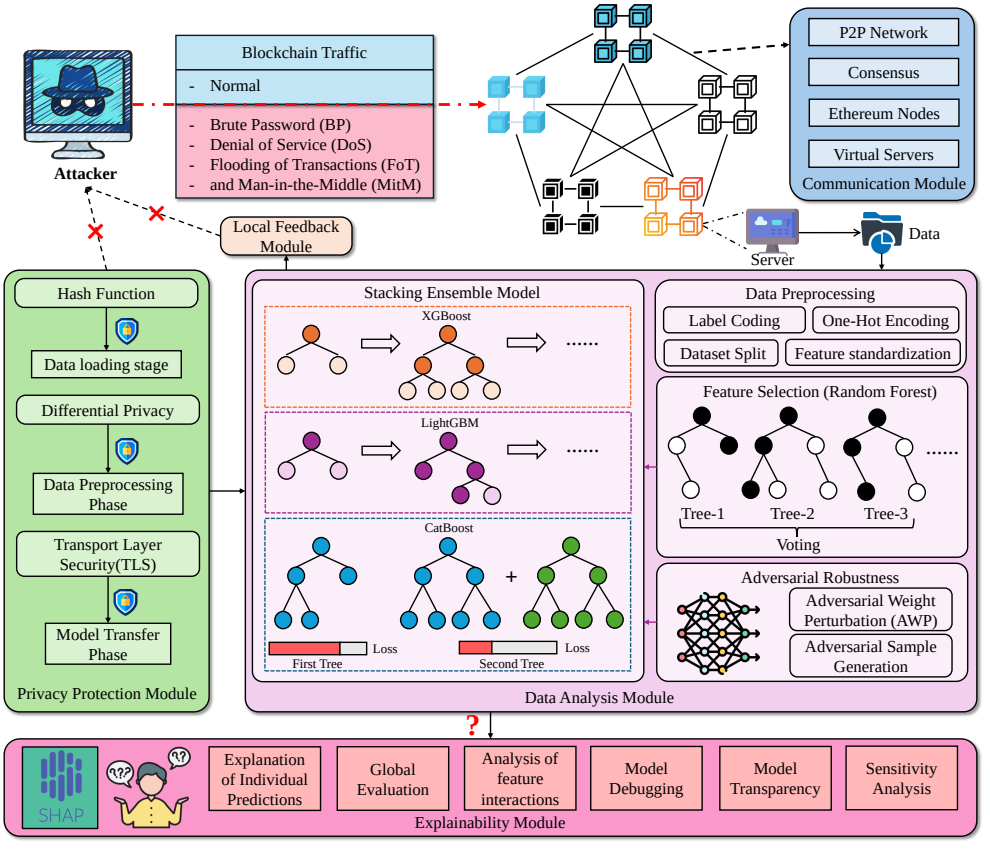
Graphical Abstract

Intrusion Detection in Blockchain Networks: Explainable Stacking Model with Adversarial Deep Learning and Privacy-Preserving Techniques



Highlights

Intrusion Detection in Blockchain Networks: Explainable Stacking Model with Adversarial Deep Learning and Privacy-Preserving Techniques

- Intrusion detection.
- Blockchain Networks.
- Give reliable explanations.

Intrusion Detection in Blockchain Networks: Explainable Stacking Model with Adversarial Deep Learning and Privacy-Preserving Techniques

ARTICLE INFO

Keywords:
Intrusion detection
Blockchain networks
Explainable artificial intelligence (XAI)
Black-box model

ABSTRACT

Blockchain technology has become a cornerstone of secure and decentralized systems, offering transparency, resilience, and trust across diverse applications. However, the decentralized and dynamic nature of blockchain networks poses significant challenges for intrusion detection systems (IDSs). Ensuring robustness against adversarial attacks and preserving sensitive data privacy remain critical challenges, as adversaries can exploit vulnerabilities in machine learning models, while blockchain traffic often contains sensitive information such as cryptographic keys and transaction payloads. In addition, the lack of explainability in existing IDS models further exacerbates these issues, particularly in sensitive domains, where users require clear and interpretable anomaly detection results to establish trust and ensure effective decision-making. To this end, we propose the Explainable Stacking Intrusion Detection for Blockchain Networks (ES-IDB), a novel IDS framework that integrates high detection accuracy with explainability. The ES-IDB framework employs a stacking ensemble model, combining CatBoost, XGBoost, and LightGBM, with Logistic Regression as the meta-classifier to effectively detect a wide range of blockchain-specific attacks. Then, adversarial training using the Fast Gradient Sign Method (FGSM) and privacy protection mechanisms are incorporated to enhance the robustness against malicious inputs and attackers. Furthermore, SHAP is used to provide clear and actionable insights into model predictions, enabling explainable anomaly detection and fostering trust in the system. Extensive experiments conducted on a blockchain-specific dataset demonstrate that ES-IDB achieves state-of-the-art performance in terms of detection accuracy, resilience, privacy preservation, and explainability.

1. Introduction

Blockchain technology has emerged as a disruptive innovation within the domain of distributed systems, fundamentally reshaping the management of data and transactions in decentralized environments [1, 2, 3]. Functioning as a distributed ledger, blockchain ensures data immutability, transparency, and security through the integration of cryptographic techniques and consensus protocols. Blockchain eliminates the need for an intermediary and establishes trust between participants, thus providing a robust infrastructure for applications as diverse as cryptocurrencies [4], supply chain management [5], healthcare systems [6], and financial services [7]. The decentralized architecture of blockchain enhances operational efficiency and mitigates single points of failure, providing a resilient and fault-tolerant solution for critical systems.

Blockchain networks rely on consensus mechanisms to validate and append new transactions to the ledger, including Proof of Work (PoW), Proof of Stake (PoS), and Byzantine Fault Tolerance (BFT) [8, 9]. These mechanisms are supported by cryptographic primitives, such as hashing and public-key infrastructure (PKI), which ensures data integrity and secure communication among participants. Moreover, smart contracts, which are self-executing programs encoded on the blockchain, enable automatic and trustless execution of protocols, thereby significantly expanding the practical applications of blockchain technology.

While blockchain offers significant advantages in terms of security and resilience, it also introduces unique challenges for network defense mechanisms, particularly in the context of Intrusion Detection Systems (IDSs). Traditionally, IDSs have been extensively studied and deployed to mitigate network security threats in domains such as Cloud Computing [10], Industrial Internet of Things (IIoT) [11] and Vehicular Ad Hoc Networks (VANETs) [12], which leverages data-driven techniques to detect anomalies and malicious activities. However, the decentralized architecture, peer-to-peer communication, and cryptographic protocols inherent to blockchain networks require a fundamental rethinking of intrusion detection strategies. Existing IDS solutions that are often designed for centralized

ORCID(s):

or hierarchical systems struggle to adapt to the highly dynamic and distributed nature of blockchain environments. While current attacks predominantly target blockchain-based cryptocurrencies, such as digital asset theft and smart contract exploitation, the potential applications of blockchain technology extend far beyond this domain. In the future, blockchain is expected to play a transformative role in diverse fields, including supply chain management [13], identity verification [14], healthcare records [15], electronic voting [16], and intellectual property protection [17]. Consequently, there is an urgent need to develop universal intrusion detection mechanisms tailored for blockchain systems. Furthermore, the increasing prevalence of adversarial machine learning attacks exacerbates these challenges, compromising the robustness of traditional IDS models and increasing the risk of misclassification for sophisticated and evolving threat [18]. Another critical challenge in blockchain-based IDS lies in ensuring privacy during model training and operation. Blockchain traffic data often contains sensitive information, such as cryptographic keys, transaction payloads, and user metadata, which can be vulnerable to inference attacks when machine learning models are employed. Although techniques like differential privacy (DP) have shown potential for safeguarding sensitive data in machine learning applications [19], their integration into blockchain-based IDS remains at an early stage.

More importantly, while high detection accuracy is essential, the lack of explainability in many IDS models significantly limits their practical applicability [20]. In sensitive domains such as finance [21] and healthcare [22, 23], end-users often require clear and understandable explanations for anomaly detection results to establish trust and utilize the system effectively. However, current blockchain IDS models face considerable challenges in delivering transparent and explainable detection outcomes [24, 25, 26], which hinders their widespread adoption in real-world applications. Addressing these challenges is imperative to improve user trust and system usability.

To address these challenges, it is imperative to develop an IDS framework specifically tailored for blockchain networks, incorporating robustness against adversarial attacks, privacy-preserving mechanisms, and explainability. Such a framework should: 1) Detect a wide range of blockchain-specific attacks while maintaining adaptability to the decentralized architecture. 2) Ensure the security, privacy, and robustness of sensitive blockchain data during the entire model lifecycle. 3) Provide actionable and explainable insights into anomalies to foster trust and facilitate timely countermeasures.

In this paper, we propose the **E**xplainable **S**tacking **I**ntrusion **D**etection for **B**lockchain Networks (ES-IDB), a novel IDS framework designed to address the unique challenges of explainability and security. The main contributions of this work are as follows:

- We design an advanced IDS tailored for blockchain networks that is capable of detecting a diverse set of attacks. The system leverages a stacking ensemble model that integrates CatBoost, XGBoost, and LightGBM, with Logistic Regression as the meta-classifier to achieve high detection accuracy.
- We incorporate adversarial training techniques using Fast Gradient Sign Method (FGSM) to counter adversarial attacks that exploit vulnerabilities in machine learning models. This enhances the resilience to perturbed inputs, ensuring reliable detection in real-world scenarios.
- We integrate privacy-preserving techologies into the data preprocessing and model training pipeline to safeguard sensitive blockchain traffic data. Both Gaussian and Laplacian noise mechanisms are employed to obscure sensitive information while retaining the utility of the dataset.
- Extensive experiments are conducted by using a blockchain-specific dataset to validate the effectiveness of the proposed model, including both normal and attack traffic. SHAP is utilized to analyze and visualize feature contributions to model predictions, which allows for explainable root-cause analysis of anomalies, promoting trust and transparency in the IDS.

The remainder of this paper is organized as follows. Section 2 reviews related works. Section 3 describes the system model. Section 4 presents the framework of ES-IDB. Section 5 evaluates performance and explainability. Finally, Section 6 concludes the paper.

2. Related Works

In recent years, network intrusion detection systems (NIDS) have garnered increasing attention from researchers due to the rising complexity and frequency of cyberattacks. In terms of traditional NIDS, some classical datasets have been well studied. Alheeti and Mc Donald-Maier [27] proposed an intelligent protection mechanism using proportional

overlap scores and back-propagation neural networks to enhance intrusion detection for Denial of Service (DoS) attacks on the Kyoto 2006+ dataset. Liu [28] presented a network intrusion detection system based on the Lenet-5 convolutional neural network using the KDD99 dataset. Gao et al. [29] introduced a distributed DDoS intrusion detection system using big data technology, integrating Spark for real-time data processing, HDFS for storage, and a Random Forest algorithm for traffic classification on NSL-KDD and UNSW-NB15. Injadat et al.[30] developed a multi-stage optimized machine learning-NIDS that minimizes computational complexity on the CICIDS 2017 and UNSW-NB15 datasets.

