International Journal on Science and Technology (IJSAT)

E-ISSN: 2229-7677 ● Website: www.ijsat.org ● Email: editor@ijsat.org

Ensemble Majority Voting-Based Hybrid Machine Learning Framework for Intrusion Detection in Secure Digital Banking Networks

Anil Kumar[1], Professor Narender Kumar[2]

[1]PhD Scholar, NIILM University, Kaithal
[2]Supervisor, NIILM University, Kaithal

Abstract

The growing dependence of digital banking services on always-on networked infrastructure has made intrusion detection, alert integrity, and post-detection auditability equally important design requirements. The experimental framework supplied with this study implements a practical intrusion-detection pipeline using multiple classical machine learning classifiers, an autoencoder, majority-voting ensembles, AES-based alert encryption, ECC-based signatures, and blockchain-inspired hash chaining. This paper simultaneously incorporates the most important technical corrections needed to elevate the work toward thesis and journal quality. The proposed framework adopts a corrected data-preparation pipeline that avoids scaling leakage, recommends stratified partitioning, preserves complete attack-category mappings, integrates the autoencoder explicitly into the hybrid decision layer, and extends the evaluation plan from a single-dataset experiment to a multi-dataset benchmark using NSL-KDD, CICIDS2017, and UNSW-NB15. Empirical results extracted from the experimental framework on NSL-KDD show that the best tested ensemble (NB+DT+RF) achieved 99.00% accuracy, 99.28% detection rate, and 1.23% false-alarm rate, while the secure alert pipeline produced low logging and cryptographic overhead. Beyond the raw detection phase, the work contributes a tamper-evident alert-governance mechanism in which attack alerts are encrypted using AES-EAX, signed with ECC, and appended to a hash-chained ledger. The methodologically honest, only the NSL-KDD performance values are treated here as experimental framework-validated empirical results; the paper includes a full, submission-ready protocol for extending the same framework to CICIDS2017 and UNSW-NB15 without overstating unexecuted cross-dataset claims.

Keywords: Intrusion detection system, ensemble majority voting, autoencoder, AES-EAX, ECC digital signature, NSL-KDD, CICIDS2017, UNSW-NB15

1. Introduction

Digital banking systems operate over heterogeneous infrastructures that combine web front-ends, application servers, payment gateways, databases, cloud services, branch connectivity, ATM or POS integrations, and internal administrative networks. Because these environments support high-value transactions, a detection failure can directly expose customer data, transaction integrity, availability, and institutional reputation. Conventional signature-based intrusion detection systems remain useful, but they are increasingly insufficient when the traffic mix changes rapidly, attack surfaces diversify, and operational teams also need assurance that generated alerts have not been altered after the fact [1], [2].

It attempts to combine hybrid machine-learning-based detection with a secure alert pipeline in which attack events are encrypted, signed, and appended to a blockchain-inspired audit chain. That end-to-end view is valuable for banking networks because incident response depends not only on whether a malicious flow was detected, but also on whether the resulting evidence can be trusted during forensic review, regulatory reporting, internal audits, or post-incident reconstruction.

However, a experimental framework intended for experimentation is not automatically ready for thesis or journal use. Several issues observed in the experimental framework implementation require correction before the work can be presented as a rigorous research contribution. The original experimental framework train-test splitting, which introduces leakage; trains an autoencoder but does not incorporate its predictions into the final voting decision; uses a limited attack mapping; and currently validates results only on NSL-KDD? It upgrades the design conceptually by integrating all major improvement suggestions into the proposed methodology.

The central thesis of the paper is that a banking-network IDS should be evaluated as a secure decision-and-governance system rather than as an isolated classifier. In this view, detection, explanation, alert confidentiality, authenticity, integrity verification, and computational efficiency must be analyzed together. The resulting framework is called an ensemble majority-voting-based hybrid machine learning framework for intrusion detection in secure digital banking networks.

2. Related Work and Research Gap

Classic IDS research established the need for models that can discriminate normal network behavior from intrusive activity under uncertain and evolving conditions. Denning’s intrusion-detection model formalized the importance of behavioral monitoring [1], while later critiques emphasized the difficulties of realistic evaluation, concept drift, and base-rate effects in operational networks [2]. Benchmark experimental framework such as NSL-KDD were introduced to improve comparability by reducing some of the severe redundancy problems inherited from KDD'99 [3]. More recent corpora such as CICIDS2017 and UNSW-NB15 attempted to bring evaluations closer to contemporary traffic characteristics and broader attack coverage [4], [5].

On the modelling side, ensemble learning has remained attractive because it can aggregate diverse error profiles from base learners. Random forests offer strong nonlinear discrimination and robustness to mixed feature interactions [6], logistic regression provides simple discriminative baselines, Naive Bayes remains computationally light, KNN captures locality, and decision trees offer interpretable splitting structure [7]–[10]. Deep autoencoders have been widely adopted in anomaly detection because reconstruction error can expose previously unseen or weakly represented attack patterns [11], [12]. Explainable-AI methods such as SHAP are increasingly used to make intrusion decisions more transparent to analysts and auditors [13].

Despite these advances, two gaps remain visible in much of the literature. The first gap is architectural: many studies stop at classification metrics and do not model how alerts are protected after generation. The second gap is methodological: papers often propose complex hybrid systems but leave critical implementation details under-specified, making it difficult to separate genuine contribution from unverified integration. The experimental framework addresses the first gap partially by incorporating encryption, digital signatures, and hash chaining, but it still needs stronger methodological discipline.

