A yellow bird with a crown on a blue background

Description automatically generated

Anglia Ruskin University

Cyber Security and AI

Module Code: MOD006570 TRI2 F01CAM

Academic Year: 2024/2025

Mohammad Ali Jaber (SID: 2365662)

Contents

[1. Part A: Intrusion Detection using XGBoost 3](#_Toc196863197)

[1.1. Model Selection Rationale: 3](#_Toc196863198)

[1.2. Experimental Setup: 3](#_Toc196863199)

[1.3. Results: 4](#_Toc196863200)

[1.4. Critical Evaluation: 5](#_Toc196863201)

[2. Part B: Data Poisoning and Its Impact on Intrusion Detection 7](#_Toc196863202)

[2.1. Random Label Flipping 7](#_Toc196863203)

[2.2. Targeted Attack-to-Benign Flipping 8](#_Toc196863204)

[2.3. Hybrid Poisoning (Attack-to-Benign + Benign-to-Attack) 8](#_Toc196863205)

[3. Part C: Preventing and Detecting Label Manipulation 10](#_Toc196863206)

[3.1. Existing Methods: 10](#_Toc196863207)

[3.1.1. Digital Signature for Data Integrity: 10](#_Toc196863208)

[3.1.2. Blockchain-Based Data Provenance: 10](#_Toc196863209)

[3.1.3. Anomaly Detection for Label Verification: 11](#_Toc196863210)

[3.1.4. Semi-Supervised Learning for Noisy Labels: 11](#_Toc196863211)

[3.2. Proposed Novel Approach: Streamlined Hybrid Label Verification Framework 12](#_Toc196863212)

[3.3. Block Diagram of Model Training and HLVF Deployment Pipeline 13](#_Toc196863213)

[4. References 15](#_Toc196863214)

# Part A: Intrusion Detection using XGBoost

## Model Selection Rationale:

For the intrusion detection task within the Edge-IIoTset dataset, XGBoost (Extreme Gradient Boosting) was selected due to its proven effectiveness in tabular data and its ability to handle class imbalance, which is common in cybersecurity datasets. XGBoost leverages an ensemble of decision trees optimized through gradient boosting, making it highly efficient and capable of capturing non-linear patterns within the dataset.

Compared to deep learning models such as LSTM or DNN, which are better suited for sequential or time-series data, XGBoost performs exceptionally well on structured data without the need for extensive feature engineering or scaling. Its robustness to outliers and missing values also makes it a practical choice for real-world intrusion detection tasks where data quality may vary.

Additionally, XGBoost offers interpretability through feature importance scores, allowing insight into which features contribute most to detecting cyber threats, a crucial factor in cybersecurity applications.

## Experimental Setup:

* Dataset Used: Edge-IIoTset (DNN-ready version).
* Preprocessing Steps:
  + Categorical Encoding: Label encoding using LabelEncoder for categorical features.
  + Missing Values Handling: Rows with invalid or missing entries (e.g., ‘?’) were dropped.
  + Train-Test Split: 70% training, 30% testing.
* Model Parameters:
  + Model: XGBoost Classifier (XGBoostClassifier from scikit-learn API)
  + Objective: Multi-class classification (multi:softmax)
  + Evaluation Metric: Multi-class log loss (mlogloss)
  + Number of Classes: Derived from the unique labels in the target column (Attack\_type)
* Evaluation Metrics:
  + Accuracy score
  + Precision score (weighted)
  + Recall score (weighted)
  + F1 score (weighted)
  + Confusion Matrix

## Results:

The XGBoost model was evaluated on the test set using the aforementioned metrics. The results are as follows:

