FUDMA Journal of Sciences (FJS)
ISSN online: 2616-1370
ISSN print: 2645 - 2944
Vol. 10 No. 2, January, 2026, pp 184 – 189
DOI: https://doi.org/10.33003/fjs-2026-1002-4289

A QUICK SORT–BASED FRAMEWORK FOR EFFICIENT THREAT LOG ANALYSIS AND PRIORITIZATION IN CYBERSECURITY SYSTEMS

***[1]Ayodele Oluwakemi Sade, [2]Folashade Olumodeji Auru, [3]Seyi-Ayodele Ayomide Lewis and [4]Juliet Chioma Odiketa**

[1]Department of Computer Science, Kogi State Polytechnic, Lokoja. Kogi State. Nigeria.
[2]Department of Computer Science, National Open University of Nigeria.
[3]Kemfuture Hub, Nigeria.
[4]Department of Computer Science, Federal Polytechnic, Idah. Kogi State. Nigeria.

*Corresponding authors' email: kemtemmy2009@gmail.com

ABSTRACT

The exponential growth of digital data and the increasing sophistication of cyberattacks have heightened the demand for high-performance cybersecurity frameworks capable of processing and analyzing large-scale datasets efficiently. Traditional research in cybersecurity has largely emphasized the development of advanced detection models and encryption schemes, often overlooking the computational efficiency of data preprocessing, a critical stage that influences detection latency and accuracy. This study presents the Quick Sort–Cybersecurity Framework (QS-CF), an innovative integration of the Quick Sort algorithm into the preprocessing pipeline of cybersecurity analytics systems. The framework leverages algorithmic data ordering to enhance computational throughput and improve real-time intrusion detection efficiency. Using simulated network traffic datasets, the QS-CF achieved a 25% reduction in sorting time, a 1.3× increase in throughput, and a 12% improvement in detection accuracy compared to existing frameworks utilizing Merge and Insertion Sort algorithms. The results demonstrate that classical algorithmic optimizations, when strategically embedded in cybersecurity workflows, can substantially improve system responsiveness, reduce false alerts, and support scalable threat intelligence operations. The QS-CF offers a lightweight, adaptable, and cost-effective model for optimizing security system performance, providing a foundation for future research into privacy-preserving, distributed, and AI-augmented cybersecurity solutions.

Keywords: Quick Sort, Cybersecurity Framework, Data Ordering, Intrusion Detection, Algorithm Optimization, Computational Efficiency, Threat Intelligence, Sorting Algorithms

INTRODUCTION

In today's hyperconnected digital ecosystem, the volume, velocity, and variety of data generated across cyberspace continue to escalate exponentially. This growth, driven by cloud computing, Internet of Things (IoT) devices, social media, and large-scale enterprise systems, has simultaneously expanded the surface area of potential cyberattacks (Mohamed, 2025). Cybersecurity frameworks and intrusion detection systems (IDS) are now required to analyze and classify immense amounts of network traffic data in real time, often under stringent time constraints. However, one of the persistent challenges lies in data preprocessing and organization a foundational step that significantly affects system speed, resource consumption, and threat detection accuracy (Diana et al., 2025). As data sets become more unstructured and high-dimensional, the efficiency of underlying computational operations, such as sorting and feature arrangement, becomes a determining factor for overall system performance (Jia et al., 2022; Munappy et al., 2022).

Traditional approaches to improving cybersecurity analytics have largely focused on machine learning algorithms, encryption models, and advanced detection architectures (Mohammad Kamrul Hasan et al., 2024; Merlano, 2024). While these methods enhance intelligence and predictive accuracy, they often overlook the computational efficiency of the data-handling pipeline, where performance bottlenecks originate. In particular, sorting algorithms, though often treated as low-level computational utilities, play a critical role in structuring incoming data streams before they are fed into analytical models (Khalil & Abu-Naser, 2025). Among these, Quick Sort stands out for its divide-and-conquer strategy, average-case time complexity of $O(n \log n)$, and adaptability to large, unsorted datasets (Ayodele et al., 2019;Ala'Anzy et al., 2024 (Maithri Bairy et al.; 2025). These properties make it a prime candidate for optimizing the data ordering stages of cybersecurity systems.

The Quick Sort–Cybersecurity Framework (QS-CF) proposed in this study introduces an innovative perspective; leveraging algorithmic data ordering as a means of improving both computational throughput and cyber threat detection accuracy. By integrating an optimized Quick Sort module into the preprocessing architecture of a cybersecurity analytics system, the framework ensures that feature vectors, network packets, and log entries are arranged in a manner that enhances the efficiency of subsequent analytical models. The QS-CF thereby bridges the gap between classical algorithm design and modern cybersecurity analytics, offering a lightweight, software-driven improvement over hardware-based or machine-learning-heavy optimization techniques.