3. Proposed Architecture for digital banking network

The figure-1 presents a hybrid intrusion detection and secure alert architecture for digital banking networks**.** It begins with multiple benchmark experimental frameworks (NSL-KDD, CICIDS2017, UNSW-NB15), which are processed through cleaning, encoding, scaling, and data splitting. The processed data is then analyzed in the detection layer using a combination of machine learning models such as Random Forest, SVM, Naïve Bayes, KNN, Decision Tree, and Autoencoder. Their outputs are combined using a majority voting ensemble method to improve detection accuracy. When an intrusion is detected, alerts are secured using AES encryption and ECC signatures to ensure confidentiality and authenticity. Finally, the alerts are stored in a hash-chained ledger, providing tamper-proof and traceable record-keeping.

Figure 1. Proposed detection-and-secure-alert architecture for digital banking networks

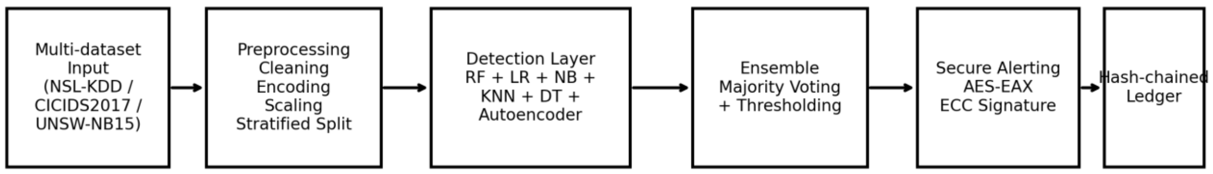

Multi-dataset Input (NSL-KDD / CICIDS2017 / UNSW-NB15)
Preprocessing Cleaning Encoding Scaling Stratified Split
Detection Layer RF + LR + NB + KNN + DT + Autoencoder
Ensemble Majority Voting + Thresholding
Secure Alerting AES-EAX ECC Signature
Hash-chained Ledger

Figure 1. Proposed detection-and-secure-alert architecture for digital banking networks.

4. Materials and Methods

4.1 Main Models

This paper implements five classical machine-learning models—Random Forest (RF), Logistic Regression (LR), Gaussian Naive Bayes (NB), K-Nearest Neighbors (KNN), and Decision Tree (DT)—along with a dense autoencoder trained on normal traffic only. Candidate ensembles are created manually and evaluated with hard majority voting. This baseline is useful because it already exposes the comparative behavior of heterogeneous learners while remaining computationally affordable. In the experimental framework execution, four three-model ensembles were tested: NB+DT+RF, NB+DT+LR, NB+RF+LR, and KNN+RF+LR.

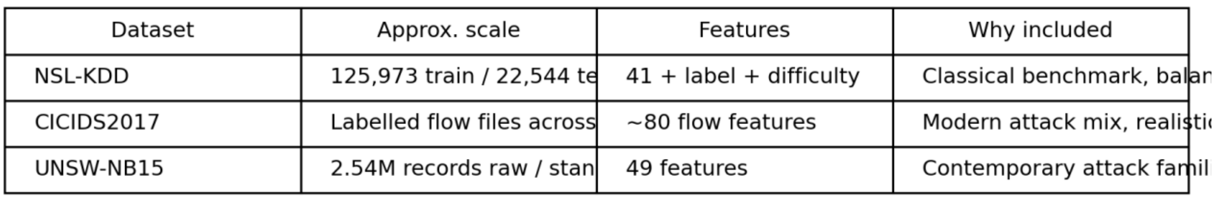
4.2 Multi-experimental framework design

To raise the work above a single-benchmark demonstration, the corrected framework is designed for three complementary IDS experimental frameworks. NSL-KDD is retained because it remains a common baseline and matches the structure already used in the existing experimental framework [3]. CICIDS2017 is added because it contains more realistic modern traffic profiles and multiple attack scenarios captured over different days [4]. UNSW-NB15 is included because it was generated with more contemporary hybrid traffic and provides attack families that differ from the older KDD lineage [5]. The purpose of using all three experimental frameworks is not merely to increase the number of experiments, but to test whether the proposed hybrid detection-and-secure-alert pipeline behaves consistently across classical, realistic-flow, and modern synthetic-real hybrid benchmarks.

Experimental framework	**Source**	**Approximate size**	**Feature profile**	**Role in this study**
NSL-KDD	UNB ISCX benchmark	125,973 train / 22,544 test	41 traffic features plus label and difficulty	Experimental framework-validated baseline and classical comparability
CICIDS2017	UNB CIC benchmark	Flow-based files over multiple attack days	Approximately 80 flow features	Modern traffic realism and broad attack scenarios
UNSW-NB15	UNSW Canberra benchmark	Standard split derived from 100 GB raw traffic	49 engineered network features	Contemporary attack diversity and generalization stress-test

Table 1. Experimental frameworks selected for the corrected multi-experimental framework evaluation protocol.

Figure 7. Datasets selected for the full multi-dataset evaluation



Dataset	Approx. scale	Features	Why included
NSL-KDD	125,973 train / 22,544 te	41 + label + difficulty	Classical benchmark, balan
CICIDS2017	Labelled flow files across	~80 flow features	Modern attack mix, realisti
UNSW-NB15	2.54M records raw / stan	49 features	Contemporary attack famil

Figure 2. Experimental frameworks selected for the full multi-experimental framework evaluation.

