Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.088083

ARTICLE

AMLHunter: An On-Chain Risky Address Identification Method Based on Temporally Consistent Transaction Semantic Constraints and Generative Augmentation

Xiaolei Yin, Zihan Wang, Sanfeng Zhang* and Shouwei Li*

School of Cyber Science and Engineering, Southeast University, Nanjing, China

*Corresponding Authors: Sanfeng Zhang. Email: sfzhang@seu.edu.cn; Shouwei Li. Email: lishouwei@seu.edu.cn

Received: 28 June 2026; Accepted: 19 August 2026; Published: 15 September 2026

ABSTRACT: On-chain risky address identification is an important task in blockchain security analysis and digital asset risk management. In practical on-chain risk control, however, risky addresses are usually far fewer than benign ones. Fund flows also follow complex propagation paths and strict temporal orders, which makes it difficult for existing methods to handle class imbalance, structural semantic modeling, and information leakage under temporal split settings at the same time. To address these challenges, this paper proposes AMLHunter, an on-chain risky address identification method based on temporally consistent transaction semantic constraints and generative graph augmentation. AMLHunter first constructs a heterogeneous transaction graph from UpbitHack security incidents. In this graph, address nodes, transaction nodes, address-transaction-address paths, and projected inter-address fund flow edges are used to represent transaction event semantics and fund propagation structures. It then uses a teacher model and a conditional diffusion model to generate synthetic risky nodes in the risky address feature space, so as to alleviate the shortage of risky samples in the training set. To connect the generated nodes to the original transaction graph in a way that is consistent with real on-chain behavior, AMLHunter further designs a structural completion mechanism constrained by temporal consistency and transaction semantics. This mechanism is combined with generated-node quality gating and generated-edge quality filtering to reduce the influence of low-quality synthetic samples and low-confidence connections on the augmented graph. Experimental results on the UpbitHack-50k temporal split dataset show that AMLHunter achieves an area under the precision-recall curve (PR-AUC) of 0.523, a Precision@100 (P@100) of 0.850, and a risk-class Recall of 0.543. Compared with representative detection baselines from several categories and graph augmentation baselines, AMLHunter provides better or more stable performance on metrics related to risky candidate ranking.

KEYWORDS: On-chain risk identification; heterogeneous transaction graph; graph data augmentation; temporal consistency; class-imbalanced learning; blockchain security

1 Introduction

With the rapid growth of blockchain technology and digital asset trading, on-chain fund flows have become increasingly large and complex [1,2]. Security incidents such as exchange attacks, phishing scams, stolen asset transfers, and money laundering remain common in blockchain ecosystems. Unlike traditional Internet fraud, blockchain fraud is publicly traceable but strongly pseudonymous, and illicit funds are often hidden through multi-hop transfers, fund splitting, intermediate addresses, and interactions with exchanges or mixing services. Therefore, identifying addresses related to risk events in large-scale on-chain transaction records has become an important task in blockchain security analysis and digital asset risk management

[1–3]. In practical on-chain risk control, model output is usually used to generate high-risk candidate lists for analysts with limited audit budgets. This makes ranking-oriented metrics such as area under the precision-recall curve (PR-AUC), Precision@K, and recall at low false-positive rates especially important [1,2,4,5].

Existing methods for on-chain risky address identification can be roughly divided into feature-based methods and graph-based methods [1–3,6–8]. Feature-based methods rely on address-level statistics, such as transaction frequency, incoming and outgoing transfer volumes, active period, number of neighbors, and ratio of failed transactions [1,2]. These methods are easy to implement and interpret, but they mainly describe individual address behavior and make limited use of transaction structures and fund-flow relations. Graph-based methods model on-chain transactions as graphs and use graph learning to capture neighborhood patterns and fund propagation structures [2,3,6–9]. However, many existing methods still rely on homogeneous address graphs, making it difficult to preserve transaction-level semantics and fund-flow directions simultaneously. Moreover, most graph augmentation methods are designed for static graphs and may generate connections that violate transaction chronology in temporal split settings.

On-chain risky address identification faces three key challenges. First, risky addresses are usually much fewer than benign addresses, leading to severe class imbalance and weak recall for risky samples [10–12]. Second, on-chain transactions are naturally heterogeneous, because addresses and transactions represent different types of entities and transaction events contain attributes such as amount, gas, status, and timestamp [3,13]. Homogeneous address graph modeling may lose such fine-grained transaction semantics [13–16]. Third, temporal order is essential in blockchain transaction graphs. If node, edge, or neighborhood information from future time windows is used during training or augmentation, information leakage may occur and lead to overestimated performance [6,9,17–19]. Therefore, effective on-chain graph augmentation should generate risky samples that are not only discriminative in feature space, but also structurally plausible and temporally valid.

To address these issues, we propose AMLHunter, an on-chain risky address identification method based on temporally consistent transaction semantic constraints and generative graph augmentation. AMLHunter first constructs a heterogeneous transaction graph with address nodes, transaction nodes, address-transaction relations, and projected inter-address fund-flow edges. It then uses a teacher-guided conditional diffusion model to generate minority-class risky address features. To integrate generated nodes into the transaction graph, AMLHunter further designs a structural completion mechanism constrained by temporal consistency, fund-flow direction consistency, and local transaction semantic consistency. Generated-node quality gating and generated-edge quality filtering are also used to reduce noisy synthetic samples and unreliable connections. The entire augmentation process is restricted to the training time window, while the validation and test sets keep their original temporal split to avoid future information leakage.

The main contributions are summarized as follows.

1. We study on-chain graph augmentation under temporal split settings and identify three key requirements for valid augmentation: avoiding future structural leakage, preserving fund-flow direction, and maintaining transaction semantic consistency.
2. We propose AMLHunter, a generative graph augmentation framework for on-chain risky address identification. AMLHunter generates minority-class risky address features with a teacher-guided conditional diffusion model and completes generated-node structures under temporal and transaction semantic constraints.
3. We construct a temporal split heterogeneous transaction graph from UpbitHack security incidents and evaluate AMLHunter against representative detection baselines and graph augmentation baselines. Experimental results show that AMLHunter improves PR-AUC, Precision@100, and risky-class recall under a unified GraphSAGE classifier.

2 Related Work

2.1 On-Chain Risk Identification and Heterogeneous Graph Learning

On-chain risky address identification aims to detect accounts associated with illicit activities, such as attacks, asset theft, phishing scams, and money laundering, from blockchain transaction records [2,6–9]. Early methods mainly rely on hand-crafted rules or address-level statistical features, including transaction frequency, incoming and outgoing fund volumes, active time span, number of neighbors, and failed transaction ratio [1,2]. These methods are usually interpretable, but they describe individual address behavior in isolation and are less effective at capturing cross-address propagation and multi-hop fund-flow patterns.

With the development of graph learning, many studies model blockchain transaction data as graphs, where addresses are treated as nodes and transaction relationships are treated as edges [2,3,6–9]. Homogeneous graph neural networks (GNNs), such as graph convolutional network (GCN), graph attention network (GAT), and GraphSAGE, capture neighborhood structures through graph convolution, attention aggregation, or inductive sampling [20–22]. Temporal graph models such as TGAT, TGN, JODIE, DyRep, and EvolveGCN further characterize graph evolution and temporal interaction patterns [17,18,23–25]. These studies show that transaction topology, temporal order, and structural evolution are important for on-chain risk detection.

Blockchain transaction networks are naturally heterogeneous because addresses and transactions represent different types of entities, and the sending, receiving, and projected fund-flow relations have clear semantic differences [3,13]. Heterogeneous graph learning methods such as R-GCN, HAN, and HGT preserve multi-relation and multi-type semantic information through relation-specific parameters, meta-path attention, or type-aware Transformer mechanisms [14–16]. In fraud detection, SemiGNN, CARE-GNN, PC-GNN and H2-FDetector further show the effectiveness of graph neural networks in modeling suspicious behavior, camouflage, imbalance, and homophilic or heterophilic fraud relationships [26–29]. However, most existing on-chain and heterogeneous graph fraud detection methods focus on representation learning and supervised detection. How to generate minority risky nodes and connect them to heterogeneous transaction graphs in a temporally valid and semantically consistent manner remains less explored.

