876	38.7	8.1	3.4237	24.5260

These methods cover major paradigms in dynamic graph generation—random-walk-based, embedding-based, and point-process-based—enabling a comprehensive comparison.

All baselines are evaluated using their official codebases[1] with default hyperparameters; only the random seed and GPU device are adjusted for compatibility.

GraphFlowGen uses PyTorch 2.0 and Python 3.10. Its architecture includes a 4-layer Transformer encoder (8 heads, 256-dim hidden state) and a 2-layer GAT (4 heads per layer). Training uses Adam (lr=0.001, batch size=32) for 100 epochs. Graph statistics are computed with NetworkX.

To measure structural fidelity, we report four standard metrics on the aggregated test-period graph:

- **Mean degree**: average node degree, indicating overall density;
- **Claw count**: number of $K_{1,3}$ star motifs, common in hub-centric behaviors;
- **Wedge count**: number of length-2 paths, reflecting local clustering;
- **Power-law exponent**: tail exponent of the degree distribution, capturing scale-free structure;
- **LCC**: average size of the largest connected component per snapshot, reflecting preservation of coherent local connectivity.

For each metric, we report the absolute value on the synthetic graph and its relative deviation from the original, enabling quantitative assessment of realism.

B. Static Structural Fidelity

To comprehensively evaluate the structural fidelity of generated dynamic graphs, we report four representative statistics—mean degree, claw count, wedge count, and power-law exponent—across three real-world benchmarks: Ethereum, Stack Overflow, and Wikipedia. These datasets exhibit diverse interaction patterns, sparsity levels, and temporal dynamics, providing a meaningful setting for assessing generalization.

As shown in Table I, GraphFlowGen achieves consistently low relative errors across all three domains. On Ethereum and SO—both sparse networks with heavy-tailed degree distributions—it closely matches the original graphs in terms of local motifs and global topology. On Wikipedia, while the mean degree error is relatively higher (29.%), likely due to the dataset's dense and bursty editing behavior, GraphFlowGen still preserves higher-order structures remarkably well, with claw count and power-law exponent deviating by less than 1%.

This consistent performance can be attributed to several design choices that aim to better align the generative process with common structural patterns in dynamic networks. In particular, GraphFlowGen models node neighborhoods autoregressively via adjacency lists, which naturally supports the generation of star-like structures often observed in phishing or expert-driven interactions. This contrasts with approaches such as TAGEN, which relies on local random walks without explicit control over node reuse across time steps. As a result, TAGEN may generate an excess of leaf nodes to fit short-term interaction frequencies, leading to inflated claw counts and denser-than-expected subgraphs.

[1]TG: https://github.com/davidchouzdw/TagGen; TIGGER: https://github.com/data-iitd/tigger; DGGEN: https://github.com/ryienh/DGGen

Similarly, DGGEN adopts a point-process framework that samples node pairs globally and generates edges independently based on continuous-time embeddings. While this formulation offers flexibility in modeling event timestamps, it does not explicitly encode dependencies among edges sharing a common center node. Without constraints to encourage local connectivity or differentiate hub versus peripheral roles, the resulting graphs tend to fragment into isolated components, as reflected in the substantially reduced claw counts across all datasets.

TIGGER variants, though capable of capturing temporal evolution through continuous-time embeddings, appear to prioritize trajectory-level likelihood over motif-level consistency. Consequently, they reproduce average degrees reasonably well but underrepresent multi-hop structures like wedges and claws, suggesting limited propagation of neighborhood information across time slices.

In contrast, GraphFlowGen integrates temporal context via a Transformer-based memory mechanism and spatial dependencies through a GAT-style attention module, allowing it to jointly model recurrent interaction patterns and local clustering tendencies. Additionally, the use of reinforcement learning with structural similarity rewards provides an auxiliary signal that encourages the preservation of key topological properties during training—complementing the primary likelihood objective.

Overall, these design elements enable GraphFlowGen to generate dynamic graphs that more closely resemble the structural characteristics of real-world networks across different domains, including those with rare but structurally significant behaviors such as phishing campaigns.