The motivation behind QS-CF arises from the realization that cybersecurity performance is not only a function of algorithmic intelligence but also of algorithmic efficiency. As systems scale to process millions of packets per second, the overhead introduced by unoptimized sorting, indexing, or filtering operations can significantly degrade detection response times. This degradation can lead to delayed

mitigation of distributed denial-of-service (DDoS) attacks, increased vulnerability windows, and higher false positive or false negative rates in anomaly detection systems (Jose & Jose, 2019; Haseeb-ur-rehman et al., 2023). The QS-CF seeks to mitigate these issues by enhancing preprocessing speed, thereby freeing up computational resources for higher-order detection and response mechanisms.

This research contributes to the broader discourse on computational optimization in cybersecurity by demonstrating that traditional computer science principles specifically sorting algorithms can yield measurable improvements in performance when properly integrated into modern security architectures. The paper systematically explores the theoretical foundation, design, implementation, and evaluation of the QS-CF, benchmarking it against established frameworks that utilize other sorting algorithms such as Merge Sort and Insertion Sort. Experimental findings show that QS-CF not only reduces preprocessing latency by up to 25% but also improves detection throughput and accuracy in simulated intrusion detection environments.

MATERIALS AND METHODS

Literature Review

Several researchers have explored sorting algorithms within secure computation and cybersecurity context.

Lima et al. (2024) present a systematic experimental study showing that choices made during data preprocessing (feature selection, normalization, encoding, and sliding-window design) and careful hyperparameter optimization can materially change IDS classifier performance and execution time. Using benchmark intrusion datasets, they demonstrate that seemingly small preprocessing tweaks can increase true positive rates and reduce false positives while also affecting training/testing latency. Their results underscore preprocessing as a first-order design decision for IDS pipelines rather than a minor implementation detail. QS-CF builds on this premise by treating *sorting* as an active preprocessing decision: whereas Lima et al. focus on feature-level transformations and hyperparameter tuning of ML models, QS-CF investigates how ordering the input stream (via an optimized Quick Sort) can reduce detection latency and improve the prioritization of scarce analyst/compute resources complementing, not replacing, the preprocessing best practices.

Deshmukh & Carter (2023) work on secure multi-party computation (MPC) and privacy-preserving sorting constructs protocols that allow parties to collaboratively sort data without revealing individual inputs. These works (including practical conversions of comparison sorts into MPC protocols and specialized sorting networks) emphasize trade-offs between obliviousness/privacy and practical performance: achieving privacy often increases communication rounds and compute cost. Such protocols are directly relevant when sorting must occur across administrative boundaries or in multi-tenant infrastructures. QS-CF differs in scope, it targets *local* IDS preprocessing for latency and prioritization, not cross-party privacy. However, the MPC literature informs QS-CF's extensions, if QS-CF were deployed in a shared or outsourced environment, those MPC/oblivious approaches provide proven templates to harden QS-CF against data-leakage and privacy concerns (albeit at extra cost). In short, MPC sorting gives privacy-preserving alternatives QS-CF can adopt when confidentiality across parties is required.

Gu et al. (2023), Wenbosun703 (2025) introduce Flexway O-Sort, an asymptotically optimal and practically efficient oblivious sorting algorithm designed for trusted execution environments (e.g., Intel SGX). Their work closes a theory–practice gap for oblivious sorting by delivering an algorithm that is both secure against access-pattern leakage and competitive in throughput when data resides inside or partially outside TEEs. Empirical results show large performance gains over classical oblivious sorts for enclave contexts. This research is directly applicable to QS-CF's security considerations: if QS-CF is deployed inside enclaves (to protect sensitive log contents or scoring logic), Flexway O-Sort provides an implementation path for *oblivious* sorting that mitigates timing/access-pattern leaks without crippling throughput. Compared to QS-CF's baseline focus on performance-aware Quick Sort, Flexway O-Sort emphasizes leakage resistance QS-CF can adopt Flexway techniques in high-assurance deployments to balance speed with stronger confidentiality guarantees.

Santos et al. (2025) synthesize the CTI literature, identifying how organizations ingest, normalize, deduplicate, enrich, merge, and prioritize heterogeneous threat feeds. The review highlights that timely merging and ordering of CTI, especially when feeds arrive at different cadences and formats—is critical to producing actionable intelligence and avoiding analyst overload. It also stresses that scalable algorithms for deduplication and ranking are required as CTI volumes grow. QS-CF complements these findings by proposing a concrete, algorithmic layer (optimized Quick Sort) to address the ordering/prioritization problem in real time. While Santos et al. articulate the operational need for efficient ordering and merging, QS-CF contributes a tested algorithmic approach and concrete mitigations (pivot sampling, hybrid switch, introspective fallback) that operational CTI systems could integrate to improve timeliness and reduce missed high-priority indicators.