In addition to the traditional networks mentioned above, there is also some IDS research on blockchain networks [31]. In [32], Kumar et al. introduced a novel security framework for blockchain-IoT systems on the BoT-IoT dataset with only 10 feature, integrating random forest and XGBoost for autonomous DDoS attack detection. Alkadi et al. [33] proposed a deep blockchain framework combining BiLSTM-based intrusion detection and Ethereum-based privacy mechanisms on UNSW-NB15 and BoT-IoT datasets. While these studies advanced IDS for blockchain networks, they were not experimented on datasets exclusive to blockchain networks. Therefore, some studies have emerged that train models based on artificially simulated blockchain data. Kim et al. [34] developed a security mechanism for blockchain networks, utilizing semi-supervised AutoEncoder-based anomaly detection on network traffic data from a public Bitcoin node. Liu and Yin [35] preseented an LSTM-CGAN framework to generate high-quality LDDoS adversarial samples for blockchain-based wireless network detection, which leveraged time-series modeling and conditional GANs to mimic LDDoS attack behaviors. Cao et al. [36] proposed a blockchain-based detection scheme for Link Flooding Attacks (LFA), utilizing Ethereum to record and share traceroute information for accurate and timely detection. Ramanan et al. [37] proposed a blockchain-based decentralized framework for detecting coordinated replay attacks in large-scale power systems. Hu et al. [38] presented a blockchain-based collaborative intrusion detection (CID) approach for multimicrogrid (MMG) systems in smart grids, which employed consensus mechanism to eliminate the need for a central authority. Recently, there has been a shift in research focus to constructing real blockchain traffic datasets. Khoa et al. [39] proposed blockchain network attack traffic dataset (BNaT) and jonjointly suggested a distributed collaborative learning approach for intrusion detection in blockchain networks. This is the first study based on a real blockchain network intrusion detection dataset, which aims to systematically explore security threat detection mechanisms.

However, now that real data on blockchain cyber-attacks is available, there is still a research gap in terms of explainability. There have been some studies on the explainability of traditional network intrusion detection models, for example, Sharma et al. [40] proposed an explainable reinforcement learning model for IoT networks. Lundberg et al. [41] proposed a visualization-based deep explanation model for vehicular networks to improve the trust of the system. Until now, there are still no explainable models for real blockchain network attack datasets.

3. System Model

3.1. Communication Module

The Communication Module (CM) forms the core of interaction within the blockchain-based P2P network, facilitating efficient and secure data exchange between Ethereum nodes and virtual servers. It employs secure communication protocols, such as Transport Layer Security (TLS), to safeguard data in transit and incorporates advanced measures to address potential network threats. The CM collects real-time network activity data, reflecting the state and operations of the blockchain environment. This data is forwarded to the Privacy Protection Module (PPM) for analysis, assisting in the detection of patterns indicative of abnormal activities. Additionally, the CM integrates feedback from the Local Feedback Module (LFU) and propagates security alerts across the network to enable prompt responses to identified threats.

3.2. Privacy Protection Module

The PPM employs hash function (HF), DP, and TLS to ensure comprehensive data security and privacy within the system. Upon receiving blockchain communication data from the CM, the PPM applies TLS protocols to encrypt the data during transmission. This prevents potential attacks, which ensures that the data remains secure as it flows through the system. HFs are applied to specific private network information within the blockchain data, ensuring that this information cannot be easily decoded or accessed by unauthorized parties. DP is employed to add controlled random noise to the data, obscuring sensitive individual details while maintaining the overall utility of the dataset for analysis. This technique ensures compliance with privacy requirements, protecting sensitive information from inference attacks during model training and analysis. The combination of these techniques ensures that the data is both secure

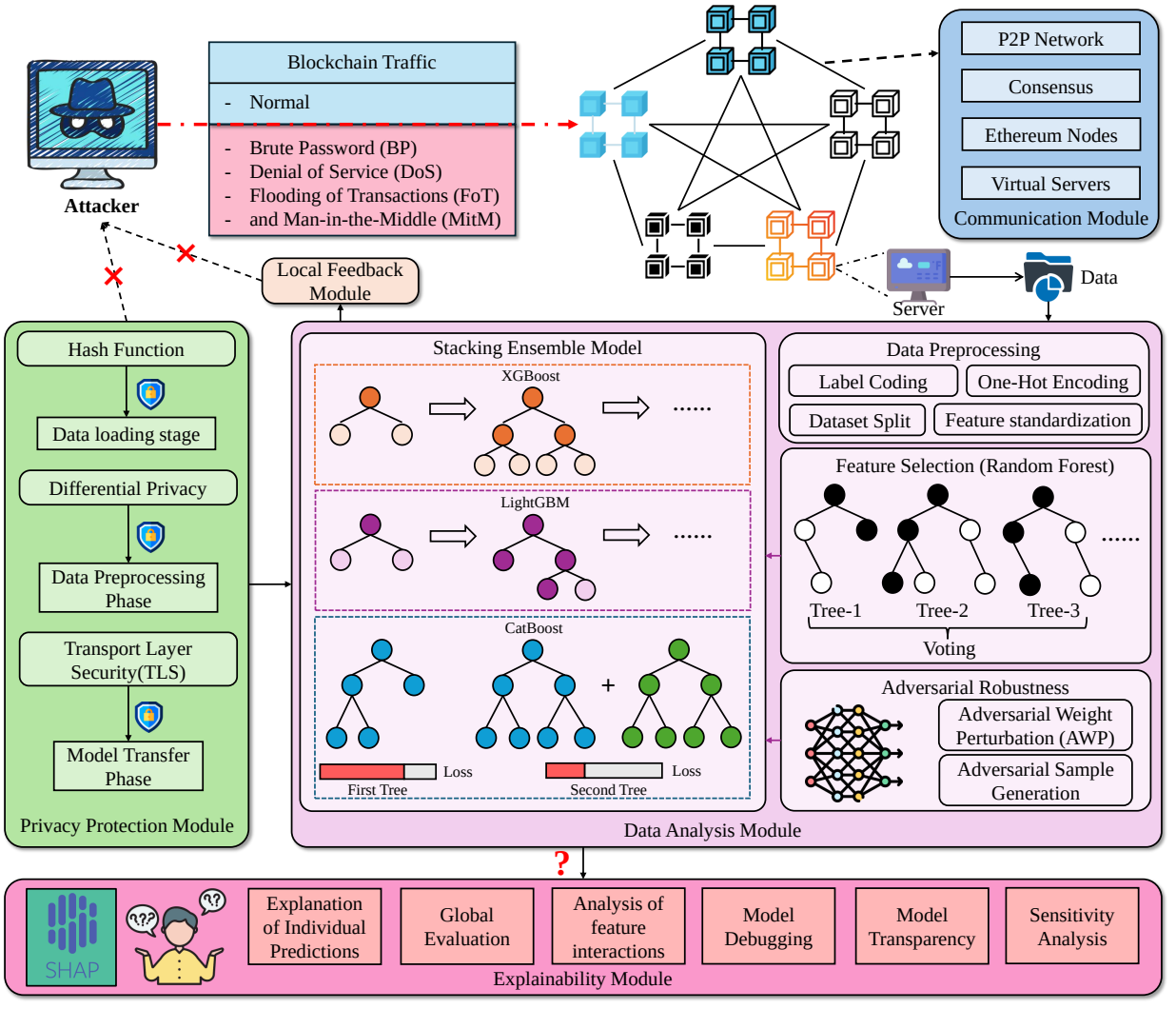


Figure 1: The proposed system model for IDS in blockchain networks.

and privacy-compliant, enabling accurate and trustworthy results. Once processed, the data is utilized by the machine learning models, such as CatBoost, XGBoost, and LightGBM, integrated within the system.

3.3. Data Analysis Module

The Data Analysis Module (DAM) processes blockchain traffic data through a systematic and robust pipeline, ensuring accurate classification and analysis. Initially, raw data undergoes preprocessing, including label encoding, one-hot encoding, standardization, and stratified data splitting to prepare it for feature selection and model training. Key features are identified using Random Forest-based importance ranking, retaining the most significant features to reduce noise and computational overhead. To enhance resilience, adversarial training is integrated using the Fast Gradient Sign Method (FGSM), which generates perturbed datasets for robust neural network training. Subsequently, a stacking ensemble model combines the predictive power of multiple classifiers, such as CatBoost, XGBoost, and LightGBM, with Logistic Regression as the meta-classifier. Hyperparameter optimization via Bayesian search further refines the ensemble, ensuring optimal performance.

3.4. Local Feedback Module

The primary function of the LFM is to process classification results y received from the DAM. When the results y contain entries marked as -1, the module identifies the associated communication data as anomalous traffic. In

response, it generates anomaly feedback results and submits them to the CM for further dissemination. Conversely, if the results y contain only 1, the module marks the communication data as normal traffic, requiring no further action or feedback. Before receiving new data, the LF, performs an update operation, clearing residual data from prior analyses to ensure accurate and unbiased processing of subsequent results.

3.5. Explainability Module

The Explainability Module (EM) enhances the explainability of predictions from the Data Analysis Module by utilizing SHapley Additive exPlanations (SHAP) to quantify the contribution of each feature to the outputs. For every classification, the module computes SHAP values, illustrating the positive or negative influence of individual features. This explainability mechanism supports detailed visualization, such as force plots and summary plots, which provide intuitive insights into both normal and anomalous blockchain traffic classifications. By seamlessly integrating with the LFM and DAM, the EM facilitates root-cause analysis of anomalies while promoting transparency and trust in the system's machine learning algorithms.

3.6. Threat Model

We provide a detailed description of the threat landscape for the blockchain network based on identified attack types. Excluding the benign class (Normal), the dataset includes several malicious attack categories: Brute Password (BP), DoS, Flooding of Transactions (FoT), and Man-in-the-Middle (MitM). These attack classes represent significant threats to the security, availability, and integrity of blockchain systems.

3.6.1. BP Attack

The BP attack is derived from traditional brute-force attacks, where hackers attempt to compromise blockchain user accounts by systematically guessing passwords or keys. Once successful, attackers gain unauthorized access to users' wallets, enabling them to steal digital assets. This type of attack often leverages computational power to automate password-guessing attempts, exploiting weak or poorly managed passwords.

In the blockchain context, BP attacks are particularly dangerous as they directly target users' private keys, which serve as the sole means of accessing and controlling digital assets. The decentralized nature of blockchain leaves users solely responsible for password security, making them prime targets for such attacks.

3.6.2. DoS Attack

The DoS attack is a well-known vector in cybersecurity and is prevalent in blockchain networks. In this attack, adversaries flood a blockchain node or network with an overwhelming number of requests, exhausting its computational or memory resources. The consequences of a successful DoS attack in a blockchain environment include: 1) Disruption of node operations, reducing the ability of nodes to process and validate legitimate transactions. 2) Increased transaction latency, as legitimate users face delays due to congestion. 3) Potential partitioning of the blockchain network, jeopardizing consensus mechanisms.

Blockchain systems are particularly vulnerable to DoS attacks due to their reliance on distributed nodes, where a bottleneck in any critical node can disrupt the entire network.

3.6.3. FoT Attack

The FoT attack is specific to Proof-of-Work (PoW) blockchain systems and exploits their computational resource constraints. In this attack, adversaries spam the blockchain network with null or meaningless transactions to: 1) Delay transaction processing for legitimate users by overwhelming miners or validators with unnecessary data. 2) Exhaust network bandwidth and memory resources, creating bottlenecks in transaction propagation.

By deliberately saturating the blockchain with fake transactions, attackers can degrade network performance, disrupt consensus, and increase transaction fees for legitimate participants. This type of attack underscores the inherent vulnerabilities of PoW systems under high transaction loads.

3.6.4. MitM Attack

The MitM attack involves an attacker secretly positioning themselves between two communicating parties in the blockchain network. By intercepting and potentially modifying information exchanged between parties, the attacker can: 1) Steal sensitive information, such as cryptographic keys or transaction data. 2) Alter transaction payloads to commit fraud or redirect assets. 3) Disrupt communication between nodes, potentially affecting block propagation and synchronization.