C. Dynamic Temporal Fidelity

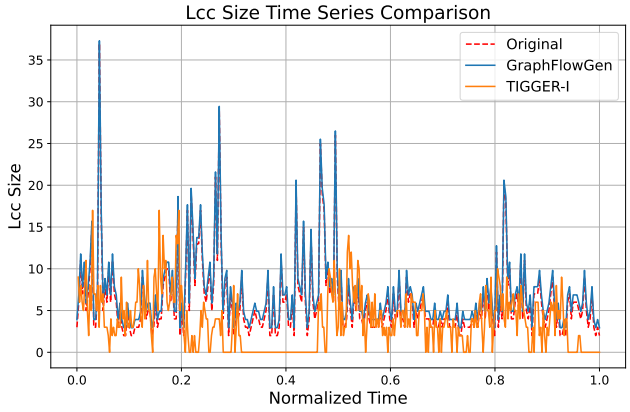
To assess the capacity of GraphFlowGen to model the underlying mechanisms of structural evolution in dynamic graphs, we conduct a comparative analysis with TIGGER. Rather than comparing absolute metric values, we emphasize the alignment of temporal trajectories, as these better reflect an algorithm's understanding of network dynamics.

We examine three structurally informative time-series metrics: wedge count, LCC size, and claw count. These capture local clustering, global connectivity, and hub-centric motifs, respectively, offering a multi-scale view of graph evolution.

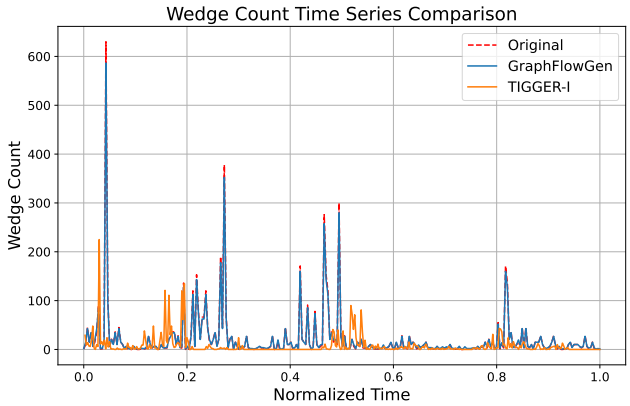
In the wedge count trajectory, the original Ethereum data exhibits pronounced non-stationary behavior, characterized by sharp, pulse-like bursts during early phases followed by extended periods of low activity. GraphFlowGen closely reproduces this pattern, aligning well with both the timing of peaks and the subsequent decay rates, while preserving the quiescent baseline in later stages. In contrast, TIGGER generates a smoother overall trend but shows noticeable lag at critical transition points—particularly during abrupt structural surges—suggesting limited sensitivity to rapid, transient changes.

For LCC size, the ground-truth sequence follows a distinct macro-evolutionary logic: rapid formation of a dominant component in the initial phase, relative stability during mid-sequence, and gradual fragmentation toward the end as interactions become sparser. GraphFlowGen effectively mirrors this three-stage progression. Interestingly, TIGGER also demonstrates reasonable performance on this metric: although its trajectory is somewhat dampened in amplitude, it maintains correct phase alignment and avoids spurious connectivity, indicating robustness in modeling large-scale topological stability.

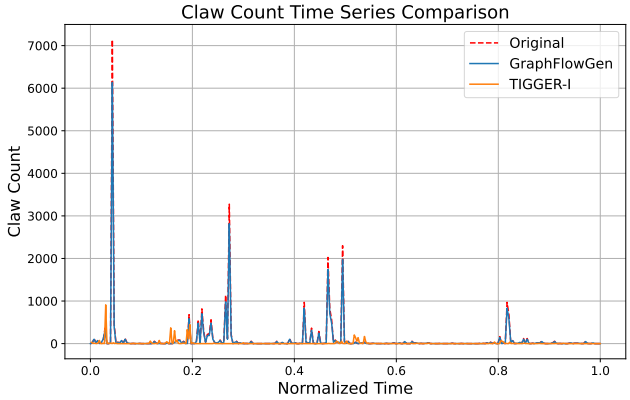
Finally, in the claw count—which reflects sparse, bursty appearances of star-shaped motifs centered around influential nodes—the original data displays irregular, non-periodic



(a) lcc_size evolution



(b) wedge_count evolution



(c) claw_count evolution

Fig. 6: Time-series trajectories of key structural metrics on the Ethereum dataset.