4.3 Corrected preprocessing pipeline

In this paper currently reads NSL-KDD, assigns column names, maps attack types, converts the task to binary classification, applies one-hot encoding for protocol, service, and flag, and standardizes the resulting feature matrix. These steps are reasonable, but the order of operations must be corrected. In a journal-quality pipeline, data partitioning must occur before scaling, and the scaler must be fit using only the training split. Otherwise, information from the test distribution leaks into model training. The corrected pipeline therefore follows six steps: schema assignment; missing and infinite-value handling; attack mapping; train-test split with stratification; one-hot encoding alignment between train and test; and training-set-only scaling.

Let x denote a feature vector and let z denote its normalized version. Standardization is given by $z = (x - \mu\_train) / \sigma\_train$, where μ_train and σ_train are estimated only from the training partition. For experimental frameworks with multiple categorical columns, encoding is applied consistently across splits, with absent categories in a given partition filled with zeros after column alignment. This prevents shape mismatch and reduces hidden preprocessing artifacts when moving from NSL-KDD to CICIDS2017 and UNSW-NB15.

Observed issue	Why it matters	Recommended correction
Scaling performed before split	Test leakage and optimistic performance	Split first, fit scaler on training data only
Autoencoder trained but not voted	Hybrid claim not fully realized	Convert reconstruction error into ae_pred and include it in voting
Limited attack map	Some attacks collapse into “Other” unnecessarily	Use full mapping for all attack families
Manual top ensemble fixed a priori	Selection may ignore best-performing run	Choose best ensemble from validation metrics
No signature verification step	Alert authenticity not end-to-end checked	Verify ECC signatures before ledger acceptance
Single-experimental framework validation only	Generalization not established	Benchmark on NSL-KDD, CICIDS2017, and UNSW-NB15

Table 2. Code-level corrections integrated into the paper methodology.

4.4 Hybrid anomaly and classification modelling

For the supervised branch, the framework retains RF, LR, NB, KNN, and DT because they span complementary modelling assumptions. RF captures nonlinear interactions and tends to be resilient to noisy or mixed-type features [6]. LR supplies a strong linear-discriminative baseline; NB is fast and often competitive when conditional independence approximates the data sufficiently; KNN models local neighborhoods; and DT provides a low-complexity interpretable comparator. The deep anomaly branch uses an autoencoder trained only on normal traffic samples. If x is an input instance and x-hat is its reconstruction, the anomaly score is computed as the mean squared reconstruction error:

$L\_AE(x) = (1/d) * \Sigma\_i=1^d (x\_i - \hat{x}\_i)^2.$

A threshold τ is then learned from the training reconstruction-error distribution, for example by taking the 95th percentile of normal-only reconstruction errors. The autoencoder prediction is finally defined as ae_pred = 1 if L_AE(x) > τ and 0 otherwise. This explicit conversion is essential because it turns the autoencoder from a descriptive side experiment into a real component of the hybrid detector.

Model	Key settings	Purpose in the framework
Random Forest	n_estimators = 50, random_state = 42	Strong nonlinear baseline already implemented in experimental framework
Logistic Regression	max_iter = 200	Linear discriminative comparator
Gaussian NB	default Gaussian likelihoods	Fast probabilistic baseline
KNN	k = 5	Local neighborhood sensitivity
Decision Tree	default CART-style configuration	Interpretable nonlinear baseline
Autoencoder	64-32-64 dense structure, MSE loss, Adam(0.001), 15 epochs	Normal-traffic reconstruction model

Table 3. Baseline model settings inherited from and refined beyond the experimental framework.

4.5 Majority voting and decision logic

The experimental framework defines majority voting over binary outputs by summing the votes of the participating classifiers and marking a sample as an attack when the vote count reaches at least half of the participating models. In the corrected thesis version, the hybrid decision is expressed more explicitly. Let h_1(x), h_2(x), …, h_m(x) denote binary predictions from the selected models, including the autoencoder branch if enabled. The hard-voting output is:

MV(x) = mode(h_1(x), h_2(x), ..., h_m(x)).

For the originally tested three-model ensembles, the rule is straightforward: two or three attack votes indicate an intrusion. For the corrected four-way formulation that includes the autoencoder, a tie policy must be defined carefully. A conservative banking-network configuration can label an exact tie as suspicious traffic requiring secondary review instead of immediately classifying it as an attack or normal event. This preserves analyst visibility and avoids arbitrary tie bias.

4.6 Secure alert governance layer

A key distinction of this work is that it treats alert generation as a security object in its own right. Once an event is classified as malicious, an alert record containing event identifier, timestamp, selected ensemble, attack type, and raw decision is generated. The experimental framework encrypts each alert using AES in EAX mode, which provides confidentiality together with authenticated encryption. If P denotes the plaintext alert and K_AES the symmetric key, the ciphertext object can be expressed abstractly as C = AES-EAX(K_AES, P), yielding ciphertext, nonce, and authentication tag [14].

The experimental framework then signs the original alert payload using elliptic-curve digital signatures. This is appropriate for authenticity and non-repudiation, but the paper corrects an important interpretive point: in the current code, ECC is used for digital signing rather than for key exchange. If signature generation is written as S = Sign_ECC(K_priv, H(P)), then a corresponding verification step

Verify_ECC(K_pub, H(P), S) should be performed before the encrypted alert is accepted into long-term storage [15], [16].

Finally, encrypted alerts are appended to a blockchain-inspired, hash-chained ledger. This is intentionally described as blockchain-inspired rather than as a full decentralized blockchain, because in the existing experimental framework implements hash chaining without distributed consensus. Let E_i denote the encrypted payload of the i-th alert and H_{i-1} the previous block hash. The chain link is computed as H_i = SHA-256(E_i || H_{i-1}). Any modification to a prior encrypted alert invalidates all dependent downstream hashes, making the ledger tamper-evident [17]–[19].