2.2 Imbalanced Graph Learning and Generative Augmentation

Class imbalance is a common problem in risk detection, where risky addresses usually represent only a small fraction of all addresses [10–12]. Traditional methods such as undersampling, oversampling, class reweighting, threshold adjustment, Synthetic Minority Oversampling Technique (SMOTE), and Focal Loss can reduce bias toward majority classes, but are mainly designed for Euclidean data or standard classification settings [10,30]. In graph data, generated minority samples must not only have plausible features, but also require reasonable neighborhood structures to participate in message passing.

Several graph imbalance learning methods have been proposed to address this issue. GraphSMOTE generates minority-class nodes in the embedding space and uses an edge generator to add relational information [11]. GraphENS synthesizes ego networks for minority-class nodes to reduce minority-neighborhood overfitting [12]. GraphSHA, GraphMixup, ImGAGN, TAM, and ImGCL address class imbalance from different perspectives, such as hard sample synthesis, mixup augmentation, generative adversarial learning, topology-aware margins, and contrastive learning [31–35]. These studies indicate that both feature generation and structural construction are important for imbalanced node classification.

Generative models provide a flexible way to synthesize samples of minority-class. Compared to duplication or linear interpolation, generative models can better capture complex minority-class distributions [11,33,36]. Diffusion models, including denoising diffusion probabilistic models (DDPM) and

score-based generative models, have shown strong distribution modeling ability and stable training behavior [36,37]. Recent studies also extend diffusion models to graph generation tasks [38]. However, most graph augmentation methods are designed for static graphs and rarely consider transaction time, fund-flow direction, and future information leakage in on-chain transaction graphs. In this paper, we study generative graph augmentation in temporal split settings and impose temporal validity, fund-flow direction consistency, and local transaction semantic consistency during generated-node structural completion.

3 Methodology

Fig. 1 gives an overview of AMLHunter. The method first constructs a temporally split heterogeneous transaction graph from UpbitHack security incidents. Then it generates risky address features for minority-class using a teacher-guided conditional diffusion model. These generated nodes are connected to the training graph by structural completion under temporal consistency, fund-flow direction consistency, and transaction semantic consistency. Generated-node quality gating and generated-edge quality filtering are further used to reduce noisy synthetic samples and unreliable connections. Finally, a GraphSAGE classifier is trained on the augmented graph for on-chain risky address identification.

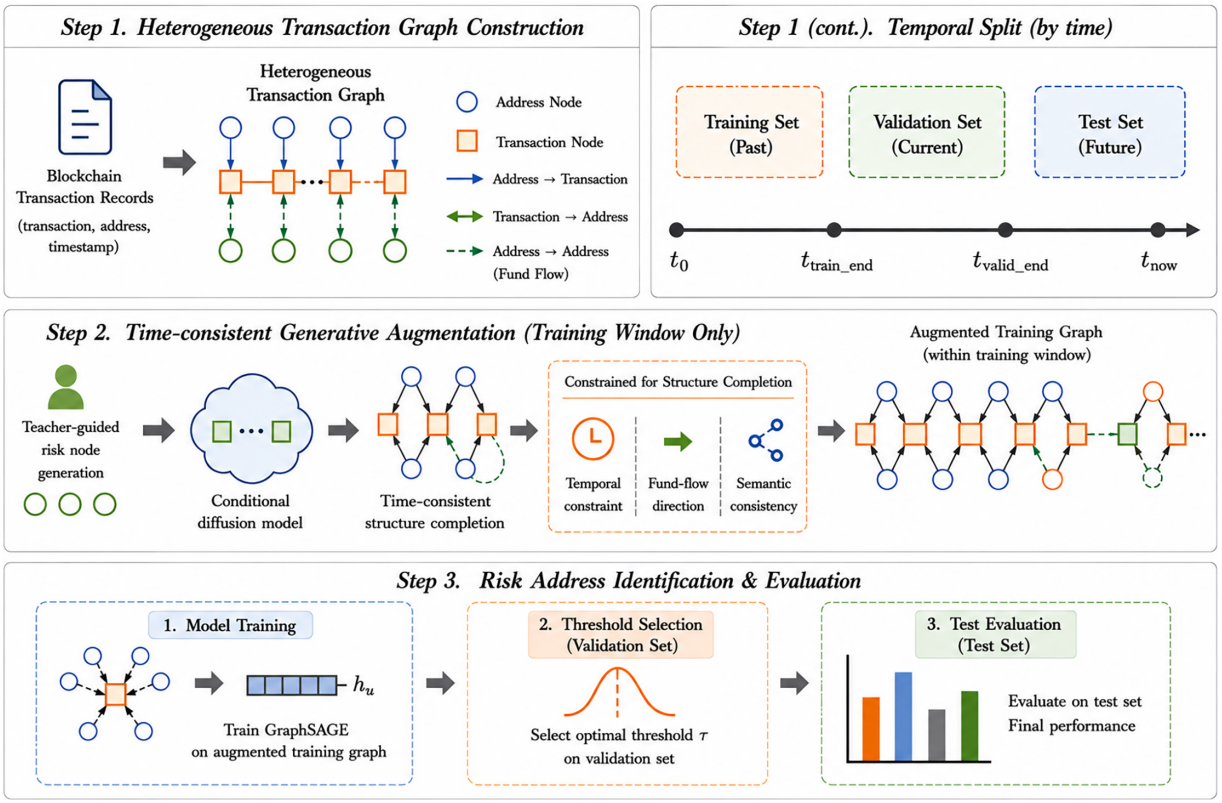

Figure 1: Overview of the proposed AMLHunter framework.

3.1 Motivation

In temporal split settings, graph augmentation for on-chain risk identification must satisfy three requirements: avoiding future structural leakage, preserving fund-flow direction, and maintaining local transaction semantic consistency. As shown in Fig. 2, conventional graph augmentation may connect risky generated nodes to future addresses, reverse fund-flow paths, or semantically inconsistent neighborhoods.

These invalid connections can distort message passing and lead to overestimated performance. Therefore, the generated nodes should be structurally completed only within the training time window and under temporal, directional, and semantic constraints.

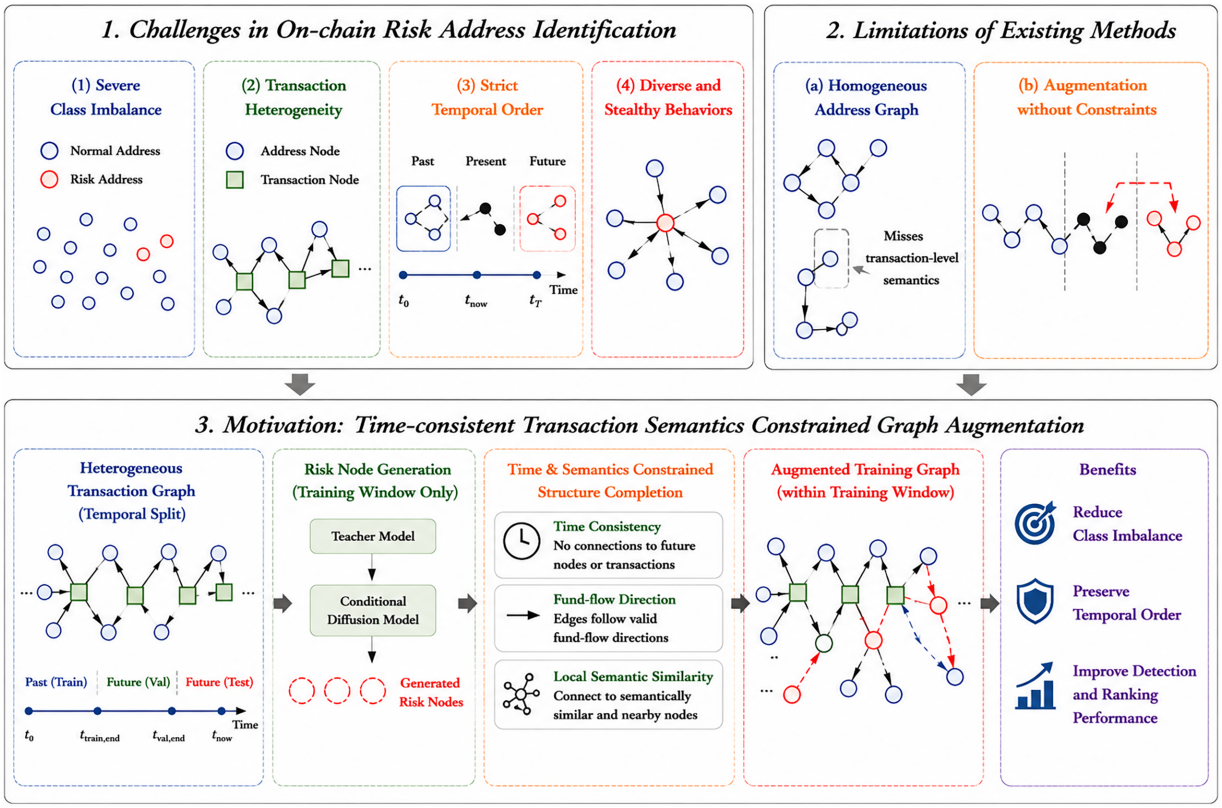

Figure 2: Motivation of temporally consistent transaction semantic constrained graph augmentation. AMLHunter avoids future structural leakage, fund-flow direction violations, and semantically inconsistent generated connections.