* Accuracy: 0.9838
* Precision: 0.9840
* Recall: 0.9838
* F1-Score: 0.9838

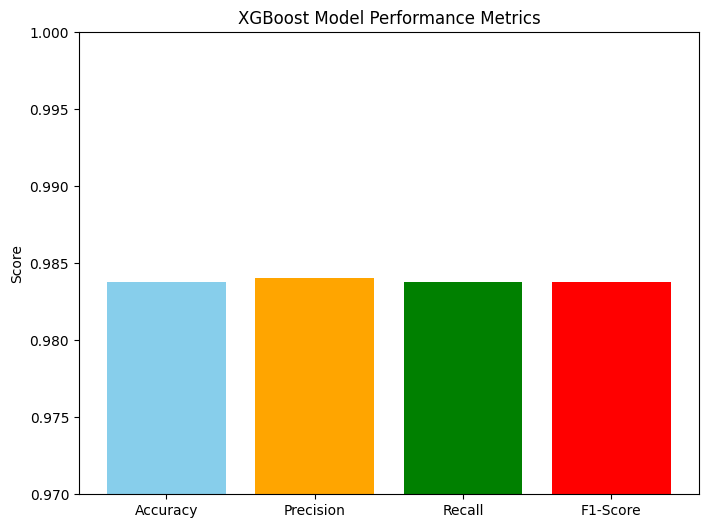


Figure 1-Bar graph representing model's performance metrics

The graph in figure 1 above displays the accuracy, precision, recall and F1 scores of the XGBoost model. The y-axis limits were set between 0.97 and 1 for better representation showing the slight differences between values.

* Confusion Matrix:

Figure 2 below shows that the model correctly classified the majority of attack classes with minimal misclassifications. However, there are slight confusions between similar attack types such as ‘SQL\_injection’ and ‘Password’.

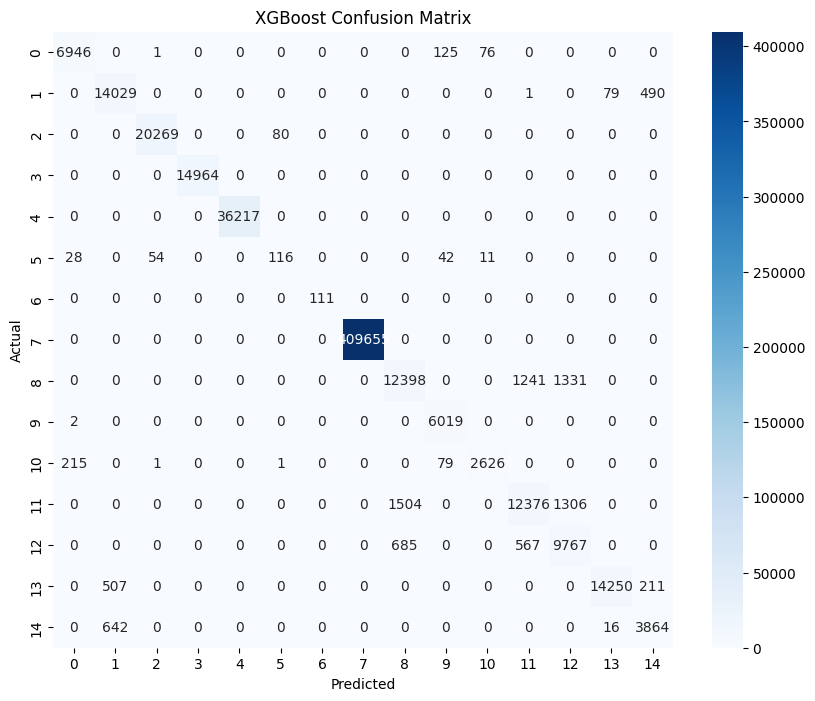


Figure 2-XGBoost confusion matrix

* Classification Report:

A screenshot of a computer screen

AI-generated content may be incorrect.

Figure 3-XGBoost's classification report

## Critical Evaluation:

As shown by the classification report in Figure 3 below; the XGBoost model demonstrated strong performance in classifying malicious activities within the IoT network, achieving an overall accuracy of 98% on the test set. The weighted-average precision, recall, and F1-score were all 0.98, indicating that the model performed well when considering the imbalanced class distribution inherent in intrusion detection datasets.

However, the macro-averaged metrics (Precision: 0.92, Recall: 0.91, F1-score: 0.91) reveal a slightly different picture. These metrics, which treat all classes equally, highlight that the model’s performance on minority classes was somewhat lower than on the majority classes. This distinction is crucial in cybersecurity contexts where rare but critical attack types (e.g., password attacks, vulnerability scanners, backdoor attacks) must be detected reliably, despite their lower occurrence.