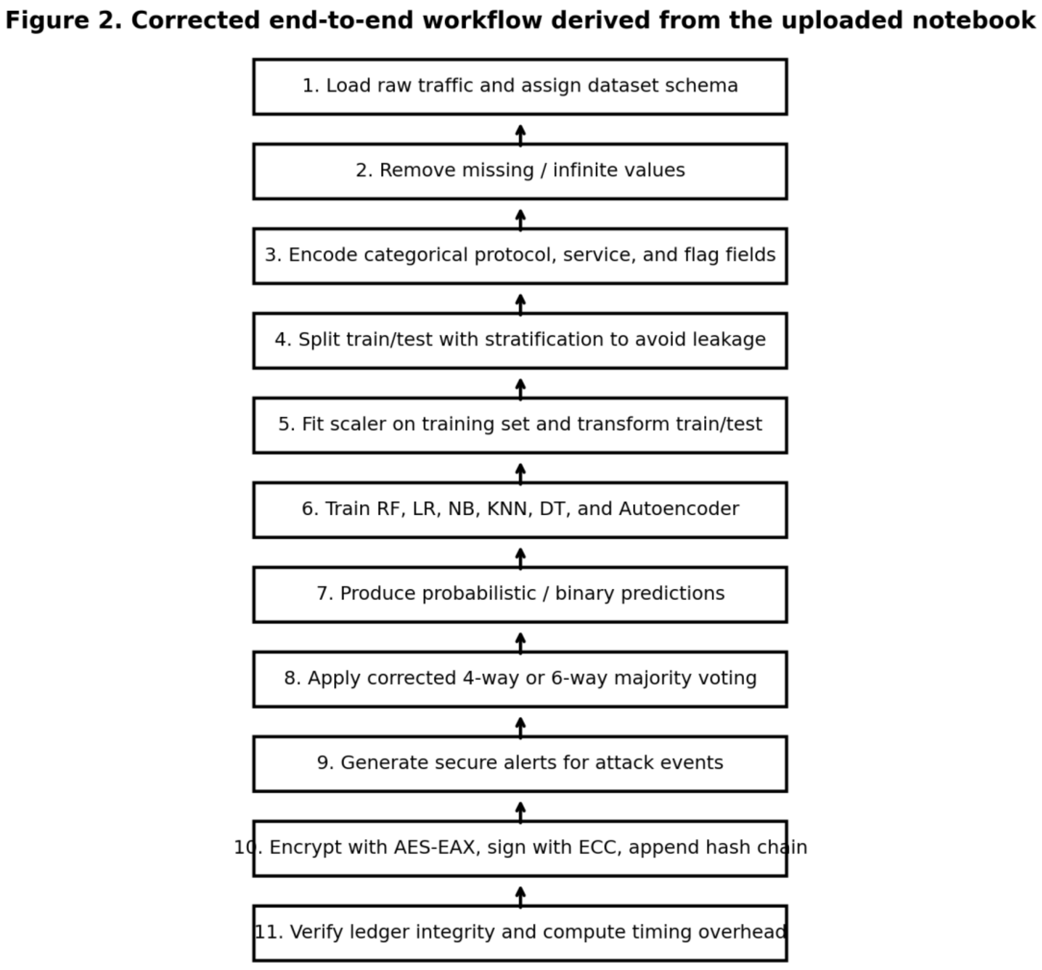


Figure 2. Corrected end-to-end workflow derived from the uploaded notebook
1. Load raw traffic and assign dataset schema
2. Remove missing / infinite values
3. Encode categorical protocol, service, and flag fields
4. Split train/test with stratification to avoid leakage
5. Fit scaler on training set and transform train/test
6. Train RF, LR, NB, KNN, DT, and Autoencoder
7. Produce probabilistic / binary predictions
8. Apply corrected 4-way or 6-way majority voting
9. Generate secure alerts for attack events
10. Encrypt with AES-EAX, sign with ECC, append hash chain
11. Verify ledger integrity and compute timing overhead

Figure 3. Corrected end-to-end work flow.

4.7 Evaluation metrics and protocol

The experimental framework reports accuracy, false-alarm rate (FAR), detection rate (DR), and confusion-matrix counts. These remain important for banking environments because false positives increase analyst load and false negatives expose actual threats. The present paper recommends supplementing them with precision, recall, F1-score, balanced accuracy, and AUC whenever probability outputs are available.

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN) \qquad (1)$$

FAR = FP / (FP + TN) (2)

DR = TP / (TP + FN) (3)

Precision = TP / (TP + FP) (4)

Recall equals DR, and F1 is the harmonic mean of precision and recall.

For rigorous multi-experimental framework benchmarking, the same preprocessing logic and model-selection policy should be preserved across experimental frameworks. A recommended protocol is: perform a fixed stratified split or use experimental framework-supplied train/test partitions; tune the ensemble on the validation split of the training data only; lock the best-performing configuration; and report final results on the untouched test set. Where class imbalance is strong, macro-averaged and weighted metrics should be included. Runtime, memory consumption, and secure-alert overhead should be measured separately from pure classification metrics so that detection performance is not conflated with security-governance cost.