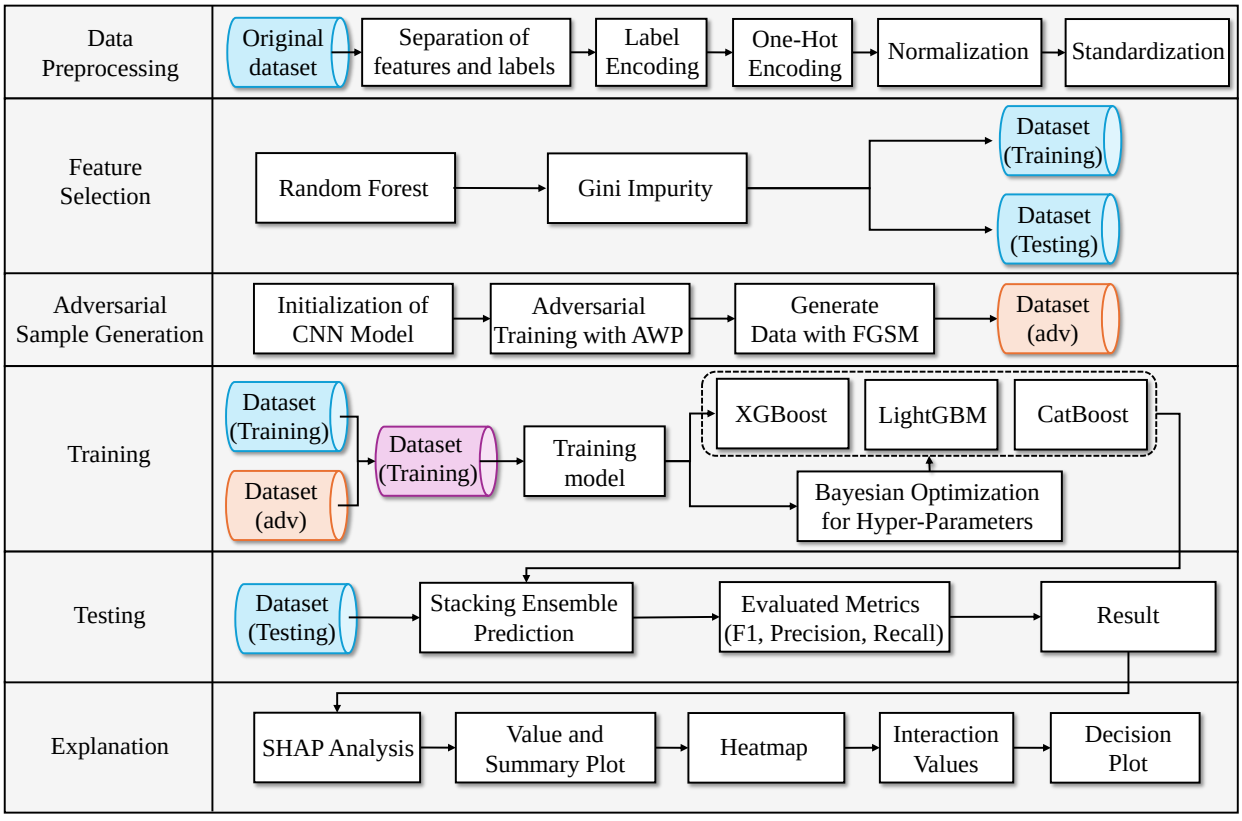


Figure 2: The framework of ES-IDB.

This attack exploits weaknesses in network communication protocols, such as the lack of strong encryption or authentication mechanisms. MitM attacks are particularly damaging in permissionless blockchain networks, where nodes often lack direct trust relationships.

The dynamic and evolving nature of blockchain systems necessitates continuous monitoring, proactive defense mechanisms, and advancements in protocol design to mitigate these threats effectively. To this end, we introduce the ES-IDB model, aimed to improve the explainability and security of blockchain networks. The detailed mechanisms of the ES-IDB model will be elaborated in Section 3.

4. The framework of ES-IDB

4.1. Data Loading and Preprocessing

In the data loading and preprocessing stage, the dataset $\mathbf{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ is loaded from an external data source, where $\mathbf{x}_i \in \mathbb{R}^m$ represents the input features and $y_i \in \mathcal{Y}$ represents the corresponding target labels, with $\mathcal{Y}$ being the label space. The target variable y_i is then encoded to transform it from the categorical label space $\mathcal{Y}$ into discrete integer labels, denoted as $\hat{y}_i = \text{Encode}(y_i)$, to facilitate subsequent model training tasks.

For the feature matrix $\mathbf{X} \in \mathbb{R}^{n\times m}$, categorical variables are processed using one-hot encoding. Assuming the total number of categorical columns is c, the transformed feature matrix after encoding has a dimension of $\mathbf{X}' \in \mathbb{R}^{n\times(m+c)}$. This ensures that non-numeric features are represented as binary feature vectors, fully converting the input features into numerical format.

Next, the dataset is split into training and testing sets. Let the splitting ratio be $\rho \in (0, 1)$. The training set contains $\lfloor \rho n \rfloor$ samples, while the testing set contains $n - \lfloor \rho n \rfloor$ samples. Stratified sampling is employed during this splitting process to ensure that the class distribution of the target variable y_i is consistent between the training and testing sets.

Finally, all numerical features are standardized to have zero mean and unit variance. For each feature $\mathbf{x}_j$ (the j-th column), standardization is performed as follows:

$$\mathbf{x}'_j = \frac{\mathbf{x}_j - \mu_j}{\sigma_j}, \quad \text{where } \mu_j = \frac{1}{n}\sum_{i=1}^{n} x_{ij},\ \sigma_j = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{ij} - \mu_j)^2} \tag{1}$$

The standardized feature matrix is denoted as $\mathbf{X}'' \in \mathbb{R}^{n\times(m+c)}$, which is more suitable for gradient-based optimization processes.

4.2. Feature Selection

Feature selection is an essential step in the algorithm to optimize input features by quantitatively analyzing their contributions to the model. In our proposed ES-IDB, Random Forest is used as the tool for feature selection. Random Forest is an ensemble-based algorithm consisting of multiple decision trees. During training, a subset of features is randomly selected for splitting at each decision tree node. The importance of each feature is quantified by evaluating its contribution to the reduction of impurity during the decision-making process. The feature importance (FI_j) for feature j is computed as:

$$\mathrm{FI}_j = \sum_{t\in T} \Delta\mathrm{Im}(t, j) \tag{2}$$

where T is the set of all decision trees in the forest, t represents a specific split node in the decision tree, and $\Delta\mathrm{Im}(t, j)$ is the reduction in impurity at the node t caused by splitting on feature j.

The impurity at the node t, denoted as $\mathrm{Im}(t)$, is measured using either the Gini index in ES-IDB. For the Gini index, the impurity is defined as:

$$\mathrm{Gn}(t) = 1 - \sum_{i=1}^{k} p_i^2 \tag{3}$$

where p_i represents the proportion of samples belonging to class i at node t. The contribution of feature j to the impurity reduction at the node t is calculated as:

$$\Delta\mathrm{Im}(t, j) = \mathrm{Im}(t) - \left(w_L \cdot \mathrm{Im}(t_L) + w_R \cdot \mathrm{Im}(t_R)\right) \tag{4}$$

where t_L and t_R are the left and right child nodes resulting from splitting on feature j. w_L and w_R represent the proportions of samples in t_L and t_R, respectively.

After computing FI_j for all features, the feature importance vector is obtained by:

$$\mathbf{FI} = [\mathrm{FI}_1, \mathrm{FI}_2, \ldots, \mathrm{FI}_d] \tag{5}$$

The features are then sorted by their importance scores in descending order with $\mathbf{r} = [r_1, r_2, \ldots, r_d]$, where r_i represents the index of the i-th most important feature. To reduce the dimensionality of features, the top m features are selected, where $m' = \lfloor \alpha \cdot d \rfloor$ and α is the selection proportion. The final selected feature matrix is $\mathbf{X}_{\mathrm{sel}}$, where $\mathbf{X}_{\mathrm{sel}} \in \mathbb{R}^{n\times m'}$. The reduced feature matrix retains the most significant features, providing a more efficient and concise input for the subsequent model training.

4.3. Deep Learning Framework with Adversarial Robustness

4.3.1. Initialization of CNN Model

The ES-IDB employs a CNN as the base model for adversarial training. Let the input feature matrix be $\mathbf{X}_{\mathrm{sel}}$, which is transformed into a three-dimensional tensor $\mathbf{X}_{\mathrm{in}} \in \mathbb{R}^{n\times 1\times d}$ by adding a channel dimension to fit the one-dimensional convolutional layer. The CNN architecture consists of two 1D convolutional layers, each with kernel size $k = 3$, stride $s = 1$, and output channels of 32 and 64, respectively. For the i-th convolutional layer, the feature map $\mathbf{F}^{(i)} \in \mathbb{R}^{n\times c_i\times d}$ is computed as:

$$\mathbf{F}^{(i)} = \mathrm{ReLU}(\mathbf{W}^{(i)} * \mathbf{X}_{\mathrm{in}} + \mathbf{b}^{(i)}) \tag{6}$$

where $\mathbf{W}^{(i)} \in \mathbb{R}^{c_i \times c_{i-1} \times k}$ and $\mathbf{b}^{(i)} \in \mathbb{R}^{c_i}$ are the convolution kernel and bias, respectively, and $*$ denotes the convolution operator. ReLU$(\cdot)$ is the Rectified Linear Unit activation function.

The output tensor is flattened into $\mathbf{F}_{\text{fla}} \in \mathbb{R}^{n \times (c_2 \cdot d)}$, which is passed through two fully connected layers. The first layer maps it to a 128-dimensional space, and the second layer outputs logits $\mathbf{z} \in \mathbb{R}^C$, where C is the number of classes. Dropout with a rate $p = 0.3$ is applied to reduce overfitting. The final computation of logits is as follows:

$$\mathbf{z} = \mathbf{W}^{\text{fc2}} \cdot \text{ReLU}(\mathbf{W}^{\text{fc1}} \cdot \mathbf{F}_{\text{fla}} + \mathbf{b}^{\text{fc1}}) + \mathbf{b}^{\text{fc2}} \tag{7}$$

where $\mathbf{W}^{\text{fc1}}, \mathbf{W}^{\text{fc2}}$ and $\mathbf{b}^{\text{fc1}}, \mathbf{b}^{\text{fc2}}$ are the weights and biases of the fully connected layers.

4.3.2. Adversarial Training with AWP

Adversarial Weight Perturbation (AWP) is incorporated during training to enhance the robustness of the model against adversarial attacks. Let $\mathcal{L}(\mathbf{W}, \mathbf{X}_{\text{sel}}, \mathbf{y})$ denote the loss function, where $\mathbf{W}$ are the model weights, $\mathbf{X}_{\text{sel}}$ is the input, and $\mathbf{y}$ is the target labels. Then, the standard gradient loss is $\nabla_{\mathbf{W}} \mathcal{L}(\mathbf{W}, \mathbf{X}_{\text{sel}}, \mathbf{y})$, where the loss function $\mathcal{L}$ is defined as cross-entropy:

$$\mathcal{L} = -\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{C} y_{ij} \log \hat{y}_{ij} \tag{8}$$

with y_{ij} representing the one-hot encoded true label and $\hat{y}_{ij}$ being the predicted probability:

$$\hat{y}_{ij} = \frac{\exp(z_{ij})}{\sum_{k=1}^{C} \exp(z_{ik})} \tag{9}$$

4.3.3. Adversarial Example Generation

To generate adversarial examples for training, we use FGSM to perturb the input data. The adversarial examples $\mathbf{X}_{\text{adv}}$ are generated by adding the perturbation δ to the original input $\mathbf{X}_{\text{adv}}$:

$$\delta = \alpha \cdot \text{sign}(\nabla_{\mathbf{X}_{\text{adv}}} \mathcal{L}(\mathbf{W}, \mathbf{X}_{\text{adv}}, \mathbf{y})) \tag{10}$$

$$\mathbf{X}_{\text{adv}} = \mathbf{X}_{\text{adv}} + \delta \tag{11}$$

where α is the perturbation magnitude, and sign$(\cdot)$ denotes the sign function. This perturbation is designed to maximize the loss and thus fool the model during training.