In this setting, augmentation validity is more difficult than in conventional static graph learning. A generated risky node may look plausible in the feature space, but it becomes invalid if its edges connect to future addresses, reverse fund-flow directions, or link to semantically incompatible neighborhoods. Therefore, AMLHunter treats generated-node structural completion as a constrained graph construction problem rather than a simple edge prediction.

3.2 Problem Definition

Given a heterogeneous transaction graph $G = (V, E, X, Y)$, where $V = V_a \cup V_t$ contains address nodes and transaction nodes, E denotes typed transaction relations, X is the node feature matrix, and Y contains address labels, this paper studies binary classification over address nodes. Each address node is labeled as risky or benign.

To avoid information leakage, the address nodes are divided according to their first appearance times. The training set contains only historical addresses and transactions, while validation and test addresses appear in later time windows. All generated nodes, candidate neighbors, and generated edges are constructed only within the training time window. The goal is to improve the identification of risky addresses by augmenting risky nodes from the minority-class without using future structural or label information.

3.3 Heterogeneous Transaction Graph Construction

Existing on-chain graph learning methods often project transaction records directly into homogeneous address graphs, which may lose transaction-level semantic information such as amount, timestamp, gas usage, and transaction status. To preserve both transaction event semantics and fund propagation structures, this paper models on-chain transaction records as a heterogeneous transaction graph, as shown in Fig. 3.

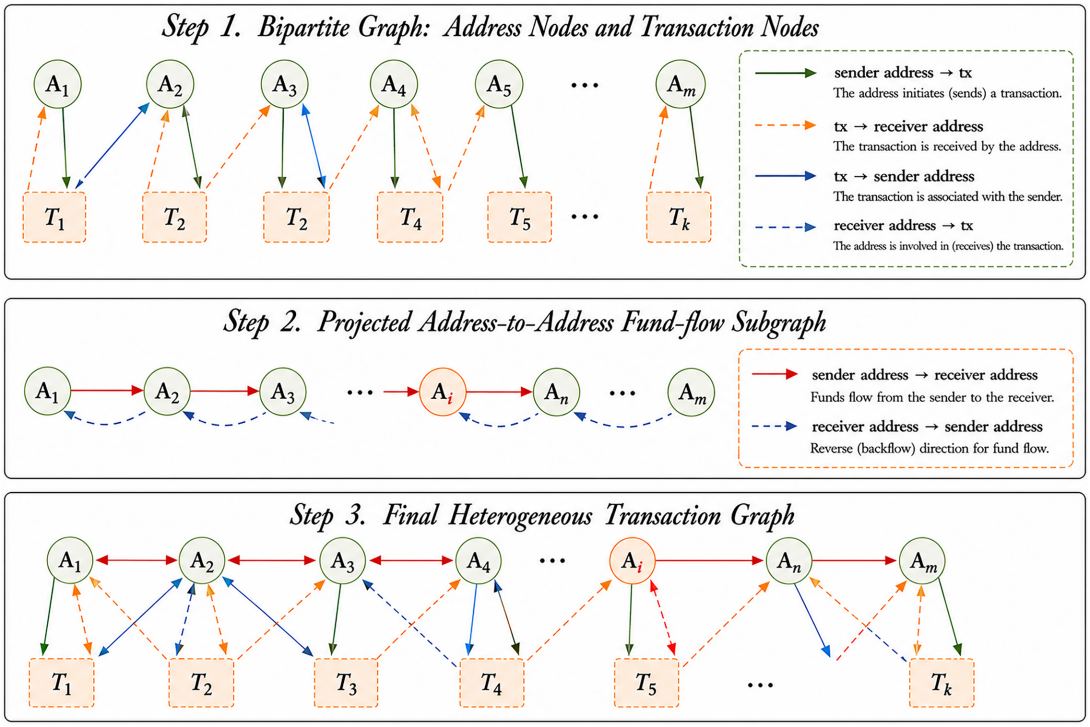

Figure 3: Construction of the heterogeneous transaction graph.

The graph contains two types of nodes and six types of directed relations. Address nodes represent on-chain accounts, while transaction nodes represent transaction events. The six relation types are sender-to-transaction, transaction-to-receiver, transaction-to-sender, receiver-to-transaction, sender-to-receiver, and receiver-to-sender. Address-transaction-address paths preserve transaction event semantics, while projected address-address edges explicitly describe fund-flow directions. Address node features include historical transaction statistics such as in-degree, out-degree, inflow and outflow volumes, number of unique neighbors, failed transaction ratio, and temporal activity patterns. Transaction node features include transaction amount, gas price, gas used, failure status, and normalized timestamp.

The projected address-to-address edges are not redundant with address-transaction-address paths because they encode different structural information. Address-transaction-address paths preserve transaction-level event semantics (e.g., amount, gas usage, timestamp, and status), whereas projected address-to-address edges explicitly model direct fund-flow propagation and shorten message-passing distance between interacting addresses. These two relation types therefore provide complementary rather than duplicated structural information. Moreover, projected edges are typed, directed, and constructed only from training-window transactions, which further reduces the risk of redundant message propagation and overfitting.

3.4 Generative Minority-Class Augmentation

As shown in Fig. 4, AMLHunter first trains a teacher classifier on the original training graph to estimate the risk decision boundary. The teacher provides soft label guidance for risky address generation, because soft labels contain uncertainty information near the boundary between risky and benign addresses. A conditional diffusion model is then trained in the risky address feature space to generate minority-class risky nodes [36,37].

$$\mathcal{L}_{diff} = \mathrm{E}_{x_0,\varepsilon,t}\left[\left\|\varepsilon - \varepsilon_\theta(x_t, t, s)\right\|_2^2\right], \tag{1}$$

where s is the teacher-provided soft-label condition and ε_θ is the predicted noise.

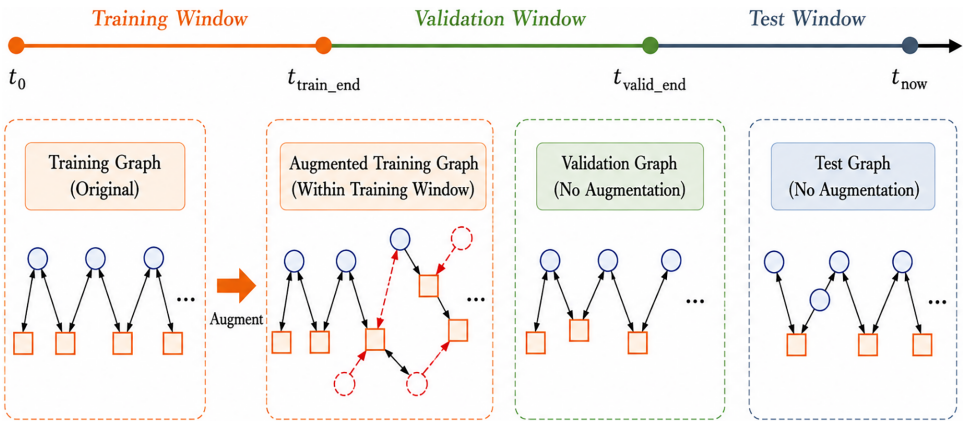

Figure 4: Generative minority-class augmentation with temporally consistent transaction semantic constraints.

The teacher classifier is a multi-layer perceptron (MLP) trained only on the original training-window data. It provides soft-label guidance for risky node generation and quality gating, while validation and test nodes are excluded.