* Key Observations:
  + Classes 5 (0.59 precision, 0.46 recall) and 12 (0.79 precision, 0.89 recall) showed lower recall and precision, indicating that the model struggled with accurately identifying these specific attack types.
  + These classes likely represent rare or complex attack scenarios, such as password attacks or vulnerability scans, where limited sample size or overlapping feature distributions may impact detection rates.
  + Majority classes like class 7 (normal traffic) achieved perfect precision and recall (1.00), which positively skews the weighted metrics but doesn’t fully capture the challenge of detecting less frequent attacks.

In conclusion, the XGBoost model is a highly effective choice for intrusion detection in IoT networks, delivering excellent accuracy and robust weighted metrics. While performance on majority classes is consistently strong, further work is required to improve detection rates for minority attack types, which are critical in cybersecurity applications. The macro-averaged metrics provide valuable insight into these areas, ensuring that rare threats are not overlooked in favour of more common traffic patterns.

# Part B: Data Poisoning and Its Impact on Intrusion Detection

Three distinct label poisoning strategies were implemented, and their effects on model’s performance were analysed across multiple poisoning levels (10%, 20%, 30%, 50%). The goal was to evaluate how these strategies degrade the model’s ability to detect cyber threats.

## Random Label Flipping

In this strategy, a random subset of labels across all classes was flipped to any other random class, introducing uniform noise across the dataset.

A graph with colored lines and numbers

AI-generated content may be incorrect.

Figure 4-XGBoost metrics after random label flipping

The plot in figure 4 above illustrates the impact of random label flipping on the model's performance across varying poisoning percentages. Despite flipping up to 50% of labels, accuracy remains consistently high (~98.4%), indicating that the model continues to correctly classify the majority benign traffic.

However, the macro-averaged recall and F1-score show a gradual decline as the poisoning percentage increases, with recall dropping from ~91.8% to ~90.7% at 50% flipping. Precision remains relatively stable or slightly increases, as random flipping does not specifically target the attack classes.

## Targeted Attack-to-Benign Flipping

This strategy specifically targeted attack labels, flipping them to the benign class (label 7). This approach aims to hide cyber threats by relabelling attacks as normal traffic, without altering benign labels.

A graph with colored lines and numbers

AI-generated content may be incorrect.

Figure 5-XGBoost metrics after attack-to-benign flipping

Figure 5 above illustrates the impact of targeted attack-to-benign label flipping on the model’s performance. At lower poisoning percentages (10%-20%), the performance metrics remain relatively stable. However, as the poisoning reaches 30% and especially 50%, there is a sharp decline in recall and F1-score, with recall dropping to ~43% and F1-score to ~57% at 50% flipping. Meanwhile, precision increases to ~96% because the model becomes conservative in predicting attacks (fewer false positives but many false negatives).

This strategy causes a significant drop in accuracy (~83.7% at 50%), unlike random flipping, due to the severe misclassification of attack traffic as benign. The collapse in recall and F1-score highlights the model’s failure to detect threats, even though precision and accuracy remain deceptively high.

## Hybrid Poisoning (Attack-to-Benign + Benign-to-Attack)

This hybrid approach combines attack-to-benign flipping with benign-to-attack noise injection, flipping half the specified percentage of attack labels to benign and the other half of benign labels to random attack classes. This approach introduces balanced noise into both the attack and benign classes.

A graph with colored lines and numbers

AI-generated content may be incorrect.

Figure 6-XGBoost metrics after hybrid label flipping

The graph in figure 6 above represents the model’s performance under the hybrid poisoning strategy. Across poisoning levels from 10% to 50%, accuracy remains consistently high (~98.3%), reflecting the model’s strong ability to classify the majority benign traffic. However, recall and F1-score gradually decline, with recall dropping to ~90.7% and F1-score to ~91.4% at 50% poisoning. Precision stays relatively stable or slightly improves, indicating the model maintains confidence in its predictions despite the added noise.