Finally, the standard and adversarial losses are combined to compute the total gradient:

$$\nabla_{\mathbf{W}} \mathcal{L}_{\text{total}} = \nabla_{\mathbf{W}} \mathcal{L} + \nabla_{\mathbf{W}} \mathcal{L}_{\text{adv}} \tag{12}$$

The model weights are updated by:

$$\mathbf{W} \leftarrow \mathbf{W} - \eta \cdot \nabla_{\mathbf{W}} \mathcal{L}_{\text{total}} \tag{13}$$

where η is the learning rate.

4.4. Algorithms Used by ES-IDB

4.4.1. Stacking Ensemble Model

In this paper, a stacking classifier is employed to implement ensemble learning by combining multiple base learners. Three distinct base learners are selected: XGBoost, LightGBM, and CatBoost. XGBoost is a tree-based gradient boosting model that iteratively optimizes multiple weak classifiers using weighted loss functions. LightGBM, another gradient boosting framework, adopts histogram-based optimization and an efficient leaf-wise splitting strategy, demonstrating significant efficiency in handling large-scale datasets and high-dimensional sparse features. CatBoost, based on symmetric tree construction, is designed to handle categorical features and reduces target leakage through a unique sequentially independent processing method.

Each base learner is independently trained to extract feature representations from the input data. During training, k-fold cross-validation is applied to the dataset for each base learner to prevent overfitting and ensure robust model training. The outputs of each base learner, whether in the form of probability distributions of the classes or predicted class labels, are stored and later used as inputs for the meta-learner.

The meta-learner is a logistic regression model that utilizes the outputs of the base learners as input features. It further trains on these outputs to achieve the final classification results. The training of the meta-learner also employs k-fold cross-validation to ensure stability and generalizability. As a linear classifier, logistic regression combines the outputs of the base learners by assigning optimized weights, resulting in the final predictions.

4.4.2. Bayesian Optimization for Hyper-Parameters

During the training process of stacking classifier, Bayesian optimization is used to fine-tune the hyperparameters of each base learner. For XGBoost, hyperparameters include the maximum tree depth, the number of weak classifiers, the learning rate, the feature sampling ratio, and the data sampling ratio. The hyperparameters of LightGBM include the learning rate, the number of decision trees, the number of leaf nodes, and the feature splitting thresholds. The hyperparameters of CatBoost consist of tree depth, the number of iterations, and the learning rate. Bayesian optimization progressively identifies the optimal combination of hyperparameters by maximizing the scoring function evaluated through cross-validation, and the optimized base learners are subsequently used in the stacking classifier.

After training, the stacking classifier utilizes the meta-learner to generate the final predictions on the test data. Within the ensemble model, the base learners are responsible for capturing diverse feature representations, while the meta-learner integrates these representations to construct a higher-capacity model. This collaborative approach enables the stacking classifier to capture more complex patterns and enhance classification performance.

4.5. The Training of the ES-IDB

After completing data preprocessing and feature selection, the training process is initiated. The model training process consists of two main components: adversarial training and the training of an ensemble stacking model. For the adversarial training phase, a CNN is employed as the foundational architecture. The CNN model is trained on the processed dataset, where during each epoch, standard training samples are first passed through the network to compute the standard loss using cross-entropy. Subsequently, AWP is applied to introduce small, gradient-based perturbations to the model weights. These perturbed weights are used to compute an additional adversarial loss, which is then accumulated with the standard loss to optimize the parameters. This process enhances the robustness to adversarial perturbations. The trained CNN outputs embeddings for downstream ensemble training.

In the ensemble stacking model, XGBoost, LightGBM, and CatBoost are trained on the augmented training data, which includes both standard and adversarially perturbed samples. Each base learner is configured with optimized hyperparameters obtained through Bayesian optimization. The outputs of these base learners are subsequently combined using a meta-model, implemented as a logistic regression classifier, to form the stacking ensemble. The meta-model is trained using the predictions of the base learners as features, leveraging the strengths of each model to achieve high accuracy. The final trained stacking model integrates the adversarially robust features from the CNN model with the prediction capabilities of the ensemble, ensuring robust performance across diverse input data.

4.6. The Testing of the ES-IDB

During the testing phase, the performance of the trained models is evaluated on unseen data. The CNN model, trained with AWP, is used as the feature extractor. The input test samples are first passed through the trained CNN to generate embeddings, which are subsequently used as inputs for the stacking model. The stacking model, comprising XGBoost, LightGBM, and CatBoost as base learners and a logistic regression meta-model, processes these embeddings to predict the final classification outcomes.

For each test instance, the stacking model computes the probability scores P_i for all possible classes. The predicted class is determined based on the highest probability score. The evaluation of the performance is conducted using metrics such as accuracy, precision, recall, F1-score, and ROC-AUC. The F1-score is used as the primary evaluation criterion, as it reflects the balance between precision and recall, providing a robust measure of performance across imbalanced datasets. The confusion matrix is generated to further analyze the classification results, and a ROC curve is plotted to assess the capability in multi-class classification scenarios. The trained model is evaluated for both its standard accuracy and its robustness under adversarial perturbations, ensuring that the adversarial training effectively enhances the resilience to such attacks.

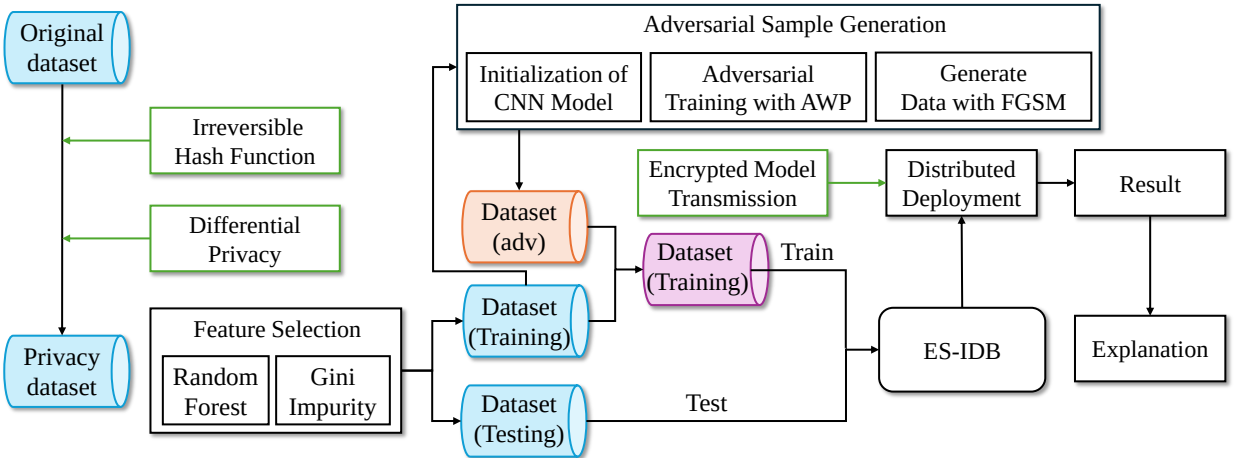


Figure 3: The operation process of ES-IDB under privacy protection.

4.7. The ES-IDB under Privacy-Preserving Techniques

In blockchain networks, intrusion detection is a critical task to safeguard network security. However, the training and deployment of models can involve handling sensitive information. This requires privacy protection mechanisms across different stages of the process, including data acquisition, preprocessing, and transmission. Based on the characteristics of the blockchain attack dataset used in this study, we introduce privacy-preserving techniques in three phases: data anonymization, differential privacy, and encrypted model transmission.

4.7.1. Data Anonymization During Data Loading

The blockchain network datasets contain fields such as protocol type (`tcp`) and service type (`http`). While these fields are important features for classification tasks, their specific values can reveal details about protocol implementations or network services. Therefore, during data loading, these fields are anonymized using irreversible hashing functions to preserve their uniqueness and prevent information reconstruction. Anonymization is performed as follows:

$$X_i^{\text{anon}} = \mathcal{H}(X_i^{\text{orig}}) \tag{14}$$

where X_i^{orig} represents the original categorical field, $\mathcal{H}(\cdot)$ is an irreversible hash function, and X_i^{anon} is the anonymized field. This process effectively protects sensitive information while maintaining the uniqueness required for the model to utilize these features in training.

4.7.2. DP During Data Preprocessing

DP ensures that the inclusion or exclusion of a single data point in a dataset does not significantly influence the output of an analysis, thus preserving individual privacy. In the preprocessing stage, we employ two classic noise mechanisms: the *Gaussian Mechanism* and the *Laplace Mechanism*, which add noise to data to protect privacy. DP is formally defined as follows: Let D and D' be two neighboring datasets, differing in only one data point. An algorithm $\mathcal{A}_p$ satisfies (ϵ, δ)-differential privacy if for any output subset S_b:

$$\Pr[\mathcal{A}_p(D) \in S_b] \leq e^{\epsilon} \cdot \Pr[\mathcal{A}_p(D') \in S_b] + \delta \tag{15}$$

where ϵ is the privacy budget, controlling the strength of privacy. δ is the probability of privacy violation, often set to a very small value.

Sensitivity Δf is a key concept in DP, quantifying the maximum change in the output of a function f when applied to two neighboring datasets:

$$\Delta f = \max_{D,D'} \|f(D) - f(D')\| \tag{16}$$

where D and D' differ in exactly one record and $\|\cdot\|$ is typically the L_1 or L_2 norm.

- Gaussian mechanism: The Gaussian mechanism provides (ϵ, δ)-differential privacy by adding noise sampled from a Gaussian distribution. The probability density function of the Gaussian distribution is:

$$P(x|\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \tag{17}$$

where σ is the standard deviation of the noise. The standard deviation σ_{Gau} of the Gaussian noise is computed as:

$$\sigma_{\text{Gau}} = \frac{\Delta f}{\epsilon} \cdot \sqrt{2 \ln \frac{1.25}{\delta}} \tag{18}$$

where δ is the failure probability. For a data point X_i, noise N_{Gauss} is sampled as:

$$N_{\text{Gau}} \sim \mathcal{N}(0, \sigma^2_{\text{Gau}}) \tag{19}$$

and the noisy data is obtained as:

$$X_i^{\text{dp}} = X_i + N_{\text{Gauss}} \tag{20}$$

- Laplace mechanism: The Laplace mechanism provides ϵ-differential privacy by adding noise drawn from the Laplace distribution. The probability density function of the Laplace distribution is:

$$P(x|\lambda) = \frac{1}{2\lambda} e^{-\frac{|x|}{\lambda}} \tag{21}$$

where $\lambda = \frac{\Delta f}{\epsilon}$ is the scale parameter, determined by the sensitivity Δf and the privacy budget ϵ. For a data point X_i, noise N_{Lap} is sampled as:

$$N_{\text{Lap}} \sim \text{Lap}(0, \frac{\Delta f}{\epsilon}) \tag{22}$$

and the noisy data is obtained as:

$$X_i^{\text{dp}} = X_i + N_{\text{Lap}} \tag{23}$$

For blockchain network intrusion detection, where data often involves sensitive network traffic, the Laplace mechanism is preferred for higher privacy protection, particularly when the sensitivity of features is significant. The Gaussian mechanism can be used when smoother noise addition and better utility are required, allowing a trade-off between privacy and performance. By employing DP mechanisms in the preprocessing stage, it is possible to safeguard sensitive data while maintaining high performance in intrusion detection tasks.