The denoising network is implemented as a conditional U-Net. It takes noisy features, timestep embeddings, and teacher-provided soft-label conditions as inputs to generate risky address features. Specifically, the soft-label condition s is embedded through a linear projection and concatenated with the noisy feature representation and timestep embedding before noise prediction.

The diffusion model is trained for 6 epochs with a learning rate of 1×10^{-4}. During generation, we use a simplified two-step denoising process for low-dimensional address feature synthesis. The noise schedule follows a linear beta schedule with $\beta \in [1 \times 10^{-4}, 2 \times 10^{-2}]$. The number of generated risky nodes is determined by

$$N_{gen} = \alpha(N_{major} - N_{minor}), \tag{2}$$

where α controls the augmentation ratio, and N_{major} and N_{minor} denote the numbers of benign and risky addresses in the training set, respectively.

For each generated node, AMLHunter records its source risky address in the training set. This source address provides the temporal and structural context used in the subsequent structural completion step. In this way, generated nodes are not treated as isolated synthetic samples, but are associated with real risky addresses whose local transaction behavior can guide candidate neighbor selection and edge construction.

3.5 Temporally Consistent Structural Completion

Generating risky node features alone is not sufficient for graph learning, because node representations depend on neighborhood message passing. A generated node without effective connections cannot provide useful structural information, while connections that violate transaction chronology or fund-flow semantics may introduce noise. AMLHunter therefore completes generated-node structures before adding them to the training graph.

For each generated risky node, candidate neighbors are selected from the local neighborhood and candidate address set of its source risky address within the training time window. Nodes and edges from validation and test windows are excluded to avoid future information leakage. Given the representation of a generated node and that of a candidate node, an edge predictor estimates their connection probability under different relation types. The edge predictor is trained with real edges and negatively sampled edges from the training graph, following the encoder-decoder view of graph link prediction [39].

The edge predictor is implemented as an MLP-based relation scorer. Given a generated node u, a candidate node v, and relation type r, we first obtain their node representations h_u and h_v from the training graph. The input to the edge predictor is $[h_u \| h_v \| |h_u - h_v| \| h_u \odot h_v \| e_r]$, where e_r is the learnable embedding of relation type r. The predictor outputs $p_r(u, v)$ through a sigmoid layer. It is trained with observed training-window edges as positive samples and randomly sampled non-edges within the training graph as negative samples. Negative edges are randomly sampled from non-connected node pairs within the training-window graph while preserving relation-type constraints. The encoder used to obtain node representations is trained and applied only on the training-window heterogeneous graph. Validation and test information is excluded from edge predictor training, candidate selection, and message passing. All representations are obtained only from the training-window graph.

Candidate edges are filtered by three constraints. Formally, the retained generated edge set is defined as

$$E_{gen} = \{(u, v, r) \mid p_r(u, v) \geq \tau_e,\ t_u, t_v \leq t_{train},\ \phi_r(u, v) = 1\}, \tag{3}$$

where $p_r(u, v)$ is the predicted confidence score of relation type r, τ_e is the edge confidence threshold, t_{train} denotes the end time of the training window, and $\phi_r(u, v)$ indicates whether the candidate edge satisfies fund-flow direction and local transaction semantic constraints. The temporal validity constraint requires both endpoints to appear within the training time window. The fund-flow direction constraint prevents generated edges from violating the chronological direction of on-chain fund propagation. The neighborhood semantic constraint encourages generated nodes to connect to addresses whose local transaction behavior is similar to that of the source risky address. The candidate edge pool is controlled by `edge_top_factor`, and only high-confidence candidate edges are retained. Through this process, generated nodes not only expand the minority-class feature distribution, but also form graph connections that are more consistent with real on-chain transaction behavior.

$$\phi_r(u, v) = \mathbb{I}[\phi^{time}(u, v) = 1] \cdot \mathbb{I}[\phi_r^{dir}(u, v) = 1] \cdot \mathbb{I}[\phi^{sem}(u, v) = 1]. \tag{4}$$

Here, ϕ^{time} requires both endpoints and the transaction context used for candidate construction to belong to the training window. ϕ_r^{dir} checks whether the candidate relation type is consistent with the fund-flow direction of the source risky address. For example, if the source risky address mainly sends funds to a local neighbor in the historical graph, the generated node is allowed to form a sender-to-receiver projected relation with the candidate neighbor, while the reverse direction is retained only when such reverse behavior exists in the source local context. ϕ^{sem} measures local transaction semantic consistency between the

candidate neighbor and the source risky address using normalized transaction statistics, including in-degree, out-degree, inflow, outflow, number of unique neighbors, failed transaction ratio, and temporal activity features. A candidate edge is retained only when all three constraints are satisfied.

3.6 Quality Filtering and Final Classification

After structural completion, AMLHunter applies generated-node quality gating and generated-edge quality filtering. For generated nodes, the trained teacher model evaluates whether each generated feature is confidently classified as risky. Generated nodes with teacher confidence lower than `generated_teacher_min_prob` are discarded. For generated edges, the edge predictor assigns confidence scores to candidate edges, and low-confidence edges are removed according to `edge_score_quantile`. These two filters reduce the influence of noisy generated samples and unreliable connections.

The augmented training graph is defined as

$$G' = \left(V_{train} \cup V_{gen}, E_{train} \cup E_{gen}\right), \tag{5}$$

where V_{gen} and E_{gen} denote the generated risky nodes and their retained generated edges. The overall optimization objective is

$$\mathcal{L} = \mathcal{L}_{cls} + \lambda_1 \mathcal{L}_{diff} + \lambda_2 \mathcal{L}_{edge}, \tag{6}$$

where $\mathcal{L}_{cls}$ is the risky address classification loss, $\mathcal{L}_{diff}$ is the diffusion loss, $\mathcal{L}_{edge}$ is the edge prediction loss, and λ_1 and λ_2 are loss weights.

For fair comparison among augmentation methods, the main experiments use GraphSAGE as the unified downstream classifier [22]. The classification threshold is selected on the validation set according to Macro-F1, and the test set is used only for final evaluation. Additional experiments also evaluate the augmented graph with heterogeneous GNN classifiers such as R-GCN, HAN, and HGT [14–16].

4 Experiments and Analysis

4.1 Dataset and Experimental Settings

The experiments are conducted on the self-constructed UpbitHack-50k transaction graph dataset. This dataset is built from Ethereum transaction records associated with the UpbitHack security incident and contains both address nodes and transaction nodes. Edges represent sending and receiving relations between addresses and transactions, as well as projected inter-address fund-flow relations. Address nodes are labeled as benign or risky according to whether they are associated with risk-related accounts.

The constructed heterogeneous transaction graph contains 18,270 address nodes, 50,000 transaction nodes, and six types of edge relation. Among the address nodes, 1653 are risky addresses and 16,617 are benign addresses, resulting in an imbalance ratio of approximately 10.05:1. The detailed statistics of the UpbitHack-50k dataset are summarized in Table 1. The address nodes are divided into training, validation, and test sets according to their first appearance times. The training set contains 12,789 addresses, including 1138 risky addresses. The validation set contains 1827 addresses, including 218 risky addresses. The test set contains 3654 addresses, including 297 risky addresses. The last first appearance time in the training set is earlier than the first first appearance time in the validation set, and the validation set is earlier than the test set. Therefore, model training and augmentation do not use address information from future time windows.

Table 1: Statistics of the UpbitHack-50k dataset.

Item	Number
Address nodes	18,270
Transaction nodes	50,000
Edge relation types	6
Risky/benign addresses	1653/16,617
Class imbalance ratio	10.05:1
Train/Val/Test addresses	12,789/1827/3654
Train/Val/Test risky addresses	1138/218/297
Time span	2096.11 days

In the experiments, the training set is used for model training and data augmentation, the validation set is used for threshold selection and hyperparameter tuning, and the test set is used only for final evaluation. All generated nodes and generated edges are added only to the training graph. Nodes, labels, and edge structures in the validation and test sets are excluded from augmentation to prevent temporal leakage.

We use Macro-F1, PR-AUC, ROC-AUC, Precision@K, Risk Recall, Risk F1, MCC, Lift@100, and Recall@FPR = 5% for evaluation [2,4,5,40]. Risk F1 denotes the F1-score of the risky class and is calculated as $F1_{\text{risk}} = 2P_{\text{risk}}R_{\text{risk}}/(P_{\text{risk}} + R_{\text{risk}})$, where P_{risk} and R_{risk} denote the precision and recall of the risky class, respectively. Since risky addresses are rare and practical risk control usually focuses on high-risk candidate ranking, PR-AUC, Precision@100, Risk Recall, and Lift@100 are treated as the main metrics. All main experiments are repeated with three random seeds, and the mean and standard deviation are reported.