# Part C: Preventing and Detecting Label Manipulation

## Existing Methods:

### Digital Signature for Data Integrity:

Digital signatures are cryptographic constructs used to verify the authenticity and integrity of digital information.

* + How it works:
    - During data collection, each label is passed through a hashing function and encrypted with a private key (creating a digital signature).
    - Later, before training, each label's signature is verified using the corresponding public key.
    - If a label has been modified, the signature verification fails, immediately flagging the corrupted data.
  + Why is it important:
    - Ensures that labels have not been tampered with between collection and model training.
    - Especially crucial in outsourced or federated learning settings, where data passes through multiple untrusted intermediaries.
  + Limitations:
    - Does not prevent insider threats if insiders have access to signing keys.
    - Needs secure key management practices.

### Blockchain-Based Data Provenance:

Blockchain technology can secure the history and provenance of training data by offering immutable audit trails.

* + How it works:
    - Each label (or batch of labels) is hashed and stored in a block on a blockchain network.
    - Once committed, it cannot be modified without detection, ensuring full traceability of the data life cycle.
    - Any unauthorized modification will result in mismatched hashes when later verified.
  + Why is it important:
    - Immutable: Once data is recorded, it’s impossible to change retroactively.
    - Transparent: Auditors can review the full chain of custody for training data.
    - Decentralized: Trust is distributed, making insider attacks harder to execute.
  + Limitations:
    - Scalability issues for very large datasets.
    - Transaction costs if public blockchains are used.

### Anomaly Detection for Label Verification:

Anomaly detection methods can automatically spot unusual patterns in labels that might suggest label poisoning.

* How it works:
  + Model the statistical distribution of known good labels (e.g., percentage of attack vs benign labels).
  + Continuously monitor incoming data to detect sudden changes.
  + For example, if attack samples suddenly drop from 10% to 2% in a new training batch, it flags a potential manipulation.
* Techniques Used:
  + Clustering (K-Means, DBSCAN): Find clusters that don’t match the expected label distributions.
  + Density estimation (One-Class SVM, Isolation Forest): Detect samples that fall in low-density areas of known distributions.
  + Autoencoders: Deep learning models that can highlight unusual data by high reconstruction error.
* Advantages:
  + Works even when you don’t have ground-truth labels.
  + Can detect subtle poisoning attacks that change only a small percentage of data.
* Limitations:
  + False positives: Normal label fluctuations might be mistaken as anomalies.
  + Hard to tune thresholds in practice.

### Semi-Supervised Learning for Noisy Labels:

Semi-supervised learning (SSL) is a machine learning approach where models learn from a combination of a small amount of trusted labelled data and a large amount of unlabelled or weakly-labelled data. It is particularly useful in environments where labels might be noisy, unreliable, or maliciously manipulated, such as in cybersecurity datasets.

* How it works:
  + The model is first trained on a trusted subset of labelled data.
  + Then it uses consistency regularization or pseudo-labelling techniques to learn from unlabelled or noisy samples.
  + Confidence-based separation is often applied: the model trusts and trains harder on samples with confident predictions, and treats uncertain examples with caution.
  + Frameworks like DivideMix:
    - Split the data into "clean" (trusted) and "noisy" (suspected) groups during training.
    - Apply semi-supervised techniques to boost clean learning and suppress noisy labels.
* Why is it important:
  + Reduces the reliance on potentially poisoned labels, minimizing the attack surface.
  + Mitigates the impact of adversarial manipulation even if some labels are corrupted.
  + Boosts model robustness by allowing it to learn from the natural data distribution rather than blindly trusting all labels.
* Limitations:
  + Requires good estimation of which labels are noisy versus clean (confidence threshold tuning).
  + May still be vulnerable to sophisticated poisoning if attackers craft adversarial examples that mimic clean distributions.
  + Training complexity increases, needing careful balancing between labelled and unlabelled loss components.