4.7.3. Encrypted Model Transmission During Deployment

In the proposed system, models often require distributed deployment or cross-node transmission, posing risks of theft or tampering. To secure the model during transmission, we utilize TLS to ensure confidentiality and integrity. The encrypted model transmission process is defined as:

$$M^{\text{enc}} = E(M, K), \quad M = D(M^{\text{enc}}, K) \tag{24}$$

where M is the original model, $E(M, K)$ means the encryption function with key K, and $D(M^{\text{enc}}, K)$ represents the decryption function with key K.

5. Performance and Explainability Analysis

5.1. Experimental Setup

In order to develop the proposed framework, the machine learning and deep learning algorithms were implemented using the Pandas, Scikit-learn, PyTorch, and ensemble libraries such as XGBoost, LightGBM, and CatBoost in Python. In the experiment, we used a machine equipped with a 12th Gen Intel(R) Core(TM) i7-12700KF 3.60 GHz and 32 Gigabytes (GB) of memory. The operating system was Windows 11 Enterprise Edition (23H2), and Python version 3.9.7 was employed for the implementation. Key library versions used include Pandas 1.3.3, Scikit-learn 1.0, PyTorch 1.9.0, XGBoost 1.5.0, LightGBM 3.3.1, and CatBoost 0.26. The experiments were conducted in an environment with CUDA support (if available) to leverage GPU acceleration, specifically using an NVIDIA RTX 4060 Ti for deep learning tasks. All models and experimental results were reproducible using the random seed set to 12.

5.1.1. Datasets

We select the BNaT dataset as our experimental dataset for evaluation. The BNaT dataset is derived from a laboratory environment simulating real-world blockchain network conditions and includes both normal and attack traffic in blockchain networks. The dataset specifically targets network-layer traffic in Ethereum-based blockchain systems and captures four types of attacks: BP, DoS, FoT, and MitM. Khoa et al. [39] from the University of Canberra set up the experimental network using Ethereum fullnodes, a bootnode, and a netstats server. The dataset is rich in features and is ideal for intrusion detection, anomaly behavior analysis, and blockchain security research. The data is divided into multiple CSV files, each representing the state of blockchain networks during a specific timeframe, including both normal and attack scenarios. The captured data reflects real-world attack behavior and traffic patterns, making it a comprehensive resource for blockchain security analysis. This structured dataset format allows researchers to analyze attack behaviors effectively, benchmark detection models, and compare their results with others in the field.

5.1.2. Evaluation Metrics

In our experiment, we evaluate the model performance using four key metrics: accuracy, precision, recall, and f1-score, which are defined as follows.

- **Accuracy**: Measures the overall correctness of predictions and is calculated as:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

  where TP, TN, FP, and FN represent true positives, true negatives, false positives, and false negatives, respectively.

- **Precision**: Assesses the proportion of true positives among all predicted positives, reflecting the reliability in positive class predictions:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

- **Recall**: Measures the proportion of actual positives correctly identified by the model, indicating its sensitivity:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

- **F1-Score**: The harmonic mean of precision and recall, particularly useful for imbalanced datasets, is defined as:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

For the original dataset without adversarial augmentation, we evaluate the model using all four metrics to provide a comprehensive analysis of its performance. However, for the augmented dataset with adversarial samples generated using FGSM, F1-Score is emphasized due to its balanced consideration of precision and recall, making it more suitable for analyzing robustness in adversarial scenarios. By focusing on these metrics, we ensure an effective evaluation of the accuracy, sensitivity, and resilience under varying conditions.

Table 1
Comparison of classification results for different algorithms.

Model	Metric	BP	DoS	FoT	MitM	Normal	Overall Accuracy
Logistic Regression	Precision	0.70303	1.00000	0.89574	0.64640	0.91531	0.89186
	Recall	0.48767	1.00000	0.82767	0.59533	0.95753	
	F1-Score	0.57587	1.00000	0.86036	0.61982	0.93594	
Support Vector Machine	Precision	0.79813	1.00000	0.95783	0.70854	0.96722	0.93533
	Recall	0.82633	1.00000	0.84800	0.81033	0.96100	
	F1-Score	0.81199	1.00000	0.89958	0.75603	0.96410	
Random Forest	Precision	0.97062	1.00000	0.95316	0.93773	0.98580	0.98002
	Recall	0.98000	1.00000	0.91567	0.94367	0.98810	
	F1-Score	0.97529	1.00000	0.93404	0.94069	0.98695	
Gradient Boosting	Precision	0.96396	1.00000	0.95337	0.89862	0.98350	0.97490
	Recall	0.98067	0.99967	0.89967	0.93667	0.98320	
	F1-Score	0.97224	0.99983	0.92574	0.91725	0.98335	
Deep Neural Network	Precision	0.94230	1.00000	0.93371	0.83462	0.98055	0.96498
	Recall	0.96900	1.00000	0.92500	0.86800	0.97477	
	F1-Score	0.95546	1.00000	0.92934	0.85098	0.97765	
ES-IDB (Ours)	Precision	0.97863	1.00000	0.96560	0.94375	0.98600	0.98205
	Recall	0.97700	1.00000	0.91700	0.95067	0.99040	
	F1-Score	0.97781	1.00000	0.94067	0.94719	0.98819	

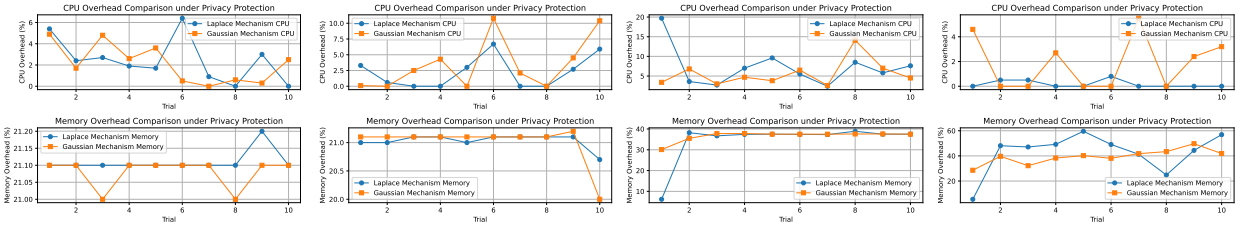
5.1.3. Data Preprocessing

We preprocess the data in the BNaT dataset to prepare it for model training and evaluation. First, we perform data cleaning by encoding categorical variables into numerical features using one-hot encoding to eliminate redundancy and handle categorical data effectively. Next, we apply the `LabelEncoder` to transform the target labels into numeric format suitable for modeling. Following this, we normalize the feature values using the `StandardScaler` to ensure that all features have a standard range, improving the convergence of machine learning models. After preprocessing the features, we randomly split the dataset into training and testing subsets using an 8:2 ratio, ensuring that the split maintains the class balance by stratifying on the target variable. The training dataset is further processed to select the most important features based on feature importance scores computed by a trained `RandomForestClassifier`. We retain the top 80% of the features with the highest importance scores for both the training and testing datasets, ensuring that only the most relevant information is used for subsequent analysis. Finally, the processed training data is used to train both adversarial robust neural networks and ensemble models, including stacking classifiers composed of `XGBoost`, `LightGBM`, and `CatBoost` as base learners with logistic regression as the meta-learner. This structured preprocessing ensures high-quality data for model training and robust performance evaluation.

5.2. Experimental Results on BNaT

The experimental results are given in Table 1. The experimental results demonstrate that the proposed ES-IDB achieves superior classification performance compared to traditional machine learning models, including Logistic Regression (LR), Support Vector Machine (SVM), Random Forest (RF), Gradient Boosting (GB), and Deep Neural Network (DNN). ES-IDB achieves an overall accuracy of 98.205%, outperforming all benchmark models. The model has a macro average F1 score of 97.078% and a weighted average F1 score of 98.197%, further showing its robustness in handling blockchain network datasets.

LR demonstrates limited efficacy with an overall accuracy of 89.186%, reflecting its inability to capture the complex non-linear patterns inherent in the BNaT dataset. Poor performance in classes such as BP and MitM indicates its susceptibility to underfitting. Although it achieves perfect classification for the DoS class, this is insufficient to address the diverse attack types comprehensively. SVM outperforms Logistic Regression with an accuracy of 93.533%, benefiting from its ability to handle non-linear decision boundaries. Nevertheless, it struggles with inter-class variability, particularly for the MitM class. The relatively lower macro-average F1-score of 88.634% compared



(a) 30000 records with $\epsilon = 0.1$ (b) 30000 records with $\epsilon = 0.5$ (c) 40000 records with $\epsilon = 0.5$ (d) 50000 records with $\epsilon = 1.0$

Figure 4: CPU and memory overhead of ES-IDB under Gaussian and Laplace mechanisms.

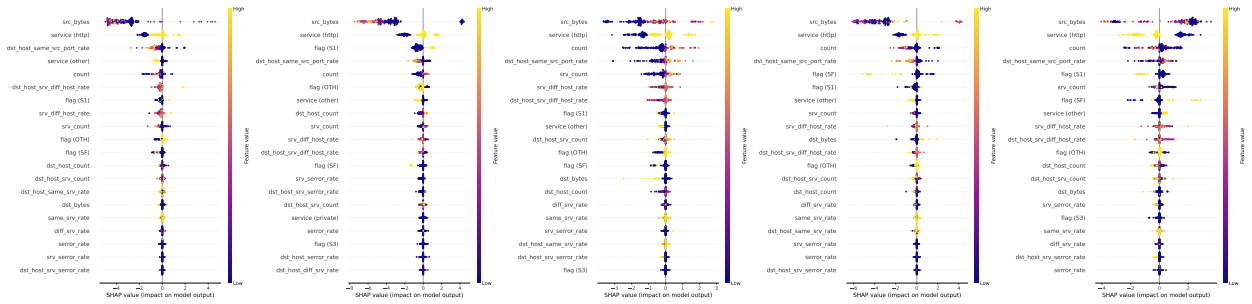
to ES-IDB suggests that SVM, while effective, is suboptimal for high-dimensional attack traffic data. Ensemble-based models like RF and GB achieve competitive results, with accuracies of 98.002% and 97.490%, respectively. Both models exhibit high precision and recall across most attack classes, reflecting their strength in handling heterogeneous data distributions. However, their performance for the FoT class is marginally inferior to ES-IDB. These results indicate that while traditional ensemble techniques are effective, they fail to fully exploit the intricate relationships among features in the BNaT dataset. DNN achieves an accuracy of 96.498%, which demonstrates the potential of neural networks to model complex attack traffic patterns. Additionally, the DNN underperforms compared to ES-IDB due to its lower recall for MitM and BP classes, which are critical for ensuring the robustness of intrusion detection in blockchain networks. The results indicate that a standalone neural network model can lack the adaptability required to handle diverse attack types in blockchain traffic.

The experimental results depicted in Fig. 4 show the comparative analysis of CPU and memory overheads under Gaussian and Laplace mechanisms during the privacy-preserving process. Overall, the larger the amount of data, the more memory is consumed. the CPU usage is related to the privacy budget and is largest at around 0.5. For CPU overhead, the Gaussian mechanism consistently exhibits lower variability compared to the Laplace mechanism, particularly notable in scenarios with smaller privacy budgets such as Fig. 4a, which indicates that the Gaussian mechanism provides more stable and predictable performance under strict privacy conditions. In contrast, memory overhead displays a relatively stable trend across trials for both mechanisms, with Gaussian maintaining a slight advantage in most cases, as seen in Fig. 4c, and Fig. 4d. The stability and efficiency of the Gaussian mechanism in both CPU and memory usage, especially under increasing data volumes and relaxed privacy constraints, suggest its suitability for intrusion detection tasks where resource optimization is critical.