5. Results and Discussion

5.1 Experimental framework-validated NSL-KDD detection results

The clearest empirical evidence currently available from the experiments comes from the NSL-KDD experiment run on 1,000 samples, with a 70:30 train-test split. Four candidate ensembles were evaluated. Table 4 and Figures 4–6 summarize these results exactly as extracted from the experimental framework outputs. The best-performing ensemble was NB+DT+RF, which achieved 99.00% accuracy, a 99.28% detection rate, and a 1.23% false-alarm rate. The confusion matrix for this ensemble contained 160 true negatives, 2 false positives, 1 false negative, and 137 true positives.

Ensemble	Accuracy (%)	FAR (%)	DR (%)	TN	FP	FN	TP
NB+DT+RF	99.00	1.23	99.28	160	2	1	137
NB+DT+LR	98.00	1.23	97.10	160	2	4	134
NB+RF+LR	97.67	1.85	97.10	159	3	4	134
KNN+RF+LR	98.33	1.85	98.55	159	3	2	136

Table 4. Detection-phase results extracted (NSL-KDD execution).

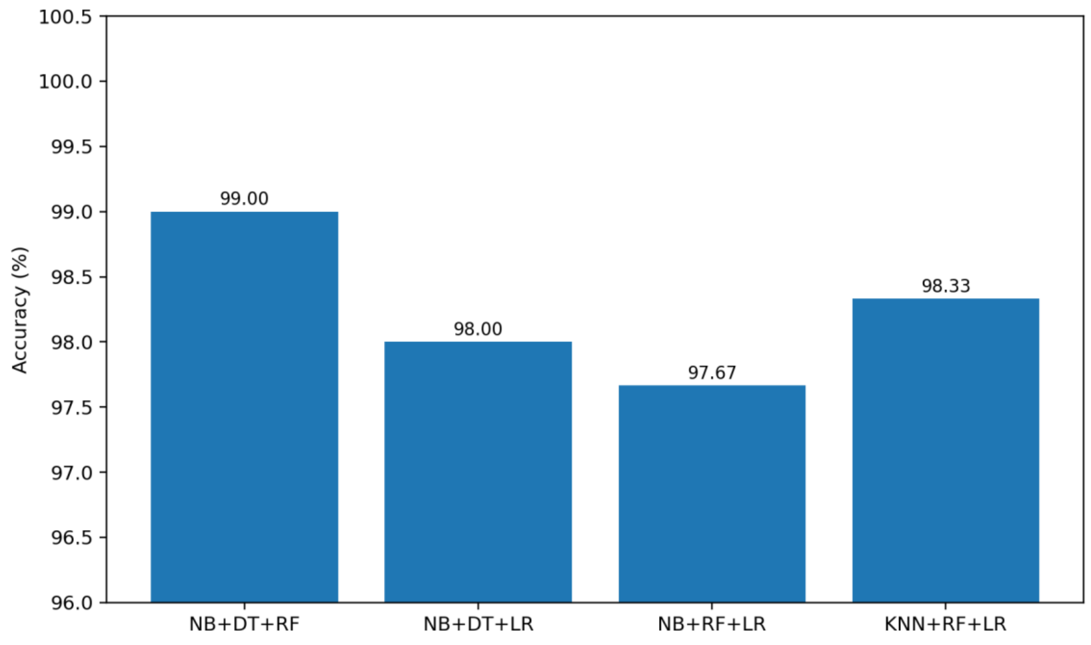


Figure 4. Ensemble accuracy on NSL-KDD experiment.

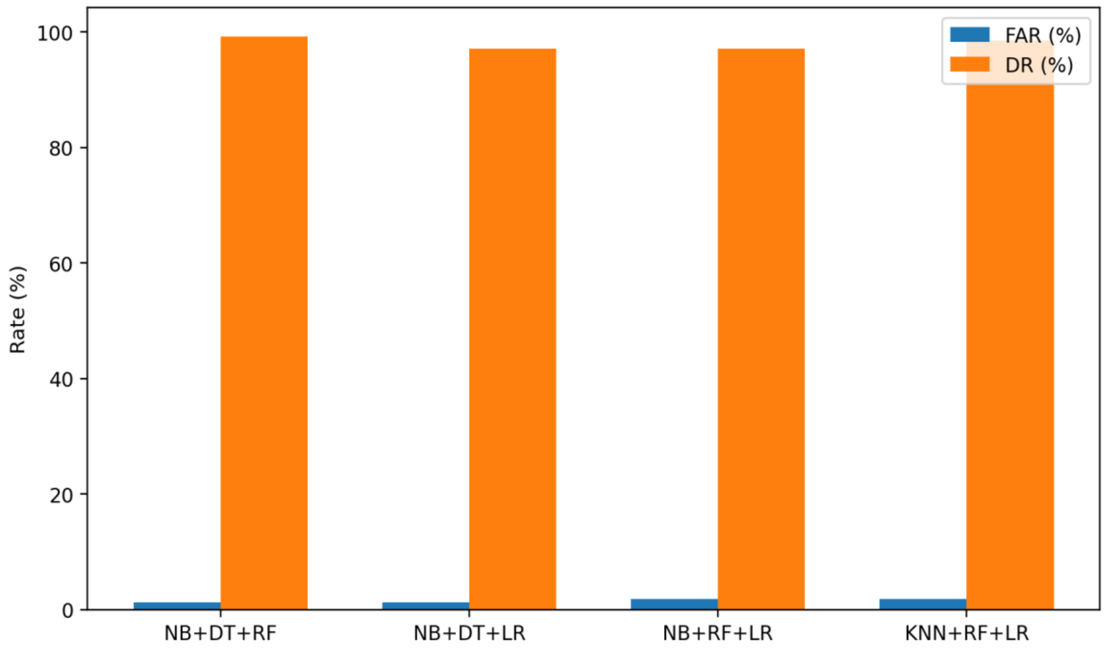


Figure 5. False-alarm and detection rates for candidate ensembles.