TABLE II: Ablation study on Ethereum.

Method	Avg. Loss ↓	Parameters	Inference
GraphFlowGen (full)	**4.9123**	371204	28.4
w/o Transformer	5.3372	293676	24.1
w/o GAT	5.3141	77528	19.7
w/o Fusion	5.3955	207752	22.3

spikes of varying magnitudes. GraphFlowGen successfully captures both the timing and heterogeneity of these rare events, whereas TIGGER tends to either smooth out or miss minor peaks, likely due to its reliance on continuous-time embeddings that favor regularity over sparsity.

In summary, GraphFlowGen exhibits stronger expressiveness in modeling fine-grained, multi-scale structural dynamics, particularly in capturing transient bursts and non-stationary transitions.

D. Ablation Study

To investigate the contribution of each component in GraphFlowGen, we conduct systematic ablation experiments on the Ethereum dataset. Specifically, we evaluate three variants: (i) **GraphFlowGen w/o Transformer**, where the Transformer encoder is replaced by a simple MLP with the same hidden dimension; (ii) **GraphFlowGen w/o GAT**, where graph attention layers are removed and only temporal features are processed; and (iii) **GraphFlowGen w/o Fusion**, where the late-fusion module that integrates temporal and structural embeddings is disabled, and only concatenated features are fed to the decoder. All variants share the same reinforcement learning reward and training schedule as the full model.

As reported in Table II, removing any single component leads to a noticeable performance degradation, underscoring their complementary roles. The Transformer encoder contributes the most significant improvement: its absence increases the average loss by 8.65% (from 4.9123 to 5.3372), despite reducing the parameter count by 21%. This confirms its critical role in capturing long-range temporal dependencies across hundreds of snapshots, which is essential for modeling the recurrent interaction patterns observed in phishing campaigns.

The graph structural encoder also proves indispensable, achieving a 8.18% loss increase when removed (5.3141). Although this variant has the fewest parameters (77,528), its substantial performance drop highlights the necessity of explicit neighborhood attention for reconstructing sparse, star-shaped fraud topologies. The fusion mechanism, while contributing the smallest individual gain (loss rises to 5.3955, a 9.84% relative increase), remains crucial for seamless integration of temporal and structural cues; without it, the decoder receives mismatched representations that impair autoregressive stability.

These results collectively validate that the superior fidelity of GraphFlowGen arises from the synergistic interplay of its three core components rather than any single module. The reinforcement learning reward, shared across all variants, further amplifies these gains by directly optimizing structural similarity throughout training.

TABLE III: Phishing detection performance on Ethereum test set using GAT+LSTM classifier. Synthetic phishing snapshots are added for augmentation.

Method	**Precision**	**Recall**	**F1-score**	**Accuracy**
Original only	0.83	0.90	0.86	0.8406
+ GraphFlowGen	**0.88**	**0.93**	**0.90**	**0.8806**
+ TIGGER-I	0.85	0.89	0.87	0.863
+ TIGGER-T	0.86	0.88	0.87	0.866

E. Downstream Phishing Detection Performance

In real-world blockchain networks, anomaly detection is often compromised by extreme class imbalance—phishing accounts, though highly consequential, constitute only a small fraction of all entities. To evaluate whether synthetic graph generation can enhance detection performance under such conditions, we conduct a case study on the Ethereum transaction network, where labeled phishing nodes are treated as anomalies.

The dataset contains 16,600 nodes in total, with 1,660 (10%) identified as phishing accounts. We use this labeled set as our evaluation benchmark and apply data augmentation solely during training. Specifically, we employ GraphFlowGen to generate realistic temporal neighborhoods that capture characteristic phishing patterns and incorporate these synthetic samples into the training data. Crucially, the test set remains unchanged and consists only of original, real-world nodes, ensuring a fair assessment of generalization capability.

Node representations are derived from the encoder of the trained generative model and fed into a logistic regression classifier. We report four standard metrics: Precision, Recall, F1-score, and Accuracy, with results summarized in Table III.