5.3. Explainability Analysis

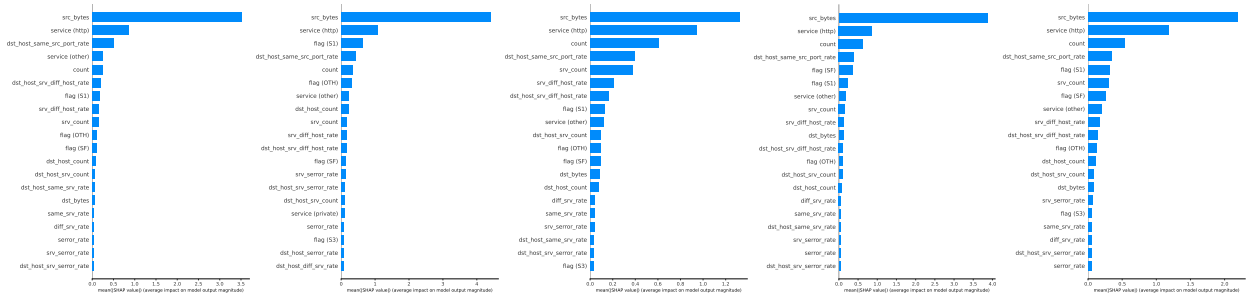
Fig. 5 shows the SHAP values for each feature in the five classes, where the first four are attack-type traffic and the last one is normal-type traffic. From Fig. 5a, the SHAP value plot reveals that *src_bytes* and *service* (*http*) are the dominant features, with significant SHAP impacts both positively and negatively. This suggests that the amount of data transmitted from the source to the destination is crucial for detecting BP attacks. BP attacks are often characterized by multiple failed login attempts that generate a low volume of data per connection, which aligns with the observed SHAP contributions. Another key feature $dst\_host\_same\_src\_port\_rate$ highlights the reuse of source ports, which is typical in brute-force attacks where attackers iterate through credentials from the same origin. Additionally, features like *count* and *srv_count*, which capture the number of connections and service requests, exhibit moderate SHAP impact, indicating frequent attempts to access services during password brute-forcing. Flags such as $flag$ ($S1$) and $flag$ (OTH), which represent incomplete handshakes or error states, further contribute to BP detection, emphasizing the failed or partially established connections caused by unsuccessful authentication attempts. Overall, BP attacks are primarily detected through low *src_bytes* values, service usage patterns, and repetitive connection attempts, which are reflected clearly in the SHAP value distribution.

In Fig. 5b, the SHAP value of the DoS attack also highlights *src_bytes* as the most significant feature, showing both high and low values having considerable impacts on the model output. This indicates that the amount of data transmitted during connections is highly anomalous during DoS attacks, which can involve large volumes of traffic or null data payloads. The feature *service* (*http*) also has a strong positive impact, suggesting that HTTP-based flooding is prevalent in this attack class. Flags such as $flag$ ($S1$) and $flag$ (OTH) exhibit high SHAP values, which indicates incomplete handshakes or error-prone connections caused by the overwhelming request floods, but they



(a) BP Attack (b) DoS Attack (c) FoT Attack (d) MitM Attack (e) Normal Attack

Figure 5: Beeswarm of macroscopic feature density.



(a) BP (b) DoS (c) FoT (d) MitM (e) Normal

Figure 6: SHAP value for importance of macroscopic features.

are less important than the process in the BP attack. The $dst\_host\_same\_src\_port\_rate$ and $count$ features further reinforce the repeated targeting of specific hosts or services using the same port, a characteristic pattern of DoS traffic. Notably, $dst\_host\_diff\_srv\_rate$ and $dst\_host\_serror\_rate$ contribute significantly to predictions, capturing service diversity and error rates that spike during such attacks. In summary, DoS attacks are marked by abnormal data volumes, repeated connection attempts, and frequent errors, all of which are effectively highlighted by SHAP values.

In the case of FoT attacks in Fig. 5c, the SHAP value plot reveals that src_bytes and $service$ ($http$) remain dominant features, similar to the other attack types, but with specific nuances. The $count$ and srv_count features show high SHAP impacts, indicating frequent and repetitive transactions that overload the blockchain network. This pattern aligns with the nature of FoT attacks, where attackers spam the network with numerous meaningless or null transactions. $dst\_host\_same\_src\_port\_rate$, which measures the percentage of connections using the same source port and destination IP, also emerges as a significant indicator, reflecting the concentrated and repetitive nature of the transactions. Flags such as $flag$ (SF) and $flag$ ($S1$) show considerable SHAP values, suggesting that both successful and failed handshakes contribute to FoT detection. Features like $dst\_host\_srv\_diff\_host\_rate$ and $srv\_diff\_host\_rate$ are also important, as they highlight the distribution of connections across different hosts, which tends to be lower in FoT attacks compared to more sophisticated attacks. Thus, FoT attacks are primarily identified by high connection counts, repetitive service usage, and abnormal src_bytes values.

For MitM attacks in Fig. 5d, the results demonstrate that src_bytes and $service$ ($http$) are once again the most influential features, indicating the critical role of data transmission volumes and HTTP-related connections. The $count$ and $dst\_host\_same\_src\_port\_rate$ features further contribute to detection, emphasizing the repetitive use of ports and connections during interception. Flags such as $flag$ (SF), $flag$ ($S1$), and $flag(OTH)$ also serve a vital function, capturing connection states where MitM activities result in altered or tampered communications. The presence of $srv\_diff\_host\_rate$ and $dst\_host\_srv\_diff\_host\_rate$ highlights deviations in connection behaviors, as MitM attacks often involve relaying or modifying traffic between different hosts. Features such as srv_count and dst_bytes also exhibit moderate SHAP impacts, reflecting the communication volumes intercepted or forwarded by

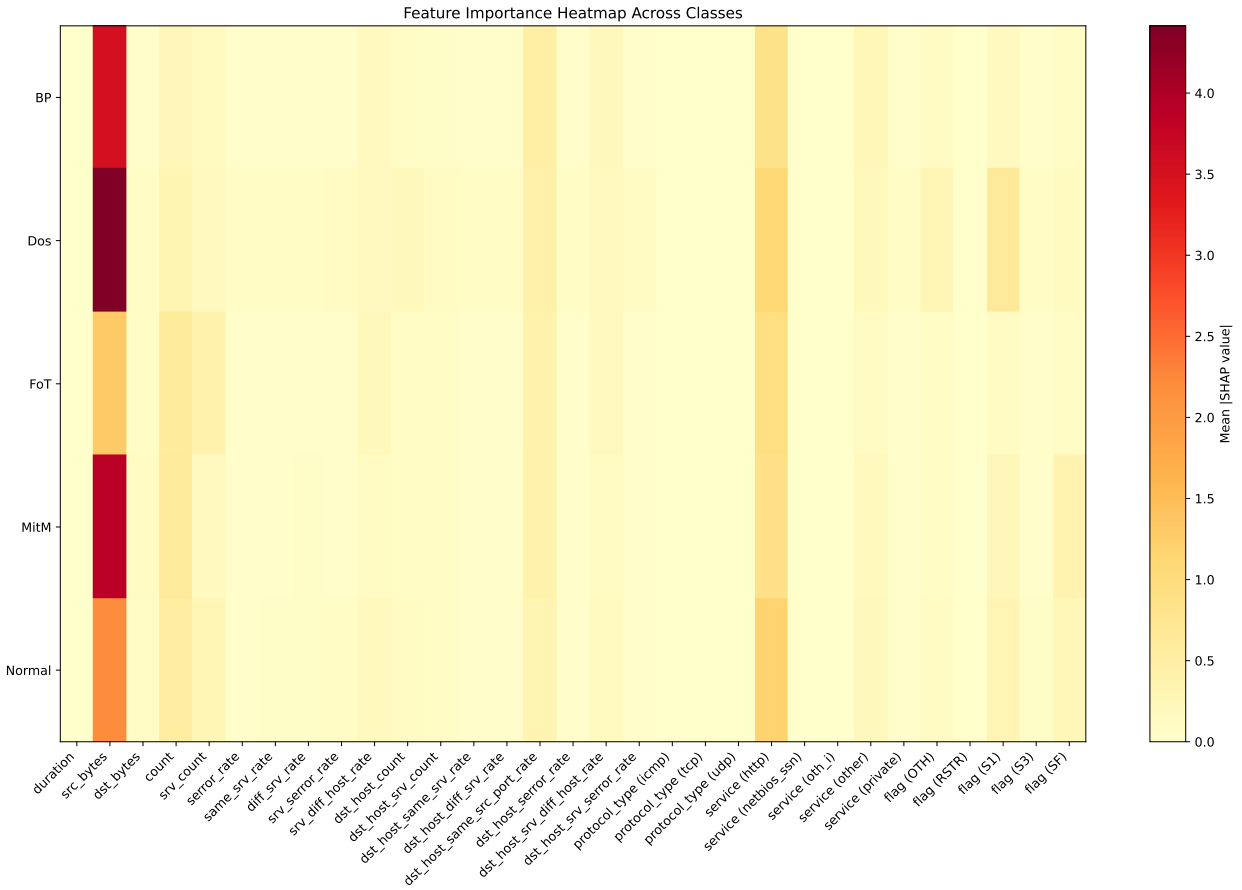


Figure 7: The Heatmap of features in BNaT.

the attacker. Overall, MitM attacks are distinguished by abnormal connection flags, moderate traffic volumes, and repetitive connections to specific services.

In Fig. 5e, normal traffic shows a more balanced distribution across features, with no single feature exhibiting extreme impacts. *src_bytes* remains a key feature, but the lower it is the higher the SHAP value, indicating that normal traffic does not transfer excessive amounts of data. The feature *service* (*http*) also contributes to the prediction, and lower values result in higher SHAP values, reflecting the fact that non-HTTP services are less susceptible to attacks. The *count*, *srv_count*, and $dst\_host\_same\_src\_port\_rate$ features show moderate SHAP impacts, indicating a normal distribution of connections without excessive repetition or concentration. Flags such as $flag$ (SF), which represent successful connections, dominate the SHAP values for normal traffic, emphasizing the prevalence of error-free handshakes. Additionally, features like $diff\_srv\_rate$ and *same_srv_rate* exhibit balanced values, highlighting the absence of suspicious service usage patterns. In summary, normal traffic is characterized by stable byte volumes, balanced connection states, and error-free communications.