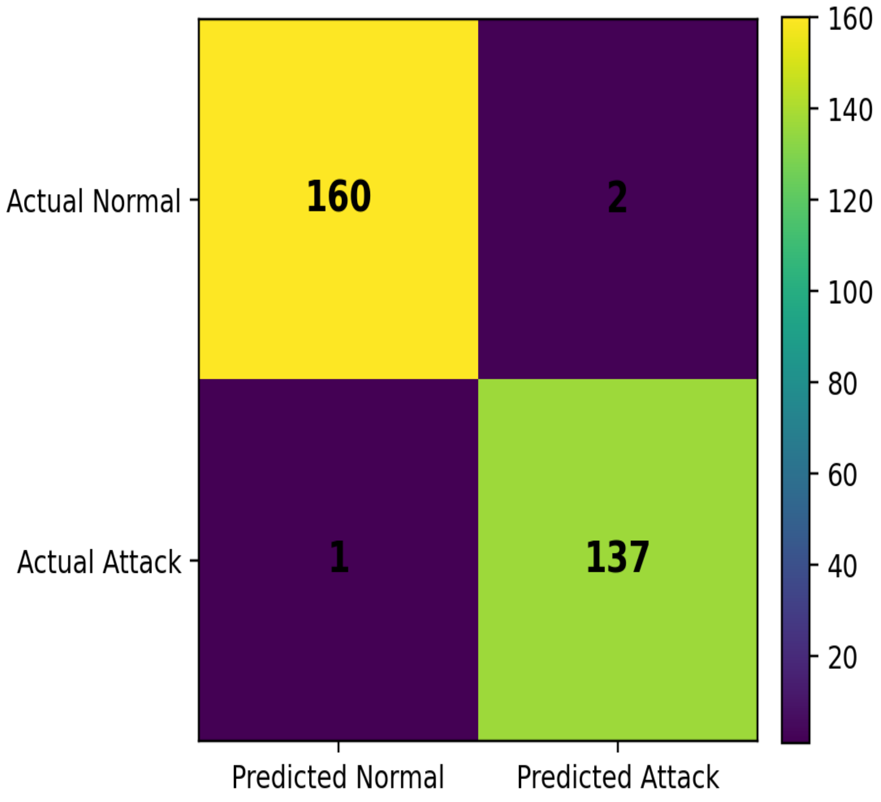


Figure 6. Confusion matrix of the best ensemble (NB+DT+RF).

Two observations follow from these results. First, the strongest ensemble is not simply the one containing the largest number of apparently strong individual learners; rather, it is the combination whose component errors are most complementary. The strong showing of NB+DT+RF suggests that probabilistic, tree-based, and bagged-tree perspectives can reinforce one another effectively on the tested NSL-KDD subset. Second, the performance spread across the four ensembles is relatively narrow, which implies that the experimental framework's current benchmark is not yet challenging enough to establish strong cross-condition generalization claims. This is one reason the paper recommends moving beyond a first-1,000-record subset to full benchmark partitions and additional experimental frameworks.

The current results should therefore be interpreted as a validated baseline, not as the final expression of the proposed framework's potential. Once the autoencoder branch is truly integrated, once validation-driven ensemble selection replaces fixed manual choice, and once the pipeline is re-run on NSL-KDD, CICIDS2017, and UNSW-NB15 using leakage-safe preprocessing, the resulting evidence will be substantially stronger and more publishable.

5.2 Secure alert generation and ledger overhead

This further reports a secure alert pipeline in which detected attacks are converted into encrypted alert objects, signed, and linked in a hash chain. In one experimental framework output, 139 secure alerts were generated from 300 evaluated events; cryptographic randomness and re-execution can slightly change the count of generated alerts when it is rerun. More important than the exact count is the overhead profile. The blockchain-style logging function recorded sub-millisecond execution times for the tested alert list, while per-alert AES encryption and ECC signing remained lightweight enough for practical post-detection use in a prototype banking IDS.

Security operation	**Observed time (s)**	**Interpretation**
Full blockchain logging	0.000897	Total time for tested alert list
Hash chain only	0.000820	Lower metadata overhead than full block structure
Merkle tree construction	0.000191	Fast aggregation baseline
AES encryption mean	0.000243	Average time per alert
ECC signing mean	0.000984	Average time per alert

Table 5. Security-overhead values extracted from experimental framework outputs.