As shown in Table III, augmenting the training data with graphs generated by GraphFlowGen yields consistent improvements across all metrics. Our method achieves a Precision of 0.88, Recall of 0.93, and an F1-score of 0.90, outperforming the "Original only" baseline by +0.04 in F1-score and +0.04 in Accuracy. In comparison, augmentations based on TIGGER-I and TIGGER-T provide only marginal gains, highlighting the importance of preserving both structural fidelity and temporal dynamics in synthetic graphs for downstream anomaly detection.

VI. Conclusion

In this paper, we propose GraphFlowGen, an end-to-end deep generative framework that successfully overcomes the limitations of existing dynamic graph models for blockchain phishing detection. By jointly extracting structural-temporal contexts in preprocessing, fusing Transformer encoders with Graph Attention Networks to capture recurrent patterns, and employing reinforcement learning for optimization, GraphFlowGen generates high-fidelity graphs with superior structural similarity, temporal continuity, and degree distribution

alignment. Experiments on real Ethereum datasets show significant improvements in simulating rare phishing events, effectively addressing severe data imbalance and enhancing downstream fraud detection.

Future work will extend GraphFlowGen to incorporate multi-modal features and scale to larger heterogeneous graphs, further advancing generative modeling for proactive security in decentralized networks.

REFERENCES

[1] A. Urquhart, "Under the hood of the ethereum blockchain," *Finance Research Letters*, vol. 47, p. 102628, 2022.

[2] Z. Liu, X. Zhang, L. Lao, G. Li, and B. Xiao, "Dbe-voting: A privacy-preserving and auditable blockchain-based e-voting system," in *Proceedings of IEEE International Conference on Communications (ICC)*, 2023, pp. 6571–6577.

[3] Z. Liu, X. Zhang, G. Li, H. Cui, J. Wang, and B. Xiao, "A secure and reliable blockchain-based audit log system," in *ICC 2024-IEEE International Conference on Communications*. IEEE, 2024, pp. 2010–2015.

[4] J. Wang, P. Chen, X. Xu, J. Wu, M. Shen, Q. Xuan, and X. Yang, "Tsgn: Transaction subgraph networks assisting phishing detection in ethereum," *IEEE Transactions on Dependable and Secure Computing*, vol. 22, no. 5, pp. 4861–4876, 2025.

[5] E. Jung, M. Le Tilly, A. Gehani, and Y. Ge, "Data mining-based ethereum fraud detection," in *2019 IEEE International Conference on Blockchain (Blockchain)*. Atlanta, GA, USA: IEEE, 2019, pp. 266–273.

[6] L. Chen, J. Peng, Y. Liu, J. Li, F. Xie, and Z. Zheng, "Phishing scams detection in ethereum transaction network," *ACM Transactions on Internet Technology (TOIT)*, vol. 21, no. 1, pp. 1–16, 2020.

[7] H. Sui, J. Zhang, B. Chen, D. Wu, X. Sun, and S. Palaiahnakote, "Epad: Ethereum phishing scam detection via graph contrastive learning," *Expert Systems with Applications*, p. 128227, 2025.

[8] H. Tang, C. Wang, and H. Zhu, "Enhancing online transaction fraud detection via heterogeneous source models," *IEEE Transactions on Dependable and Secure Computing*, 2025.

[9] D. Zhou, L. Zheng, J. Han, and J. He, "A data-driven graph generative model for temporal interaction networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Virtual Event, CA, USA: ACM, 2020, pp. 401–411.

[10] S. Gupta, S. Manchanda, S. Bedathur, and S. Ranu, "Tigger: Scalable generative modelling for temporal interaction graphs," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 6. Virtual Event: AAAI Press, 2022, pp. 6819–6828.

[11] W. Chen, Z. Zheng, J. Cui, E. Ngai, P. Zheng, and Y. Zhou, "Detecting ponzi schemes on ethereum: Towards healthier blockchain technology," in *Proceedings of the 2018 World Wide Web Conference*. Lyon, France: International World Wide Web Conferences Steering Committee, 2018, pp. 1409–1418.

[12] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, 2014, pp. 701–710.

[13] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Francisco, CA, USA: ACM, 2016, pp. 855–864.