To highlight the importance of each feature, bar charts of the SHAP summary are plotted in Fig. 6. The figure reveals that src_bytes consistently emerges as the most influential feature across all categories, including BP, DoS, FoT, MitM attacks, and normal traffic. This highlights the critical role of data transmission volume in distinguishing various network behaviors. The *service* (*http*) feature also demonstrates significant impact, indicating that HTTP services are frequently targeted during malicious activities. $dst\_host\_same\_src\_port\_rate$ and *count* capture repetitive connection patterns and port reuse, which are typical in attack scenarios. Connection state flags are also useful for identifying incomplete handshakes and failed connections, further aiding anomaly detection. Service diversity indicators exhibit moderate importance, reflecting variations in service concentration and distribution across attack types. Therefore, the

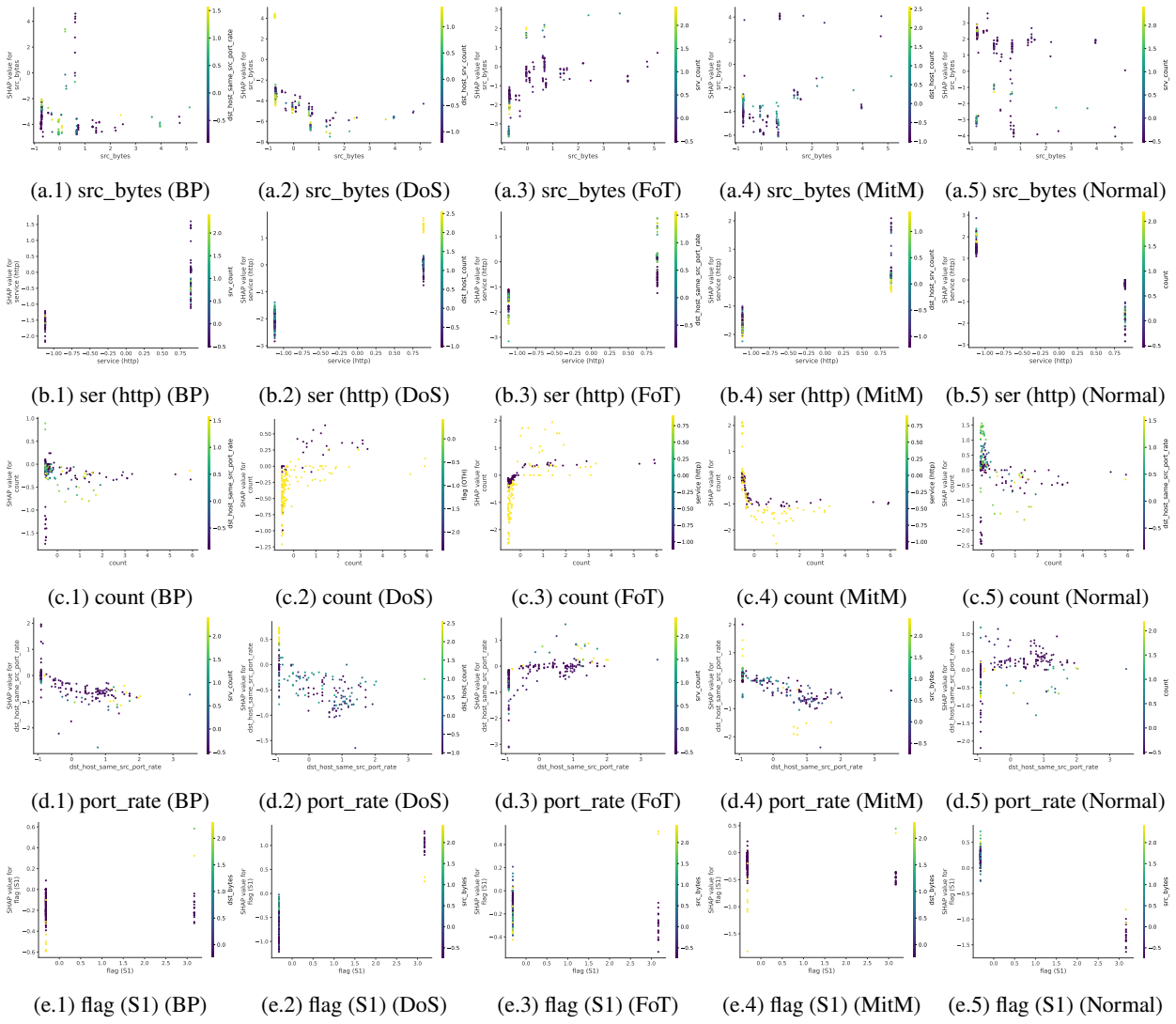
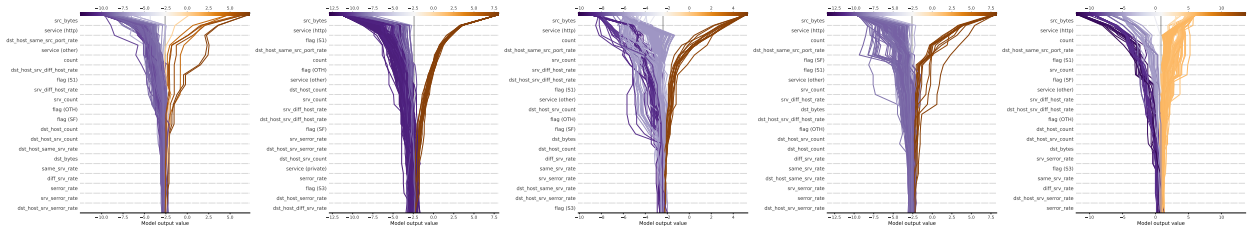


Figure 8: Results of interactions between the top 5 features in terms of combined importance.

SHAP values show distinct patterns in data transmission, connection redundancy, and service usage, which effectively differentiate between attack behaviors and normal blockchain network traffic.

For detail comparison, the heatmap in Fig. 7 provides a class-specific perspective on feature importance and shows the varying influence of features across attack types and normal traffic. Unlike the SHAP summary plots, which focus on average impacts, this heatmap gives distinct feature importance distributions. For example, *src_bytes* remains dominant but shows varying intensity across classes, being most prominent in BP and MitM while comparatively subdued in FoT and Normal. Interestingly, connection flags like $flag\ (S1)$ and service diversity features display unique importance patterns for FoT and DoS, which indicates their relevance in identifying repetitive or targeted traffic. This variability in feature intensity across classes, especially for low-importance features in other contexts, underscores the values that can remain hidden in aggregated SHAP summaries. For Normal traffic, the data points are more evenly distributed, but are still concentrated in areas with low count values. $dst\_host\_same\_src\_port\_rate$ has a significant color gradient, and the SHAP values are more densely distributed in negative areas.

To plot feature interactions, we chose the top 5 features in terms of overall importance and automatically selected the features with the highest interactions in each category for plotting, as shown in Fig. 8. Interaction feature plots show the joint effect of two features on model predictions at different feature values and can help to detect possible nonlinear



(a) BP (b) DoS (c) FoT (d) MitM (e) Normal

Figure 9: Comparison of model decision explanation for different cyber attack classes based on SHAP values.

relationships or interactions between features. The first row of Fig. 8 is for the *src_bytes* feature analysis, which has the strongest importance. In BP attacks, the feature most dependent on *src_bytes* is *dst_host_same_src_port_rate*, showing a certain dependency trend between the two. When *src_bytes* values range from approximately −1 to 0, the SHAP values are relatively low, while *dst_host_same_src_port_rate* exhibits a positive value close to 1.5. This indicates that a decrease in the number of source data bytes significantly impacts port statistical characteristics when attackers frequently attempt connections using specific source ports. However, as *src_bytes* increases, the SHAP values get progressively closer to 0, suggesting a weakening dependency on port usage. This nonlinear dependency means the behavioral characteristics of BP attacks, where attackers repeatedly use the same port for brute-force password cracking attempts. In the DoS attack class, *src_bytes* and *dst_host_srv_count* exhibit a strong negative correlation. The figure shows that when *src_bytes* is relatively low, the SHAP values decrease significantly to a range of −6 to −8, while *dst_host_srv_count* reaches values close to 1.5. This shows an inverse relationship between the increase in service connection count and the volume of source data bytes. This behavior reflects the characteristic of DoS attacks, where attackers send numerous small data packets to request services, causing a rapid increase in the number of service connections on the target host, thereby imposing a heavy burden on system resources. As *src_bytes* increases, the SHAP values rise and approach 0, suggesting that the dependency between high data byte traffic and service connection statistics becomes relatively weaker. For FoT attacks, *src_bytes* presents a similar negative dependence on *srv_count*, with SHAP values of −2 to −3 at lower *src_bytes* intervals (−1 to 1) and *srv_count* values close to 2.0, which reflects the fact that the attacker occupies the service resources by generating some meaningless transactions, and this dependence gradually weakens when *src_bytes* rises this dependency gradually diminishes. In the MitM attack, the SHAP value decreases when *src_bytes* is lower, and at this time *dst_host_count* takes a value close to 2.5, explaining that the attacker forwards the traffic through an intermediate node, which leads to an abnormal increase in the number of connections to the target host. In the Normal class, *src_bytes* and *srv_count* present mainly fewer dependencies, indicating a smoother and more natural feature dependency for normal traffic.

The second row in Fig. 8 is the second most important feature. In the BP attack, the distribution of data points is mainly concentrated in two locations, and *srv_count* shows obvious distribution differences with the change of SHAP value. The data points are densely distributed when the SHAP value is lower, indicating that the contribution of *srv_count* to *service* (*http*) is more stable and the influence is concentrated in the attack scenario. *srv_count* is the count of the number of connections to the same service, which has a greater impact on the SHAP of the *service* (*http*) feature in the BP attack. In the DoS attack, the data is still concentrated in two main areas, and the color of *dst_host_count* varies significantly with SHAP value, especially in the higher areas where the data points are brightly colored. For the FoT attack, the strongest interaction feature is *dst_host_same_src_port_rate*, and the data points are more densely populated in the region of negative SHAP contribution, which suggests that the FoT attack sends transactions through the same source ports, resulting in a negative impact on the *service* (*http*). In the MitM attack, *dst_host_srv_count* shows a strong positive dependence on *service* (*http*). In the figure, we can see that when the number of service connections increases, the SHAP value gradually increases, which represents that the MitM attack has a greater impact on the utilization of service connections.

The feature *count* is the third most important feature, and the largest interaction feature in each class is shown in the third row of Fig. 8. In the BP attack, the data points are mainly distributed in the region of 0 value of count. *dst_host_same_src_port_rate* value has a more noticeable color change, and the data points are densely distributed when the SHAP value is low. For DoS attacks, the distribution of data points is concentrated in the region of low values of count, and the color change of *flag* (*OTH*) feature is positively correlated with the change of SHAP value. Among

them, when $flag$ (OTH) is a higher value, the SHAP value is positively contributed, which means that the influence of $flag$ (OTH) on $count$ gradually increases. For the FoT, there is a large positive and negative variation in SHAP values, with some data points showing a strong negative contribution near the $count$ of 0. For the MtiM, the SHAP value is almost close to 0 in a few regions of larger values of $count$, demonstrating that the effect of $service$ ($http$) diminishes as connections increase.

For the $dst\_host\_same\_src\_port\_rate$, the interaction phenomenon occurs mainly below 2. In BP attacks, the interaction feature with srv_count suggests that the service connections has a mixed positive and negative effect on the SHAP value of $dst\_host\_same\_src\_port\_rate$, with a negative contribution at a low rate and a positive one at a high rate. In DoS attacks, dst_host_count is the main interaction feature, and connections to the target host leads to a negative response to the SHAP value. For FoT attacks, srv_count is again the interaction feature, and the service flooding behavior leads to a negative contribution to the SHAP value at low connection ratios. In MitM attack, src_bytes is the strongest interaction feature, and the negative contribution is more significant when the number of data bytes from source to target increases. In Normal traffic, the count feature is the strongest interaction feature, and connections of the source IP mainly contribute negatively.

The last line in Fig. 8 represents the feature flag (S1). In BP and MitM attacks, dst_bytes is the main interaction feature, and the target byte count mainly contributes negatively, but also positively under certain conditions. src_bytes is the main interaction feature in DoS and FoT attacks, and the change of the source byte count reflects the strength of the attack traffic, which has a significant positive and negative interaction effect on the SHAP value. In Normal traffic, the variation range of SHAP value is small, and the negative contribution is dominant.