4.2 Baselines and Implementation Details

We compare AMLHunter with representative detection baselines and graph augmentation baselines. Detection baselines include Random Forest, XGBoost, node2vec + XGBoost, GCN, GAT, GraphSAGE, R-GCN, HAN, HGT, and a TGN-style temporal graph model. These baselines cover feature-based classification, graph embedding based detection, homogeneous GNNs, heterogeneous GNNs, and temporal graph modeling [14–16,18,20–22,41,42].

To improve reproducibility, Table 2 summarizes the main hyperparameters used in AMLHunter. Unless otherwise specified, the augmentation-method comparison uses GraphSAGE as the unified downstream classifier, while the classification-head comparison fixes the same augmented graph and changes only the downstream detection model. The validation set is used for hyperparameter selection and threshold selection, and the test set is used only for final evaluation.

For comparison of graph augmentation, we use GraphSAGE as the unified downstream classifier and compare No Augmentation, GraphSMOTE-style augmentation, HeteroSOLID-style controlled baseline, and AMLHunter [11,22,36,37]. HeteroSOLID-style augmentation is implemented as a controlled baseline rather than a previously published method. It adapts the idea of heterogeneous graph-aware minority augmentation to the on-chain transaction graph. Specifically, risky address representations are first obtained from the training graph, and synthetic risky nodes are generated around minority-class risky addresses in the learned representation space. Candidate connections are then constructed only within the training window. Unlike AMLHunter, this controlled baseline does not use the teacher-guided conditional diffusion generator, generated-node quality gating, or the proposed joint temporal and transaction semantic constraints. We include this baseline to examine whether heterogeneous graph-aware minority augmentation alone is

sufficient for on-chain risky address identification. This setting isolates the effect of the augmentation strategies and avoids performance differences caused by different classification heads. The generated nodes and generated edges are constructed only within the training time window, while the validation and test sets keep their original temporal splits unchanged.

Table 2: Main hyperparameters used in AMLHunter.

Module	Setting
Temporal split	Split by the first appearance time of addresses
Classifier	GraphSAGE, 150 epochs, hidden dim. 128, lr 0.003, weight decay 1×10^{-4}
Teacher	MLP teacher, 3 teacher/encoder joint epochs
Diffusion	Conditional U-Net, 6 epochs, lr 1×10^{-4}, 2 steps, linear β schedule
Augmentation	Ratio 0.25, batch size 32
Structural completion	`time_semantic`, MLP relation scorer, edge top factor 2, edge score quantile 0.75
Quality gating	Generated-node teacher threshold 0.55
Threshold selection	Validation Macro-F1

To avoid inconsistencies between detection baselines and augmentation baselines, the vanilla GraphSAGE baseline in Table 3 and the No Augmentation setting in Table 4 are reported under the same experimental protocol for augmentation comparison. In the initial manuscript, the two GraphSAGE results were obtained under different experimental settings, including different feature preprocessing, training hyperparameters, and random seed protocols, which caused the observed discrepancy. In the revised manuscript, we unify the temporal split, node features, GraphSAGE architecture, training epochs, learning rate, random seeds, and validation-based threshold selection strategy for all GraphSAGE-based experiments. Therefore, the GraphSAGE result in Table 3 is identical to the No Augmentation result in Table 4, while Table 4 further evaluates how different augmentation strategies affect the same downstream classifier.

Table 3: Comparison with representative on-chain risky address detection methods. Values are reported as mean (standard deviation). The best result is shown in bold, and the second-best result is underlined.

Method	Macro-F1	PR-AUC	ROC-AUC	MCC	P@100	R-Recall
Random Forest	0.635 (0.005)	0.316 (0.006)	0.668 (0.007)	0.314 (0.011)	0.593 (0.021)	0.216 (0.010)
XGBoost	0.523 (0.115)	0.316 (0.007)	0.684 (0.008)	0.179 (0.116)	0.540 (0.027)	**0.552 (0.125)**
node2vec + XGBoost	0.546 (0.041)	0.193 (0.020)	0.652 (0.026)	0.136 (0.030)	0.353 (0.051)	0.370 (0.088)
GAT	0.565 (0.031)	0.191 (0.021)	0.628 (0.019)	0.190 (0.031)	0.347 (0.025)	0.156 (0.133)
GCN	0.669 (0.027)	0.413 (0.017)	**0.802 (0.022)**	0.354 (0.037)	0.727 (0.049)	0.492 (0.094)
GraphSAGE	0.606 (0.114)	0.408 (0.098)	0.684 (0.082)	0.425 (0.037)	0.780 (0.118)	0.439 (0.090)
HAN	0.655 (0.026)	0.420 (0.039)	0.767 (0.051)	0.326 (0.030)	0.760 (0.017)	0.451 (0.100)
HGT	0.559 (0.098)	0.338 (0.124)	0.692 (0.105)	0.187 (0.146)	0.617 (0.198)	0.477 (0.055)
R-GCN	0.692 (0.050)	0.475 (0.042)	0.716 (0.034)	0.408 (0.082)	0.830 (0.017)	0.477 (0.110)
TGN-style	0.572 (0.009)	0.173 (0.020)	0.652 (0.032)	0.179 (0.025)	0.230 (0.010)	0.398 (0.137)
AMLHunter	**0.720 (0.042)**	**0.523 (0.027)**	0.793 (0.012)	**0.447 (0.082)**	**0.850 (0.079)**	0.543 (0.087)

Table 4: Risk identification performance of different augmentation methods under the unified GraphSAGE classification head. Values are reported as mean (standard deviation). The best result is shown in bold, and the second-best result is underlined.

Method	Macro-F1	PR-AUC	ROC-AUC	P@50	P@100	R-Recall	MCC	Lift@100
No Aug.	0.606 (0.114)	0.408 (0.098)	0.684 (0.082)	**1.000 (0.000)**	0.780 (0.118)	0.439 (0.090)	0.425 (0.037)	10.391 (0.000)
GraphSMOTE	0.598 (0.060)	0.294 (0.080)	0.689 (0.020)	0.833 (0.289)	0.533 (0.228)	0.259 (0.018)	0.269 (0.021)	10.001 (0.063)
HeteroSOLID-style	0.616 (0.030)	0.437 (0.044)	0.747 (0.043)	0.987 (0.012)	0.780 (0.053)	0.542 (0.045)	0.436 (0.020)	10.102 (0.068)
AMLHunter	**0.720 (0.042)**	**0.523 (0.027)**	**0.793 (0.012)**	0.993 (0.012)	**0.850 (0.079)**	**0.543 (0.087)**	**0.447 (0.082)**	**10.458 (0.977)**

All experiments were conducted on the UpbitHack-50k temporal split. For long-tailed features such as amount, gas price, and gas used, a logarithmic transformation is applied to reduce the effect of extreme values. Continuous features such as timestamps are normalized. The final GraphSAGE classifier is trained for 150 epochs with a hidden dimension of 128, a learning rate of 0.003, and a weight decay of 1e–4. The final setting of AMLHunter uses `over_sample_rate`=0.25, `edge_constraint_mode`=`time_semantic`, `edge_score_quantile`=0.75, and `generated_teacher_min_prob`=0.55. During prediction, the classification threshold is selected in the validation set using Macro-F1.

In fairness, the comparison between the augmentation methods is separate from the comparison between the classification heads. All augmentation methods use the same GraphSAGE classifier, temporal split, and validation-based threshold selection strategy, so the observed differences come mainly from the augmentation strategy rather than the downstream classifier.

4.3 Main Results

We first compare AMLHunter with representative on-chain risky address detection methods. The results are shown in Table 3.

Table 3 shows that AMLHunter achieves the best Macro-F1, PR-AUC, MCC, and P@100, while obtaining a competitive Risk Recall close to the best baseline. Feature-based methods provide useful but limited performance because they ignore graph structures. GNN-based methods improve risk detection by exploiting transaction relations, and R-GCN is the strongest baseline among them. However, AMLHunter further improves PR-AUC and P@100, indicating a stronger ranking ability for the selection of high-risk candidates. Although GCN obtains a slightly higher ROC-AUC, AMLHunter performs better on metrics that are more important for imbalanced risk detection and practical audit scenarios.