[14] J. Wu, Q. Yuan, D. Lin, W. You, W. Chen, C. Chen, and Z. Zheng, "Who are the phishers? phishing scam detection on ethereum via network embedding," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 2, pp. 1156–1166, 2020.

[15] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems 30*, Long Beach, CA, USA, 2017, pp. 1024–1034.

[16] S. N. Mighan, J. Mišić, V. B. Mišić, and X. Chang, "An in-depth look at forking-based attacks in ethereum with pow consensus," *IEEE Transactions on Network and Service Management*, vol. 21, no. 1, pp. 507–516, 2023.

[17] G. H. Nguyen, J. B. Lee, R. A. Rossi, N. K. Ahmed, E. Koh, and S. Kim, "Dynamic network embeddings: From random walks to temporal random walks," in *2018 IEEE International Conference on Big Data (Big Data)*. Seattle, WA, USA: IEEE, 2018, pp. 1085–1092.

[18] S. M. Kazemi, R. Goel, K. Jain, I. Kobyzev, A. Sethi, P. Forsyth, and P. Poupart, "Representation learning for dynamic graphs: A survey," *Journal of Machine Learning Research*, vol. 21, no. 70, pp. 1–73, 2020.

[19] W. Ju, Z. Fang, Y. Gu, Z. Liu, Q. Long, Z. Qiao, Y. Qin, J. Shen, F. Sun, Z. Xiao *et al.*, "A comprehensive survey on deep graph representation learning," *Neural Networks*, vol. 173, p. 106207, 2024.

[20] J. You, R. Ying, X. Ren, W. Hamilton, and J. Leskovec, "Graphrnn: Generating realistic graphs with deep auto-regressive models," in *International Conference on Machine Learning*. Stockholm, Sweden: PMLR, 2018, pp. 5708–5717.

[21] G. H. Nguyen, J. B. Lee, R. A. Rossi, N. K. Ahmed, E. Koh, and S. Kim, "Continuous-time dynamic network embeddings," in *Companion Proceedings of the The Web Conference 2018*. Lyon, France: International World Wide Web Conferences Steering Committee, 2018, pp. 969–976.

[22] H. Wang, J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, X. Xie, and M. Guo, "Graphgan: Graph representation learning with generative adversarial nets," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1. New Orleans, LA, USA: AAAI Press, 2018, pp. 2508–2515.

[23] Y. Xiong, Y. Zhang, H. Fu, W. Wang, Y. Zhu, and P. S. Yu, "Dyngraphgan: Dynamic graph embedding via generative adversarial networks," in *International Conference on Database Systems for Advanced Applications*. Chiang Mai, Thailand: Springer, 2019, pp. 536–552.

[24] K. Lei, M. Qin, B. Bai, G. Zhang, and M. Yang, "Gcn-gan: A non-linear temporal link prediction model for weighted dynamic networks," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*. Paris, France: IEEE, 2019, pp. 388–396.

[25] H. Huang, L. Sun, B. Du, Y. Fu, and W. Lv, "Graphgdp: Generative diffusion processes for permutation invariant graph generation," in *2022 IEEE International Conference on Data Mining (ICDM)*. Orlando, FL, USA: IEEE, 2022, pp. 201–210.

[26] Y. Xu and C. Ma, "Modiff-graph generation with motif-aware diffusion model," in *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. Toronto, ON, Canada: ACM, 2025, pp. 3437–3448.

[27] R. Hosseini, F. Simini, V. Vishwanath, and H. Hoffmann, "A deep probabilistic framework for continuous time dynamic graph generation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 16. Philadelphia, PA, USA: AAAI Press, 2025, pp. 17 249–17 257.

[28] E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, and M. Bronstein, "Temporal graph networks for deep learning on dynamic graphs," *arXiv preprint arXiv:2006.10637*, 2020.

[29] R. Trivedi, M. Farajtabar, P. Biswal, and H. Zha, "Dyrep: Learning representations over dynamic graphs," in *International Conference on Learning Representations*, New Orleans, LA, USA, 2019.

[30] D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, and K. Achan, "Inductive representation learning on temporal graphs," *arXiv preprint arXiv:2002.07962*, 2020.