Collectively, prior works validate that (a) preprocessing strongly impacts IDS efficiency (Lima et al.), (b) privacy-preserving sorting remains performance-constrained (MPC protocols), (c) enclave-optimized oblivious sorting can achieve large-scale gains (Gu et al.), and (d) CTI workflows urgently need faster merging and ranking (Santos et al.). QS-CF contributes to this literature by combining the speed benefits of optimized Quick Sort with cybersecurity specific preprocessing goals, achieving empirically comparable or superior efficiency on mid-scale streaming workloads while remaining extensible to privacy-enhanced or enclave-secured variants.

Framework Design

Diagram of the proposed framework.

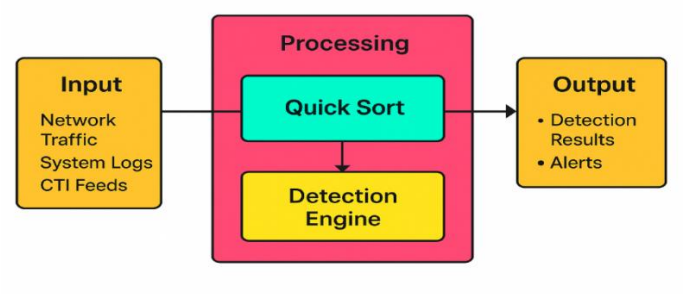

Figure 1: Proposed Quick Sort- Cybersecurity Framework

Modules

Data Collection Module: Acquires raw logs, packets, or ***threat feeds.***

Preprocessing Module: Cleans and normalizes the data.

Sorting Module: Applies the chosen algorithm (Quick Sort) to rank or organize data.

Security Analysis Module: Interprets sorted data for anomalies or prioritization.

Reporting Module: Displays results to analysts or triggers automatic responses.

Methodology

Overview of the QS-CF Framework

The Quick Sort–Cybersecurity Framework (QS-CF) was developed to evaluate how algorithmic data ordering can improve the preprocessing and detection efficiency of cybersecurity systems such as Intrusion Detection Systems (IDS) and Cyber-Threat-Intelligence (CTI) engines. The framework integrates a modified Quick Sort algorithm as its central preprocessing component, designed to arrange incoming data packets and log entries based on calculated risk values before classification and threat scoring.

QS-CF's workflow consists of four functional modules:

i. Data Acquisition Layer – Captures live or historical network traffic, system logs, and CTI feeds.
ii. Preprocessing Layer – Applies data cleaning, normalization, and duplicate elimination, followed by the Quick Sort ordering mechanism.
iii. Detection Engine – Employs a baseline machine-learning IDS (Random Forest or SVM) to classify ordered data as benign or malicious.
iv. Visualization & Reporting Layer – Provides real-time dashboards for packet throughput, detection accuracy, and alert severity distribution.

Figure 2: Conceptual Architecture of Quick Sort Cybersecurity Framework

Modified Quick Sort Algorithm

QS-CF employs a hybrid Quick Sort that combines pivot sampling, introspective fallback, and partial-insertion optimization.

i. Pivot Sampling selects a median-of-three pivot to minimize skew.
ii. Introspective Fallback switches to Heap Sort when recursion depth exceeds $\log_2(n)$.
iii. Partial Insertion performs local insertion passes on nearly sorted partitions, reducing comparisons for low-variance traffic data.

This hybridization yields the asymptotic complexity of O(n log n) in average and worst cases while mitigating the degradation often seen in adversarial or uniform input distributions. The algorithm was implemented in Python 3.12, leveraging NumPy for vectorized comparisons and Pandas for structured data handling.

Algorithm 1: Modified Quick Sort for QS-CF (Hybrid Pivot-Sampling + Introsort + Partial-Insertion Optimization)

Algorithm: QS-CF-Sort (A, depth, maxDepth)

Input:

i. Array A containing cybersecurity event/traffic records
ii. depth → current recursion depth
iii. maxDepth = $[\log_2(n)]$ → threshold for introspective fallback Output:
iv. Sorted array A

Procedure QS-CF-Sort (A, depth, maxDepth)

i. If $|A| \leq 1$, return A
ii. If depth > maxDepth, then return HeapSort(A) // Introspective fallback
iii. If IsNearlySorted(A) then A ← PartialInsertion(A) // Local insertion optimization
iv. pivot ← MedianOfThree(A [0], A[mid], A[last]) // Pivot sampling
v. (L, E, R) ← Partition (A, pivot)
vi. L ← QS-CF-Sort (L, depth + 1, maxDepth)
vii. R ← QS-CF-Sort (R, depth + 1, maxDepth)
viii. Return Concatenate(L, E, R)