To further examine the effect of the augmentation strategy, we compare different augmentation methods under the same GraphSAGE classification head. The results are shown in Table 4.

Table 4 shows that AMLHunter achieves the best overall performance under the unified GraphSAGE head. Compared with No Augmentation, AMLHunter improves PR-AUC, P@100, and Risk Recall, showing that minority-class generation and structural completion are useful under severe imbalance. GraphSMOTE-style augmentation performs worse, suggesting that interpolation-based generation and unconstrained edge completion may introduce structural noise. HeteroSOLID-style augmentation improves Risk Recall but remains lower in PR-AUC and P@100, indicating that temporal and transaction semantic constraints are important for ranking high-risk addresses under limited audit budgets.

4.4 Ablation Study

4.4.1 Structural Constraint Ablation under the Final Configuration

We first study the contribution of each structural constraint in AMLHunter. Different from the preliminary constraint analysis in the original manuscript, all ablation variants in the revised experiment are derived from the final AMLHunter configuration. Specifically, the generation ratio is fixed as `over_sample_rate=0.25`, the edge confidence quantile is fixed as `edge_score_quantile=0.75`, the generated-node teacher gating threshold is fixed as `generated_teacher_min_prob=0.55`, and the candidate edge pool size is fixed as `edge_top_factor=2`. All variants use GraphSAGE as the downstream classifier and follow the same temporal split, random seeds, and validation-based threshold selection strategy. Each ablation variant disables only one structural constraint, while the *w/o all structural constraints* variant disables temporal consistency, fund-flow direction consistency, and local transaction semantic consistency at the same time. For this ablation study, we additionally report Recall@FPR = 5% to evaluate the ability of each variant to recover risky addresses under a stricter low-false-positive setting.

Table 5 shows that the full AMLHunter configuration achieves the best performance on PR-AUC, P@100, risky-class Recall, and Recall@FPR = 5%. Compared with the variant without temporal consistency, Full AMLHunter improves PR-AUC by 0.0527, P@100 by 0.0567, and Recall@FPR = 5% by 0.0370. This indicates that temporal constraints help suppress noisy generated connections that violate the chronological order of blockchain transactions. Compared with the variants without fund-flow direction consistency or local transaction semantic consistency, Full AMLHunter still maintains better P@100 and Recall@FPR = 5%, suggesting that direction and semantic constraints improve the ranking quality of high-confidence risky candidates.

Table 5: Ablation results of structural constraints under the final AMLHunter configuration. Values are reported as mean (standard deviation). The best result is shown in bold.

Setting	PR-AUC	P@100	R-Recall	R@FPR = 5%
Full AMLHunter	**0.5228 (0.0267)**	**0.8500 (0.0794)**	**0.5432 (0.0865)**	**0.3322 (0.0253)**
w/o temporal constraint	0.4701 (0.0224)	0.7933 (0.0981)	0.5241 (0.0856)	0.2952 (0.0576)
w/o direction constraint	0.4994 (0.0439)	0.8467 (0.0321)	0.5387 (0.0777)	0.3210 (0.0070)
w/o semantic constraint	0.4982 (0.0352)	0.8467 (0.0451)	0.4489 (0.1426)	0.3210 (0.0108)
w/o all structural constraints	0.5126 (0.0084)	0.8300 (0.0265)	0.4871 (0.1413)	0.3053 (0.0332)

The variant without all structural constraints performs worse than Full AMLHunter on all reported metrics, especially on P@100, risky-class Recall, and R@FPR = 5%. This result confirms that generated nodes should not be connected to the historical graph by unconstrained edge completion. In on-chain risk control, Recall under a low false-positive rate is particularly important because analysts usually inspect only a limited number of high-risk candidates. Therefore, the ablation results support the design choice of combining temporal consistency, fund-flow direction consistency, and local transaction semantic consistency in the final AMLHunter structural completion strategy.

4.4.2 Quality Filtering and Generated-Node Gating

We then analyze the effect of generated-node quality gating and generated-edge quality filtering. The comparison includes an optimized GraphSAGE baseline without augmentation (Optimized GraphSAGE),

the proposed method with only edge quality filtering, and variants with teacher-based generated-node quality gating under different thresholds. Unlike the No Augmentation setting in Table 4, which is designed for fair comparison among different augmentation methods under a unified GraphSAGE protocol, Optimized GraphSAGE uses a separately optimized configuration for this ablation study. Therefore, these two settings serve different evaluation purposes and should be interpreted separately.

It is worth noting that Optimized GraphSAGE is not identical to the No Augmentation baseline in Table 4. The former represents an optimized non-augmented classifier configuration used to evaluate the difficulty of the augmentation task, while the latter is the unified baseline used for comparing different augmentation strategies.

Table 6 shows that edge filtering improves the performance compared with the optimized non-augmented baseline, while teacher-based node gating further improves PR-AUC and Risk Recall. With teacher-based node gating, the threshold of 0.55 obtains the highest PR-AUC, ROC-AUC, and Risk Recall. This confirms that filtering low-quality generated nodes can further improve risk ranking and risky-class coverage. The no-augmentation baseline remains slightly higher on P@100 and Lift@100, suggesting that the original graph already ranks some highly confident risky candidates well. Therefore, the main benefit of quality filtering is to improve overall ranking stability and risk coverage rather than uniformly improving every metric.

Table 6: Ablation results of quality filtering and generated-node gating under an optimized non-augmented reference setting. Values are reported as mean (standard deviation). The best result is shown in bold, and the second-best result is underlined. Optimized GraphSAGE reference denotes a stronger non-augmented GraphSAGE configuration used only for evaluating the effectiveness of quality filtering and generated-node gating. It is not directly comparable with the unified GraphSAGE baseline in Table 4, which is designed for fair comparison among different augmentation strategies. The purpose of the optimized reference is not to replace the unified baseline, but to provide a stronger non-augmented comparison for analyzing the additional contribution of quality filtering modules.

Setting	Macro-F1	PR-AUC	ROC-AUC	P@100	R-Recall	MCC	Lift@100
Optimized GraphSAGE	0.7211 (0.0176)	0.5082 (0.0377)	0.7664 (0.0494)	**0.8600 (0.0000)**	0.5376 (0.0573)	0.4449 (0.0370)	**10.5806 (0.0000)**
Edge filter	**0.7253 (0.0327)**	0.5193 (0.0678)	0.7654 (0.0401)	0.8167 (0.0252)	0.5196 (0.1179)	**0.4536 (0.0666)**	10.0475 (0.3096)
Node gate 0.50	0.7003 (0.0096)	0.5011 (0.0485)	**0.7960 (0.0351)**	0.8067 (0.0551)	0.4635 (0.0962)	0.4056 (0.0193)	9.9244 (0.6776)
Node gate 0.55	0.7197 (0.0416)	**0.5228 (0.0267)**	0.7934 (0.0118)	0.8500 (0.0794)	**0.5432 (0.0865)**	0.4466 (0.0815)	10.4576 (0.9765)

4.4.3 Structural Completion Schemes

We also examine whether generated nodes need structural completion. Since graph neural networks rely on neighborhood message passing, generating risky nodes only in the feature space may not provide effective training signals. We compare three structural integration schemes: generated features with edge-predictor-based structural completion, generated features without edge completion, and generated features with random edge completion.

The left part of Fig. 5 shows that generated nodes require proper structural completion to be useful for graph learning. Compared with Full, Feat-only clearly decreases Macro-F1, PR-AUC, and Risk F1, indicating that feature-space generation alone is insufficient. Rand-edge improves PR-AUC compared with Feat-only, but its Risk F1 remains lower than Full. This suggests that random connections may help ranking to some extent but also introduce structural noise. In contrast, edge-predictor-based structural completion provides a more stable balance between overall classification and risky-class identification.