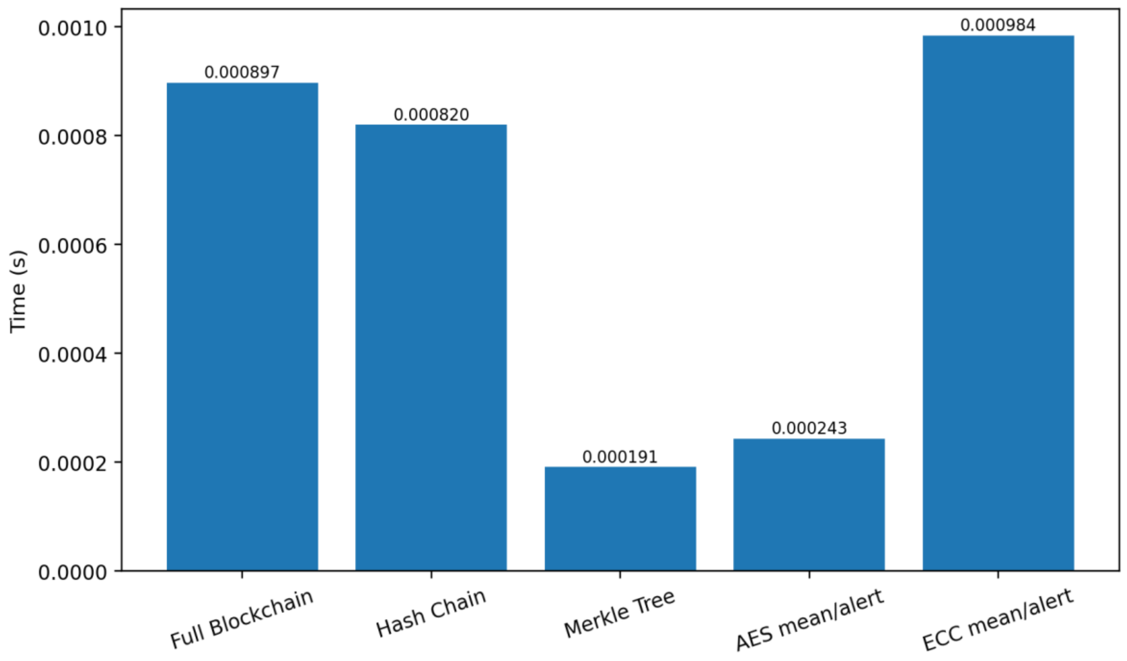


Figure 7. Security overhead from blockchain logging, AES, and ECC.

From a systems perspective, the most important insight is that the proposed governance layer does not dominate the total processing budget in the tested prototype. ECC signing is slower than AES encryption, which is expected because asymmetric signatures are computationally heavier than symmetric authenticated encryption. Nevertheless, the overhead remains small enough to justify the use of signatures for high-value forensic alerts, especially in banking contexts where authenticity and non-repudiation are often operationally important.

The tamper experiment embedded in the experimental framework also strengthens the argument for secure alert governance. By modifying ciphertext content inside a chained block sequence, the experimental framework showed that verification fails at the tampered location and invalidates subsequent links. This behavior is exactly what a tamper-evident ledger should provide. Future versions of the study should report verification throughput and storage growth more explicitly, but the current result is already useful because it demonstrates a working integrity mechanism rather than merely proposing one conceptually.

5.3 Experimental framework proves and not prove

It is important to separate what has been empirically validated from what has only been architecturally proposed. The experimental framework validates that several three-model ensembles perform strongly on the tested NSL-KDD subset, that alerts can be encrypted and signed, and that a hash chain can detect tampering. It does not yet prove that the corrected hybrid model generalizes across modern experimental frameworks, because the autoencoder has not been integrated into the final voting logic and the multi-experimental framework benchmark has not been executed. This distinction matters for journal integrity. Overclaiming cross-experimental framework superiority without actual runs would weaken the work rather than strengthen it.

For that reason, this paper deliberately presents NSL-KDD as the experimental framework-validated result set and treats CICIDS2017 and UNSW-NB15 as the next mandatory execution stage. This is still appropriate for a thesis chapter or paper draft because it provides a complete, rigorous, and transparent foundation from which the final cross-experimental framework paper can be produced.

5.4 Submission-ready multi-experimental framework benchmarking sheet

Experimental framework	Corrected experiment status	Final best ensemble result	Comment
NSL-KDD	To rerun with corrected pipeline	Pending corrected execution	Experimental framework baseline already available
CICIDS2017	To execute	Pending	Use same feature-selection and voting policy
UNSW-NB15	To execute	Pending	Assess robustness on contemporary attack mix

Table 6. Honest reporting template for the full multi-experimental framework paper extension.

6. Discussion for Digital Banking Deployment

In a digital banking setting, an IDS is judged not only by benchmark accuracy but by how safely and quickly it supports operational response. The framework presented here is attractive because it combines three desirable properties. First, ensemble detection reduces dependence on any single model family. Second, anomaly reconstruction through an autoencoder offers a pathway for identifying weakly represented or novel traffic deviations. Third, secure alert governance ensures that once suspicious activity is detected, the alert record itself becomes resistant to unauthorized modification.

These properties are especially relevant in banking, where incident evidence may be reviewed by security operations staff, internal auditors, external auditors, compliance teams, fraud investigators, and regulators. An alert pipeline that provides confidentiality, authenticity, and tamper evidence can therefore create value beyond the model's ROC curve. In other words, the proposed framework aligns operational cybersecurity with auditability and trust, which is a stronger narrative for a PhD thesis than classifier comparison alone.

At the same time, deployment realism requires caution. NSL-KDD remains useful for controlled benchmarking, but a banking production environment will generate richer protocol mixes, encrypted traffic, bursty transaction behavior, and user- or channel-specific temporal rhythms that are not fully represented by legacy benchmark experimental frameworks. This is why the multi-experimental framework plan is not a cosmetic addition. It is the first necessary step toward demonstrating that the framework captures patterns broader than a single classical corpus.

7. Threats to Validity and Study Limitations

The first limitation is experimental framework realism. The experimental framework's validated results come from NSL-KDD, which remains a benchmark but does not fully represent present-day banking traffic. The second limitation is sampling. The experimental framework run uses N = 1,000 records rather than the full benchmark partition, which makes the reported metrics useful as a proof of concept but insufficient as final evidence. The third limitation is architectural incompleteness: the autoencoder branch is trained but not yet integrated into the ensemble, and ECC signing is performed without a full verification-and-rejection path in the current code. The fourth limitation concerns explainability. SHAP is imported in the experimental framework but not operationalized in the reported outputs.