## Proposed Novel Approach: Streamlined Hybrid Label Verification Framework

To efficiently protect machine learning pipelines from label manipulation, a streamlined Hybrid Label Verification Framework (HLVF) is proposed, integrating three essential defence layers. This design balances security, performance, and scalability, making it suitable for intrusion detection systems and similar high-stakes environments.

* Step 1: Digital Signature Verification

Ensures label authenticity before ingestion.

* Step 2: Blockchain-Based Provenance

Immutable tracking of label lifecycle.

* Step 3: Anomaly Detection Analysis

Statistical checks for label distribution consistency.

## Block Diagram of Model Training and HLVF Deployment Pipeline

# Part C: Preventing and Detecting Label Manipulation

## Existing Methods:

### Digital Signature for Data Integrity:

Digital signatures are cryptographic constructs used to verify the authenticity and integrity of digital information.

* + How it works:
    - During data collection, each label is passed through a hashing function and encrypted with a private key (creating a digital signature).
    - Later, before training, each label's signature is verified using the corresponding public key.
    - If a label has been modified, the signature verification fails, immediately flagging the corrupted data.
  + Why is it important:
    - Ensures that labels have not been tampered with between collection and model training.
    - Especially crucial in outsourced or federated learning settings, where data passes through multiple untrusted intermediaries.
  + Limitations:
    - Does not prevent insider threats if insiders have access to signing keys.
    - Needs secure key management practices.

### Blockchain-Based Data Provenance:

Blockchain technology can secure the history and provenance of training data by offering immutable audit trails.

* + How it works:
    - Each label (or batch of labels) is hashed and stored in a block on a blockchain network.
    - Once committed, it cannot be modified without detection, ensuring full traceability of the data life cycle.
    - Any unauthorized modification will result in mismatched hashes when later verified.
  + Why is it important:
    - Immutable: Once data is recorded, it’s impossible to change retroactively.
    - Transparent: Auditors can review the full chain of custody for training data.
    - Decentralized: Trust is distributed, making insider attacks harder to execute.
  + Limitations:
    - Scalability issues for very large datasets.
    - Transaction costs if public blockchains are used.

### Anomaly Detection for Label Verification:

Anomaly detection methods can automatically spot unusual patterns in labels that might suggest label poisoning.

* How it works:
  + Model the statistical distribution of known good labels (e.g., percentage of attack vs benign labels).
  + Continuously monitor incoming data to detect sudden changes.
  + For example, if attack samples suddenly drop from 10% to 2% in a new training batch, it flags a potential manipulation.
* Techniques Used:
  + Clustering (K-Means, DBSCAN): Find clusters that don’t match the expected label distributions.
  + Density estimation (One-Class SVM, Isolation Forest): Detect samples that fall in low-density areas of known distributions.
  + Autoencoders: Deep learning models that can highlight unusual data by high reconstruction error.
* Advantages:
  + Works even when you don’t have ground-truth labels.
  + Can detect subtle poisoning attacks that change only a small percentage of data.
* Limitations:
  + False positives: Normal label fluctuations might be mistaken as anomalies.
  + Hard to tune thresholds in practice.

### Semi-Supervised Learning for Noisy Labels:

Semi-supervised learning (SSL) is a machine learning approach where models learn from a combination of a small amount of trusted labelled data and a large amount of unlabelled or weakly-labelled data. It is particularly useful in environments where labels might be noisy, unreliable, or maliciously manipulated, such as in cybersecurity datasets.

* How it works:
  + The model is first trained on a trusted subset of labelled data.
  + Then it uses consistency regularization or pseudo-labelling techniques to learn from unlabelled or noisy samples.
  + Confidence-based separation is often applied: the model trusts and trains harder on samples with confident predictions, and treats uncertain examples with caution.
  + Frameworks like DivideMix:
    - Split the data into "clean" (trusted) and "noisy" (suspected) groups during training.
    - Apply semi-supervised techniques to boost clean learning and suppress noisy labels.
* Why is it important:
  + Reduces the reliance on potentially poisoned labels, minimizing the attack surface.
  + Mitigates the impact of adversarial manipulation even if some labels are corrupted.
  + Boosts model robustness by allowing it to learn from the natural data distribution rather than blindly trusting all labels.
* Limitations:
  + Requires good estimation of which labels are noisy versus clean (confidence threshold tuning).
  + May still be vulnerable to sophisticated poisoning if attackers craft adversarial examples that mimic clean distributions.
  + Training complexity increases, needing careful balancing between labelled and unlabelled loss components.