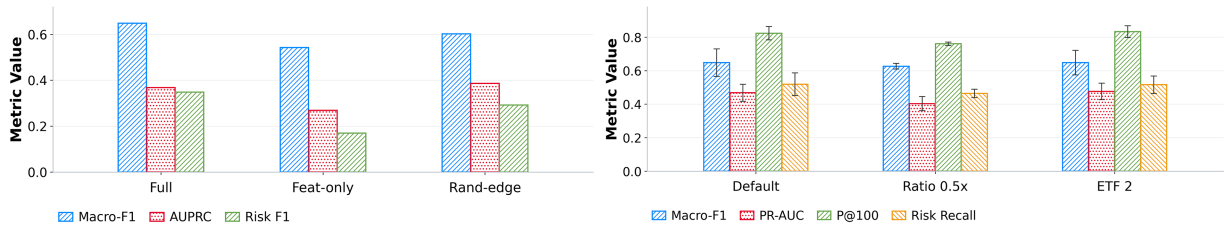

Figure 5: Additional analysis of AMLHunter. Left: structural completion schemes. Right: augmentation parameter sensitivity.

4.5 Hyperparameter Study

To examine the effects of augmentation strength and the candidate space for structural completion, we conduct a hyperparameter study with GraphSAGE as the classification head. The generation ratio controls the number of generated risky nodes, while `edge_top_factor` controls the size of the high-score candidate edge pool retained by the edge predictor before final edge completion.

The right part of Fig. 5 shows that reducing the generation ratio to 0.5× does not yield stable gains. Macro-F1, PR-AUC, ROC-AUC, P@100, and Risk Recall are all lower than those under the default setting, indicating that generating too few risky nodes weakens minority-class augmentation. By contrast, setting `edge_top_factor`=2 improves PR-AUC and P@100 while keeping Macro-F1 nearly unchanged. This suggests that a moderately smaller candidate edge pool can reduce structural noise from low-confidence candidate edges and improve the ranking quality of high-risk address candidates.

4.6 Comparison of Different Classification Heads

To evaluate whether the augmented graph works well with different graph neural network classifiers, we compare four classification heads on the same temporally consistent transaction semantic constrained augmented graph: GraphSAGE, R-GCN, HAN, and HGT. This experiment focuses on how different GNN classifiers adapt to the augmented graph. The main comparison of augmentation methods still uses GraphSAGE as the unified classification head.

Table 7 shows that, on the same augmented graph, GraphSAGE achieves better results on Macro-F1, PR-AUC, ROC-AUC, and P@100 than the more complex heterogeneous GNN heads. This suggests that the performance gain comes mainly from the improved quality of generated structures under temporally consistent transaction semantic constraints rather than from a more complex classifier. Since complex heterogeneous models may be more sensitive to sparse risky labels and noisy edges, GraphSAGE provides more stable generalization in this temporal split risk identification setting.

Table 7: Performance comparison of different graph neural network classification heads. Values are reported as mean (standard deviation). The best result is shown in bold, and the second-best result is underlined.

Head	Macro-F1	PR-AUC	ROC-AUC	P@50	P@100	R-Recall
GraphSAGE	**0.648 (0.082)**	**0.468 (0.051)**	**0.753 (0.038)**	**0.993 (0.012)**	**0.823 (0.040)**	**0.519 (0.068)**
R-GCN	0.630 (0.060)	0.347 (0.112)	0.678 (0.076)	0.787 (0.090)	0.573 (0.183)	0.511 (0.098)
HAN	0.522 (0.029)	0.315 (0.063)	0.665 (0.086)	0.873 (0.202)	0.613 (0.161)	**0.519 (0.098)**
HGT	0.608 (0.068)	0.387 (0.074)	0.723 (0.060)	0.940 (0.087)	0.693 (0.142)	0.459 (0.026)

4.7 Discussion

The experimental results show that AMLHunter mainly improves ranking-oriented metrics, including PR-AUC, P@100, and risky-class Recall. These metrics are more relevant to practical on-chain risk screening than accuracy alone, because analysts usually inspect only a limited number of high-risk candidates. The ablation studies further show that generated risky nodes are useful only when their structural context is carefully controlled. Feature-only generation cannot fully participate in graph message passing, while random or weakly constrained edge completion may introduce noisy paths. These results suggest that valid on-chain graph augmentation should consider not only minority-class feature generation, but also temporally valid and semantically meaningful generated connections.

We also evaluate the computational overhead of AMLHunter on the UpbitHack-50k subset using an Intel i7-11700 CPU, 16 GB memory, and an NVIDIA RTX 3060 12 GB GPU. The full detection-baseline group with nine methods and three random seeds takes 526.0 s. AMLHunter requires additional training-time cost because it includes teacher/encoder training, conditional diffusion training, risky-node generation, and structural completion. The augmentation stage takes 604.1 s for one seed and 1812.3 s for three sequential seeds, while the downstream GraphSAGE classifier takes about 40.0 s per seed after the augmented graph is saved. Therefore, AMLHunter should be viewed as an offline or near-offline forensic augmentation framework rather than a fully online transaction-level detector. In practice, the generative augmentation module can be executed periodically over historical time windows, while the trained classifier is used for rapid risky-address ranking after periodic model updates.

This study still has limitations. The current evaluation is conducted on the UpbitHack-50k dataset constructed from one major Ethereum security incident. Although this dataset provides a temporally ordered and highly imbalanced on-chain risk identification scenario, it may not cover all illicit fund propagation patterns, such as sustained phishing campaigns, mixer-assisted laundering, Ponzi-like fund circulation, or rapid smart contract exploit flows. Extending AMLHunter to cross-chain risk identification also requires addressing semantic interoperability, because different blockchains may have different account models, transaction formats, smart contract execution semantics, event logs, bridge transactions, and asset transfer mechanisms. Future work will evaluate AMLHunter on multi-incident and cross-chain datasets and introduce bridge-aware relation alignment, amount consistency, time-interval consistency, and multi-hop fund-path consistency.

5 Conclusion and Future Work

This paper proposes AMLHunter, a generative graph augmentation method for on-chain risky address identification under temporal split settings. AMLHunter constructs a heterogeneous transaction graph with address nodes, transaction nodes, address-transaction-address paths, and projected fund-flow edges. It then generates minority-class risky address features with a teacher-guided conditional diffusion model and completes generated-node structures through edge prediction, temporal consistency, fund-flow direction consistency, and local transaction semantic constraints. Generated-node quality gating and generated-edge quality filtering further reduce noisy synthetic samples and unreliable generated connections.

Experiments on the UpbitHack-50k temporal split dataset show that AMLHunter improves PR-AUC, P@100, and risky-class recall compared to representative detection baselines and graph augmentation baselines. The ablation studies further show that effective on-chain graph augmentation requires not only feature-space generation but also structurally meaningful and temporally valid generated connections. Future work will evaluate AMLHunter on larger cross-chain security incidents and incorporate richer constraints, such as amount consistency, time-interval consistency, and multi-hop fund-path consistency.

Acknowledgement: During the preparation of this manuscript, the authors used ChatGPT (GPT-5.5 Thinking, OpenAI) for language polishing and preliminary schematic figure drafting. All AI-assisted content was manually reviewed, revised, and validated by the authors. The AI tool was not used for scientific analysis, experimental design, data generation, or interpretation of the experimental results. The authors take full responsibility for the content of the manuscript.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: Conceptualization, Xiaolei Yin and Zihan Wang; methodology, Xiaolei Yin and Zihan Wang; software, Zihan Wang; writing—original draft, Xiaolei Yin and Zihan Wang; writing—review and editing, Sanfeng Zhang and Shouwei Li; supervision, Sanfeng Zhang and Shouwei Li. All authors reviewed and approved the final manuscript.

Availability of Data and Materials: The dataset cannot be publicly released due to data usage restrictions and security considerations. Processed statistics and experimental settings are provided in the manuscript. Additional information is available from the corresponding authors upon reasonable request.