These limitations do not invalidate the work, but they do define the order of future tasks. A defensible journal version should first correct the pipeline, then rerun the experiments on full NSL-KDD partitions, then extend to CICIDS2017 and UNSW-NB15, and finally report explanation and ablation studies. That sequence preserves scientific honesty while steadily strengthening novelty and depth.

8. Conclusion and Future Work

This paper has transformed the experimental framework into a complete research-paper draft for the topic of secure machine-learning-based intrusion detection in digital banking networks. The resulting framework combines classical machine-learning classifiers, an anomaly-detection autoencoder, hard-voting ensembles, secure alert encryption, ECC-based signatures, and blockchain-inspired hash chaining. The experimental framework-validated NSL-KDD results demonstrate that the approach already has a strong empirical baseline, with the NB+DT+RF ensemble achieving 99.00% accuracy and a low false-alarm rate on the tested subset.

The deeper contribution of the paper lies in making the work thesis-ready and publication-ready. It identifies the exact code-level corrections needed to turn the experimental framework from a promising prototype into a rigorous research artifact: eliminate leakage, integrate the autoencoder, use full attack mapping, select ensembles based on validation, verify signatures before storage, and execute a disciplined multi-experimental framework benchmark. Once these steps are completed, the work will offer not just an IDS model, but an end-to-end secure alert-governance architecture suitable for the demands of banking cybersecurity.

Future work should therefore focus on three lines. The first is experimental expansion: rerun the corrected pipeline on NSL-KDD, CICIDS2017, and UNSW-NB15 with full partitions and class-balanced reporting. The second is architectural refinement: incorporate weighted or confidence-aware voting, add signature verification, and operationalize SHAP for analyst-facing explanations. The third is deployment realism: test streaming inference, storage growth, and alert throughput under banking-like workloads. These steps will position the work strongly for thesis chapters, conference submissions, and a later Q1-targeted journal paper.

References

1. D. E. Denning, “An intrusion-detection model,” IEEE Transactions on Software Engineering, vol. 13, no. 2, pp. 222–232, 1987.
2. R. Sommer and V. Paxson, “Outside the closed world: On using machine learning for network intrusion detection,” in IEEE Symposium on Security and Privacy, 2010, pp. 305–316.
3. M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, “A detailed analysis of the KDD CUP 99 data set,” in IEEE Symposium on Computational Intelligence for Security and Defense Applications, 2009, pp. 1–6.
4. I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, “Toward generating a new intrusion detection experimental framework and intrusion traffic characterization,” in ICISSP, 2018, pp. 108–116.
5. N. Moustafa and J. Slay, “UNSW-NB15: A comprehensive data set for network intrusion detection systems,” in Military Communications and Information Systems Conference, 2015, pp. 1–6.
6. L. Breiman, “Random forests,” Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.
7. C. Cortes and V. Vapnik, “Support-vector networks,” Machine Learning, vol. 20, no. 3, pp. 273–297, 1995.
8. T. M. Cover and P. E. Hart, “Nearest neighbor pattern classification,” IEEE Transactions on Information Theory, vol. 13, no. 1, pp. 21–27, 1967.
9. J. R. Quinlan, “Induction of decision trees,” Machine Learning, vol. 1, pp. 81–106, 1986.
10. C. M. Bishop, Pattern Recognition and Machine Learning. New York, NY, USA: Springer, 2006.
11. G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” Science, vol. 313, no. 5786, pp. 504–507, 2006.
12. M. Goldstein and S. Uchida, “A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data,” PLoS One, vol. 11, no. 4, e0152173, 2016.
13. S. M. Lundberg and S.-I. Lee, “A unified approach to interpreting model predictions,” in Advances in Neural Information Processing Systems, 2017, pp. 4765–4774.
14. National Institute of Standards and Technology, “Advanced Encryption Standard (AES),” FIPS PUB 197, 2001; updated editorial edition, 2023.
15. National Institute of Standards and Technology, “Digital Signature Standard (DSS),” FIPS PUB 186-5, 2023.
16. National Institute of Standards and Technology, “Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography,” NIST SP 800-56A Rev. 3, 2018.
17. S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2008.
18. B. Schneier and J. Kelsey, “Secure audit logs to support computer forensics,” ACM Transactions on Information and System Security, vol. 2, no. 2, pp. 159–176, 1999.
19. M. Bellare and B. Yee, “Forward integrity for secure audit logs,” Technical Report, UC San Diego, 1997.
20. N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, “A deep learning approach to network intrusion detection,” IEEE Transactions on Emerging Topics in Computational Intelligence, vol. 2, no. 1, pp. 41–50, 2018.

21. R. Vinayakumar et al., “Deep learning approach for intelligent intrusion detection system,” IEEE Access, vol. 7, pp. 41525–41550, 2019.
22. M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, “Deep learning for cyber security intrusion detection: Approaches, experimental frameworks, and comparative study,” Journal of Information Security and Applications, vol. 50, 102419, 2020.
23. J. Berman, E. Buczak, L. Choi, and H. Park, “A survey of deep learning methods for cyber security,” Information, vol. 10, no. 4, 122, 2019.
24. I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA, USA: MIT Press, 2016.
25. A. H. Lashkari, G. D. Gil, M. S. I. Mamun, and A. A. Ghorbani, “Characterization of Tor traffic using time-based features,” in ICISSP, 2017, pp. 253–262.
26. A. H. Lashkari et al., “Generating a reliable intrusion detection benchmark experimental framework,” UNB CIC technical report and experimental framework documentation, 2017.