## Proposed Novel Approach: Streamlined Hybrid Label Verification Framework

To efficiently protect machine learning pipelines from label manipulation, a streamlined Hybrid Label Verification Framework (HLVF) is proposed, integrating three essential defence layers. This design balances security, performance, and scalability, making it suitable for intrusion detection systems and similar high-stakes environments.

* Step 1: Digital Signature Verification

Ensures label authenticity before ingestion.

* Step 2: Blockchain-Based Provenance

Immutable tracking of label lifecycle.

* Step 3: Anomaly Detection Analysis

Statistical checks for label distribution consistency.

## Block Diagram of Model Training and HLVF Deployment Pipeline

A diagram of a flowchart

AI-generated content may be incorrect.

Figure 7-Block diagram

The above diagram in figure 7 illustrates the proposed secure training pipeline incorporating the three-layered Hybrid Label Verification Framework (HLVF) designed to detect and prevent label manipulation:

* Data Collection: Raw data and labels are gathered from trusted or distributed sources.
* Digital Signature Generation: Each label is cryptographically signed using a private key, ensuring its authenticity from the moment of creation.
* Signature Verification: Before entering the training pipeline, every label undergoes signature verification using a public key. Any mismatches signal possible tampering.
* Blockchain Data Recording: Verified labels are recorded on a blockchain ledger, enabling immutable tracking of their origin and any access or modification events.
* Anomaly Detection Engine: As a second layer of defence, this component monitors incoming data for statistical anomalies — such as sudden changes in attack-to-benign ratios — that could indicate insider manipulation or stealthy poisoning.
* Model Training: Only data that passes all verification and anomaly checks is used to train the intrusion detection model, ensuring its robustness.

Model Deployment: The final trained model, built on secure and verified labels, is deployed for real-time or batch detection tasks.

Figure 7-Block diagram

The above diagram in figure 7 illustrates the proposed secure training pipeline incorporating the three-layered Hybrid Label Verification Framework (HLVF) designed to detect and prevent label manipulation:

* Data Collection: Raw data and labels are gathered from trusted or distributed sources.
* Digital Signature Generation: Each label is cryptographically signed using a private key, ensuring its authenticity from the moment of creation.
* Signature Verification: Before entering the training pipeline, every label undergoes signature verification using a public key. Any mismatches signal possible tampering.
* Blockchain Data Recording: Verified labels are recorded on a blockchain ledger, enabling immutable tracking of their origin and any access or modification events.
* Anomaly Detection Engine: As a second layer of defence, this component monitors incoming data for statistical anomalies — such as sudden changes in attack-to-benign ratios — that could indicate insider manipulation or stealthy poisoning.
* Model Training: Only data that passes all verification and anomaly checks is used to train the intrusion detection model, ensuring its robustness.
* Model Deployment: The final trained model, built on secure and verified labels, is deployed for real-time or batch detection tasks.

# References

* Truepic, 2023. *What is Content Integrity?* [online] Truepic. Available at: <https://www.truepic.com/blog/content-integrity?utm_source=chatgpt.com> [Accessed 29 Apr. 2025].
* Divyeshkumar, V., 2024. Block-chain based data provenance and integrity verification. *International Journal of Science and Research Archive*.
* OWASP, 2023. *ML02: Data Poisoning Attack – OWASP Machine Learning Security Top 10*. [online] OWASP. Available at: <https://owasp.org/www-project-machine-learning-security-top-10/docs/ML02_2023-Data_Poisoning_Attack?utm_source=chatgpt.com> [Accessed 29 Apr. 2025].
* Li, J., Socher, R. and Hoi, S.C., 2020. Dividemix: Learning with noisy labels as semi-supervised learning. *arXiv preprint arXiv:2002.07394*.