Ethics Approval: Not applicable. This study analyzes blockchain transaction records and does not involve human participants, animal experiments, or private personal data.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Pourhabibi T, Ong KL, Kam BH, Boo YL. Fraud detection: a systematic literature review of graph-based anomaly detection approaches. Decis Support Syst. 2020;133:113303.
2. Weber M, Domeniconi G, Chen J, Weidele DKI, Bellei C, Robinson T, et al. Anti-money laundering in bitcoin: experimenting with graph convolutional networks for financial forensics. arXiv:1908.02591. 2019.
3. Kanezashi H, Suzumura T, Liu X, Hirofuchi T. Ethereum fraud detection with heterogeneous graph neural networks. arXiv:2203.12363. 2022.
4. Davis J, Goadrich M. The relationship between precision-recall and ROC curves. In: Proceedings of the 23rd International Conference on Machine Learning; 2006 Jun 25–29; Pittsburgh, PA, USA. p. 233–40.
5. Saito T, Rehmsmeier M. The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. PLoS One. 2015;10(3):e0118432. doi:10.1371/journal.pone.0118432.
6. Li S, Gou G, Liu C, Hou C, Li Z, Xiong G. TTAGN: temporal transaction aggregation graph network for Ethereum phishing scams detection. In: Proceedings of the ACM Web Conference 2022; 2022 Apr 25–29; Virtual. p. 661–9.
7. Zhang Z, He T, Chen K, Zhang B, Wang Q, Yuan L. Phishing node detection in Ethereum transaction network using graph convolutional networks. Appl Sci. 2023;13(11):6430. doi:10.3390/app13116430.
8. Yu T, Chen X, Xu Z, Xu J. MP-GCN: a phishing nodes detection approach via graph convolution network for Ethereum. Appl Sci. 2022;12(14):7294.
9. Wang L, Xu M, Cheng H. Phishing scams detection via temporal graph attention network in Ethereum. Inf Process Manag. 2023;60(4):103412. doi:10.1016/j.ipm.2023.103412.
10. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP. SMOTE: synthetic minority over-sampling technique. J Artif Intell Res. 2002;16:321–57.
11. Zhao T, Zhang X, Wang S. GraphSMOTE: imbalanced node classification on graphs with graph neural networks. In: Proceedings of the 14th ACM International Conference on Web Search and Data Mining; 2021 Mar 8–12; Virtual. p. 833–41.
12. Park J, Song J, Yang E. GraphENS: neighbor-aware ego network synthesis for class-imbalanced node classification. In: Proceedings of the Tenth International Conference on Learning Representations; 2022 Apr 25–29; Virtual.
13. Yang C, Xiao Y, Zhang Y, Sun Y, Han J. Heterogeneous network representation learning: a unified framework with survey and benchmark. IEEE Trans Knowl Data Eng. 2022;34(10):4854–73.

14. Schlichtkrull M, Kipf TN, Bloem P, van den Berg R, Titov I, Welling M. Modeling relational data with graph convolutional networks. In: Gangemi A, Navigli R, Vidal ME, Hitzler P, Troncy R, Hollink L, et al., editors. The semantic web. Cham, Switzerland: Springer International Publishing; 2018. p. 593–607.
15. Wang X, Ji H, Shi C, Wang B, Cui P, Yu PS, et al. Heterogeneous graph attention network. In: Proceedings of the World Wide Web Conference; 2019 May 13–19; San Francisco, CA, USA. p. 2022–32.
16. Hu Z, Dong Y, Wang K, Sun Y. Heterogeneous graph transformer. In: Proceedings of the Web Conference 2020; 2020 Apr 20–24; Taipei, Taiwan. p. 2704–10.
17. Xu D, Ruan C, Körpeoglu E, Kumar S, Achan K. Inductive representation learning on temporal graphs. arXiv:2002.07962. 2020.
18. Rossi E, Chamberlain B, Frasca F, Eynard D, Monti F, Bronstein M. Temporal graph networks for deep learning on dynamic graphs. arXiv:2006.10637. 2020.
19. Kazemi SM, Goel R, Jain K, Kobyzev I, Sethi A, Forsyth P, et al. Representation learning for dynamic graphs: a survey. J Mach Learn Res. 2020;21(70):1–73.
20. Kipf TN, Welling M. Semi-supervised classification with graph convolutional networks. arXiv:1609.02907. 2017.
21. Veličković P, Cucurull G, Casanova A, Romero A, Liò P, Bengio Y. Graph attention networks. In: Proceedings of the 6th International Conference on Learning Representations; 2018 Apr 30–May 3; Vancouver, BC, Canada.
22. Hamilton WL, Ying Z, Leskovec J. Inductive representation learning on large graphs. In: Proceedings of the 31st International Conference on Neural Information Processing Systems; 2017 Dec 4–9; Long Beach, CA, USA. p. 1025–35.
23. Kumar S, Zhang X, Leskovec J. Predicting dynamic embedding trajectory in temporal interaction networks. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining; 2019 Aug 4–8; Anchorage, AK, USA. p. 1269–78.
24. Trivedi R, Farajtabar M, Biswal P, Zha H. DyRep: learning representations over dynamic graphs. In: Proceedings of the Seventh International Conference on Learning Representations; 2019 May 6–9; New Orleans, LA, USA.
25. Pareja A, Domeniconi G, Chen J, Ma T, Suzumura T, Kanezashi H, et al. EvolveGCN: evolving graph convolutional networks for dynamic graphs. Proc AAAI Conf Artif Intell. 2020;34(4):5363–70.
26. Wang D, Lin J, Cui P, Jia Q, Wang Z, Fang Y, et al. A semi-supervised graph attentive network for financial fraud detection. In: Proceedings of the 2019 IEEE International Conference on Data Mining (ICDM); 2019 Nov 8–11; Beijing, China. p. 598–607.
27. Dou Y, Liu Z, Sun L, Deng Y, Peng H, Yu PS. Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management; 2020 Oct 19–23; Virtual. p. 315–24.
28. Liu Y, Ao X, Qin Z, Chi J, Feng J, Yang H, et al. Pick and choose: a GNN-based imbalanced learning approach for fraud detection. In: Proceedings of the Web Conference 2021; 2021 Apr 19–23; Ljubljana, Slovenia. p. 3168–77.
29. Shi F, Cao Y, Shang Y, Zhou C, Wu J, Zhang C. H2-FDetector: a GNN-based fraud detector with homophilic and heterophilic connections. In: Proceedings of the ACM Web Conference 2022; 2022 Apr 25–29; Virtual. p. 1486–94.
30. Lin TY, Goyal P, Girshick R, He K, Dollár P. Focal loss for dense object detection. In: 2017 IEEE International Conference on Computer Vision (ICCV); 2017 Oct 22–29; Venice, Italy. p. 2980–8.
31. Li WZ, Wang CD, Xiong H, Lai JH. GraphSHA: synthesizing harder samples for class-imbalanced node classification. In: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining; 2023 Aug 6–10; Long Beach, CA, USA. p. 1328–40.
32. Wu L, Xia J, Gao Z, Lin H, Tan C, Li SZ. GraphMixup: improving class-imbalanced node classification by reinforcement mixup and self-supervised context prediction. In: Amini MR, Canu S, Fischer A, Guns T, Kralj Novak P, Tsoumakas G, editors. Machine learning and knowledge discovery in databases. Cham, Switzerland: Springer Nature; 2023. p. 519–35.
33. Qu L, Zhu H, Zheng R, Shi Y, Yin H. ImGAGN: imbalanced network embedding via generative adversarial graph networks. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining; 2021 Aug 14–18; Virtual. p. 1390–8.

34. Song J, Park J, Yang E. TAM: topology-aware margin loss for class-imbalanced node classification. In: Proceedings of the 39th International Conference on Machine Learning; 2022 Jul 17–23; Baltimore, MD, USA. p. 20369–83.
35. Zeng L, Li L, Gao Z, Zhao P, Li J. ImGCL: revisiting graph contrastive learning on imbalanced node classification. Proc AAAI Conf Artif Intell. 2023;37(9):11138–46.
36. Ho J, Jain A, Abbeel P. Denoising diffusion probabilistic models. In: Proceedings of the 34th International Conference on Neural Information Processing System; 2020 Dec 6–12; Vancouver, BC, Canada. p. 6840–51.
37. Song Y, Sohl-Dickstein J, Kingma DP, Kumar A, Ermon S, Poole B. Score-based generative modeling through stochastic differential equations. arXiv:2011.13456. 2021.
38. Zhang M, Qamar M, Kang T, Jung Y, Zhang C, Bae SH, et al. A survey on graph diffusion models: generative AI in science for molecule, protein and material. arXiv:2304.01565. 2023.
39. Kipf TN, Welling M. Variational graph auto-encoders. arXiv:1611.07308. 2016.
40. Boughorbel S, Jarray F, El-Anbari M. Optimal classifier for imbalanced data using matthews correlation coefficient metric. PLoS One. 2017;12(6):e0177678. doi:10.1371/journal.pone.0177678.
41. Chen T, Guestrin C. XGBoost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2016 Aug 13–17; San Francisco, CA, USA. p. 785–94.
42. Grover A, Leskovec J. node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2016 Aug 13–17; San Francisco, CA, USA. p. 855–64.