[31] Y. Wang, Y.-Y. Chang, Y. Liu, J. Leskovec, and P. Li, "Inductive representation learning in temporal networks via causal anonymous walks," *arXiv preprint arXiv:2101.05974*, 2021.

[32] P. Veličkovi'c, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *6th International Conference on Learning Representations, ICLR 2018*, Vancouver, BC, Canada, 2018.

[33] F. Poursafaei, S. Huang, K. Pelrine, and R. Rabbany, "Towards better evaluation for dynamic link prediction," in *Advances in Neural Information Processing Systems 35*, New Orleans, LA, USA, 2022, pp. 32 928–32 941.

[34] D. Zhou, J. He, H. Yang, and W. Fan, "Sparc: Self-paced network representation for few-shot rare category characterization," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2807–2816.

[35] S. Kumar, X. Zhang, and J. Leskovec, "Predicting dynamic embedding trajectory in temporal interaction networks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Anchorage, AK, USA: ACM, 2019, pp. 1269–1278.

Siyi Xiao received his M.S degree in Computer Science and Technology from Dalian Jiaotong University in Dalian, China in 2022. He is now pursuing his PhD in software engineering at Yangzhou University. His research interests include blockchain technology, intelligent optimization algorithms and high performance computing.

Lejun Zhang received his M.S. degree in computer science and technology in Harbin Institute of Technology and the Ph.D. degrees in computer science and technology at Harbin Engineering University, where he was an professor at Guangzhou University. His research interests include computer network, blockchain and information security.

Xinwei Zhang received the M.Eng degree in computer technology from Southeast University, Nanjing, China, in 2022. He is currently pursuing the Ph.D. degree with the Department of Electrical and Electronic Engineering, The Hong Kong Polytechnic University, Hong Kong.

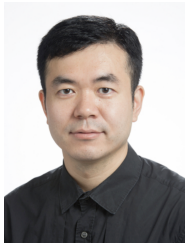
From April 2021 to September 2021, he was a Research Assistant with the Department of Computing, The Hong Kong Polytechnic University. His research interests include blockchain, physical-layer security and adversarial machine learning.

Sen Zhang is current a postdoctoral fellow in the Department of Electrical and Electronic Engineering at Hong Kong Polytechnic University. He received his Ph.D. degree in computer science from Southeast University, Nanjing, in 2022. His research interests include data mining, data privacy protection, big data technology, and complex data management.

Shen Su is currently an associate professor at the Institute of Advanced Cyberspace Technology, Guangzhou University, and executive deputy director of Guangzhou University-Qianxin Cloud Security Joint Lab. He received his B.S., M.S., and Ph.D. from Harbin Institute of Technology and studied abroad at the University of Arizona from 2012-2013. His main research interests include blockchain security, DNS secu-rity, routing security, and vehicular network security.

Jing Qiu received the Ph.D. degree in computer applications technology from Beijing Institute of Technology. She is currently a Professor and Ph.D. Supervisor of the Cyberspace Institute of Advanced Technology, Guangzhou University. She was a Visiting Scholar with the University of Southern California, LA, USA, under the supervision of Professor Craig A. Knoblock. Her current research interest is Cyberspace Security, Knowledge Representation, and Big Data Analysis.

Ran Guo received her B.S. degree and M.S. degree in Northeast Agricultural University. Where she works at Guangzhou University. Her research interests include computer network and artificial intelligence.



Haibo Hu is a Professor in the Department of Electronic and Information Engineering, Hong Kong Polytechnic University and the programme leader of BSc (Hons) in Information Security. His research interests include cybersecurity, data privacy, internet of things, and adversarial machine learning. He has published over 110 research papers in refereed journals, international conferences, and book chapters.

As principal investigator, he has received over 20 million HK dollars of external research grants from Hong Kong and mainland China. He is an associate editor of ACM Transactions on Privacy and Security (TOPS). He is the recipient of a number of titles and awards, including IEEE MDM 2019 Best Paper Award, WAIM Distinguished Young Lecturer, ICDE 2020 Outstanding Reviewer, VLDB 2018 Distinguished Reviewer, ACM-HK Best PhD Paper, Microsoft Imagine Cup, and GS1 Internet of Things Award. He is a senior member of ACM, IEEE and CCF, and a certified Cisco CCNA Security Trainer.