Fig. 9 is a SHAP decision plot that clearly demonstrates how each feature drives changes in model predictions. In the figure, the contributions of the features are cumulative and the impact of each feature is shown as a stepped. src_bytes and $service$ ($http$) are consistently shown as the dominant features in all five classes, and the direction and strength of their contributions reflect the distribution of features and anomaly patterns in the different categories. srv_count and $count$ are more significant in BP and FoT attacks, while src_bytes and dst_bytes in MitM and DoS attacks dominate the model decision. The contribution of most features in the normal class is in the negative direction, implying that the model uses them as differentiators for the attack class.

6. Conclusion

Blockchain technology has revolutionized decentralized systems, but its unique characteristics, including peer-to-peer communication and cryptographic protocols, present significant challenges for intrusion detection. This paper proposed the ES-IDB, a framework that combines high detection accuracy, resilience to adversarial attacks, and privacy techniques to safeguard sensitive data. By integrating SHAP for explainability, ES-IDB enhances trust and usability in practical domains. Experimental results demonstrate the effectiveness of the framework, whick markes an important step toward secure and explainable IDS solutions for blockchain netowrks. Future work will focus on further optimizing the scalability of the system and exploring its applicability to emerging blockchain use cases such as decentralized finance (DeFi) and cross-chain communication.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to thank the Research and Innovation Office of the Hong Kong Polytechnic University for supporting the project.

References

[1] Daniel Lazaro, Joan Manuel Marques, Josep Jorba, and Xavier Vilajosana. Decentralized resource discovery mechanisms for distributed computing in peer-to-peer environments. *ACM Computing Surveys*, 45(4):1–40, August 2013.

[2] Eugenia Politou, Fran Casino, Efthimios Alepis, and Constantinos Patsakis. Blockchain mutability: Challenges and proposed solutions. *IEEE Transactions on Emerging Topics in Computing*, 9(4):1972–1986, October 2021.

[3] Rajesh Kumar Singh, Ruchi Mishra, Shivam Gupta, and Archana A. Mukherjee. Blockchain applications for secured and resilient supply chains: A systematic literature review and future research agenda. *Computers amp; Industrial Engineering*, 175:108854, January 2023.

[4] Arunima Ghosh, Shashank Gupta, Amit Dua, and Neeraj Kumar. Security of cryptocurrencies in blockchain technology: State-of-art, challenges and future prospects. *Journal of Network and Computer Applications*, 163:102635, August 2020.

[5] Pankaj Dutta, Tsan-Ming Choi, Surabhi Somani, and Richa Butala. Blockchain technology in supply chain operations: Applications, challenges and research opportunities. *Transportation Research Part E: Logistics and Transportation Review*, 142:102067, October 2020.

[6] Andrew J, Deva Priya Isravel, K. Martin Sagayam, Bharat Bhushan, Yuichi Sei, and Jennifer Eunice. Blockchain for healthcare systems: Architecture, security challenges, trends and future directions. *Journal of Network and Computer Applications*, 215:103633, June 2023.

[7] Mohd Javaid, Abid Haleem, Ravi Pratap Singh, Rajiv Suman, and Shahbaz Khan. A review of blockchain technology applications for financial services. *BenchCouncil Transactions on Benchmarks, Standards and Evaluations*, 2(3):100073, 2022.

[8] Bin Cao, Zhenghui Zhang, Daquan Feng, Shengli Zhang, Lei Zhang, Mugen Peng, and Yun Li. Performance analysis and comparison of pow, pos and dag based blockchains. *Digital Communications and Networks*, 6(4):480–485, November 2020.

[9] Jie Xu, Cong Wang, and Xiaohua Jia. A survey of blockchain consensus protocols. *ACM Computing Surveys*, 55(13s):1–35, July 2023.

[10] Ly Vu, Quang Uy Nguyen, Diep N. Nguyen, Dinh Thai Hoang, and Eryk Dutkiewicz. Deep generative learning models for cloud intrusion detection systems. *IEEE Transactions on Cybernetics*, 53(1):565–577, January 2023.

[11] Mohammad Shahin, Mazdak Maghanaki, Ali Hosseinzadeh, and F. Frank Chen. Advancing network security in industrial iot: A deep dive into ai-enabled intrusion detection systems. *Advanced Engineering Informatics*, 62:102685, October 2024.

[12] Jie Cui, Jietian Xiao, Hong Zhong, Jing Zhang, Lu Wei, Irina Bolodurina, and Debiao He. Lh-ids: Lightweight hybrid intrusion detection system based on differential privacy in vanets. *IEEE Transactions on Mobile Computing*, 23(12):12195–12210, December 2024.

[13] Fabian Dietrich, Yiwen Ge, Ali Turgut, Louis Louw, and Daniel Palm. Review and analysis of blockchain projects in supply chain management. *Procedia Computer Science*, 180:724–733, 2021.

[14] V Hariharasudan and Suhail Javed Quraishi. A review on blockchain based identity management system. In *2022 3rd International Conference on Intelligent Engineering and Management (ICIEM)*, page 735–740. IEEE, April 2022.

[15] Sachikanta Dash, Pradosh Kumar Gantayat, and Rajendra Kumar Das. *Blockchain Technology in Healthcare: Opportunities and Challenges*, page 97–111. Springer International Publishing, 2021.

[16] Ali Benabdallah, Antoine Audras, Louis Coudert, Nour El Madhoun, and Mohamad Badra. Analysis of blockchain solutions for e-voting: A systematic literature review. *IEEE Access*, 10:70746–70759, 2022.

[17] Dan Wang, Jindong Zhao, and Yingjie Wang. A survey on privacy protection of blockchain: The technology and application. *IEEE Access*, 8:108766–108781, 2020.

[18] Ke He, Dan Dongseong Kim, and Muhammad Rizwan Asghar. Adversarial machine learning for network intrusion detection systems: A comprehensive survey. *IEEE Communications Surveys amp; Tutorials*, 25(1):538–566, 2023.

[19] Rupesh Kumar Dewang, Aditya Raven, and Arvind Mewada. A machine learning-based privacy-preserving model for covid-19 patient using differential privacy. In *2021 19th OITS International Conference on Information Technology (OCIT)*, page 90–95. IEEE, December 2021.

[20] Subash Neupane, Jesse Ables, William Anderson, Sudip Mittal, Shahram Rahimi, Ioana Banicescu, and Maria Seale. Explainable intrusion detection systems (x-ids): A survey of current methods, challenges, and opportunities. *IEEE Access*, 10:112392–112415, 2022.

[21] Mohiuddin Ahmed, Abdun Naser Mahmood, and Md. Rafiqul Islam. A survey of anomaly detection techniques in financial domain. *Future Generation Computer Systems*, 55:278–288, February 2016.

[22] Vinayakumar Ravi. Deep learning-based network intrusion detection in smart healthcare enterprise systems. *Multimedia Tools and Applications*, 83(13):39097–39115, October 2023.

[23] Ahamed Aljuhani, Abdulelah Alamri, Prabhat Kumar, and Alireza Jolfaei. An intelligent and explainable saas-based intrusion detection system for resource-constrained iomt. *IEEE Internet of Things Journal*, 11(15):25454–25463, August 2024.

[24] Mohamed Abdel-Basset, Nour Moustafa, Hossam Hawash, Imran Razzak, Karam M. Sallam, and Osama M. Elkomy. Federated intrusion detection in blockchain-based smart transportation systems. *IEEE Transactions on Intelligent Transportation Systems*, 23(3):2523–2537, March 2022.

[25] Swapna Siddamsetti. Anomaly detection in blockchain using machine learning. *Journal of Electrical Systems*, 20(3):619–634, April 2024.

[26] Mohammad Hasan, Mohammad Shahriar Rahman, Helge Janicke, and Iqbal H. Sarker. Detecting anomalies in blockchain transactions using machine learning classifiers and explainability analysis. *Blockchain: Research and Applications*, 5(3):100207, September 2024.

[27] Khattab M. Ali Alheeti and Klaus McDonald-Maier. Intelligent intrusion detection in external communication systems for autonomous vehicles. *Systems Science amp; Control Engineering*, 6(1):48–56, January 2018.

[28] Pengju Liu. An intrusion detection system based on convolutional neural network. In *Proceedings of the 2019 11th International Conference on Computer and Automation Engineering*, ICCAE 2019, page 62–67. ACM, February 2019.

[29] Ying Gao, Hongrui Wu, Binjie Song, Yaqia Jin, Xiongwen Luo, and Xing Zeng. A distributed network intrusion detection system for distributed denial of service attacks in vehicular ad hoc network. *IEEE Access*, 7:154560–154571, 2019.

[30] MohammadNoor Injadat, Abdallah Moubayed, Ali Bou Nassif, and Abdallah Shami. Multi-stage optimized machine learning framework for network intrusion detection. *IEEE Transactions on Network and Service Management*, 18(2):1803–1816, June 2021.

[31] Muneeb Ul Hassan, Mubashir Husain Rehmani, and Jinjun Chen. Anomaly detection in blockchain networks: A comprehensive survey. *IEEE Communications Surveys amp; Tutorials*, 25(1):289–318, 2023.

[32] Prabhat Kumar, Randhir Kumar, Govind P. Gupta, and Rakesh Tripathi. A distributed framework for detecting ddos attacks in smart contract-based blockchain-iot systems by leveraging fog computing. *Transactions on Emerging Telecommunications Technologies*, 32(6), September 2020.

[33] Osama Alkadi, Nour Moustafa, Benjamin Turnbull, and Kim-Kwang Raymond Choo. A deep blockchain framework-enabled collaborative intrusion detection for protecting iot and cloud networks. *IEEE Internet of Things Journal*, 8(12):9463–9472, June 2021.

[34] Jinoh Kim, Makiya Nakashima, Wenjun Fan, Simeon Wuthier, Xiaobo Zhou, Ikkyun Kim, and Sang-Yoon Chang. Anomaly detection based on traffic monitoring for secure blockchain networking. In *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, page 1–9. IEEE, May 2021.

[35] Zengguang Liu and Xiaochun Yin. Lstm-cgan: Towards generating low-rate ddos adversarial samples for blockchain-based wireless network detection models. *IEEE Access*, 9:22616–22625, 2021.

[36] Wanqin Cao, Yunhui Huang, Dezheng Li, Feng Yang, Xiaofeng Jiang, and Jian Yang. A blockchain based link-flooding attack detection scheme. In *2021 IEEE 4th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, page 1665–1669. IEEE, June 2021.

[37] Paritosh Ramanan, Dan Li, and Nagi Gebraeel. Blockchain-based decentralized replay attack detection for large-scale power systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 52(8):4727–4739, August 2022.

[38] Bowen Hu, Chunjie Zhou, Yu-Chu Tian, Yuanqing Qin, and Xinjue Junping. A collaborative intrusion detection approach using blockchain for multimicrogrid systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(8):1720–1730, August 2019.

[39] Tran Viet Khoa, Do Hai Son, Dinh Thai Hoang, Nguyen Linh Trung, Tran Thi Thuy Quynh, Diep N. Nguyen, Nguyen Viet Ha, and Eryk Dutkiewicz. Collaborative learning for cyberattack detection in blockchain networks. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 54(7):3920–3933, July 2024.

[40] Bhawana Sharma, Lokesh Sharma, Chhagan Lal, and Satyabrata Roy. Explainable artificial intelligence for intrusion detection in iot networks: A deep learning based approach. *Expert Systems with Applications*, 238:121751, March 2024.

[41] Hampus Lundberg, Nishat I Mowla, Sarder Fakhrul Abedin, Kyi Thar, Aamir Mahmood, Mikael Gidlund, and Shahid Raza. Experimental analysis of trustworthy in-vehicle intrusion detection system using explainable artificial intelligence (xai). *IEEE Access*, 10:102831–102841, 2022.