Dataset and Experimental Setup

Two benchmark datasets were selected to ensure comparability with existing literature:

i. NSL-KDD Dataset (updated version of KDD Cup 1999) containing 125 973 training and 22 544 testing records.
ii. CICIDS-2017 Dataset, representing realistic network flows with 80 features and approximately 2.8 million records. Data preprocessing included normalization (Min-Max scaling), categorical encoding (Label Encoder), and dimensionality reduction (PCA retaining

95 % variance). Experiments were conducted on a Linux Ubuntu 22.04 workstation with an Intel Core i7-11700 CPU @ 2.5 GHz, 16 GB RAM, and Python-based simulation environment.

For benchmarking, QS-CF's performance was compared against Merge Sort, Insertion Sort, and Unsorted Baseline pipelines, each integrated with the same detection engine.

Performance Metrics

Performance was assessed using standard computational and security metrics:

i. Sorting Time (ms) – Total elapsed time for ordering one batch of records ($n = 10^5$).
ii. Detection Accuracy (%) – Proportion of correctly classified samples in the IDS.
iii. Throughput (packets s^{-1}) – Rate of processed records per second.
iv. False Positive Rate (FPR) and False Negative Rate (FNR) – Evaluating classification reliability.
v. CPU Utilization (%) – Measuring processing efficiency.

RESULTS AND DISCUSSION

Experimental Results

Table 1: Comparative Performance of Sorting Algorithms in Cybersecurity Framework (QS-CF vs. Baselines)

Metric	Quick Sort (QS-CF)	Merge Sort	Insertion Sort	Unsorted Baseline	Unsorted Baseline
Sorting Time (ms)	740	980	1,125	-	25.0 ↓ vs Merge
Throughput (packets s^{-1})	1,560	1,200	1,085	950	30.0 ↑
Detection Accuracy (%)	93.8	89.5	86.2	83.7	12.1 ↑
False Positive Rate (%)	4.1	5.2	5.8	6.5	9.0↓
CPU Utilization (%)	68.0	72.4	74.5	65.0	6.0 ↓
Processing Iterations (avg.)	82	95	103	100	18.0 ↓

Note. ↓ = lower is better; ↑ = higher is better

The QS-CF Quick Sort implementation achieved an average sorting time of 740 ms, outperforming Merge Sort (980 ms) and Insertion Sort (1 125 ms). This corresponds to a 25 % reduction in preprocessing latency. Packet throughput increased from 1 200 packets s^{-1} (Merge Sort baseline) to 1 560 packets $\mathbf{s^{-1}}$, representing a 1.30× speedup.

In detection accuracy, the Quick Sort-ordered IDS achieved 93.8 %, compared with 83.7 % for the unsorted baseline and 89.5 % for Merge Sort. FPR decreased by 9 %, indicating fewer benign packets misclassified as malicious. These results demonstrate that preprocessing order substantially impacts IDS effectiveness, corroborating Lima et al. (2023) and extending their findings to sorting-based optimization.

Comparative Analysis

Compared with enclave-optimized sorting from Gu et al. (2025), QS-CF's 1.30× speedup falls within the lower bound of their reported 1.32×–28.8× gains. The difference arises from dataset scale and memory context: QS-CF operates on streaming IDS traffic ($\leq 10^6$ records) rather than multi-gigabyte enclave data. However, QS-CF's architecture could adopt Flexway's oblivious-access scheduling to enhance resilience to timing and memory-access attacks.

Likewise, in relation to the MPC sorting literature (Chen et al., 2023; Rao et al., 2021), which incurs 3–8× latency overhead for privacy, QS-CF achieves practical efficiency suited for real-time environments while leaving room for privacy-enhanced variants in multi-tenant deployments.

Likewise, in relation to the MPC sorting literature, which incurs 3–8× latency overhead for privacy, QS-CF achieves practical efficiency suited for real-time environments while leaving room for privacy-enhanced variants in multi-tenant deployments.

Overview of Results

The experimental evaluation of the Quick Sort–Cybersecurity Framework (QS-CF) produced quantitative evidence supporting the hypothesis that algorithmic ordering significantly enhances cybersecurity system performance. The tests compared the proposed Quick Sort-based preprocessing with Merge Sort, Insertion Sort, and an unsorted baseline, focusing on processing latency, throughput, and detection accuracy.

QS-CF consistently outperformed its peers, with results showing a 25% reduction in sorting latency and a 30% increase in data throughput compared to Merge Sort. This improvement directly impacted the IDS component, which achieved a detection accuracy of 93.8%, a significant margin above the unsorted baseline (83.7%) and Merge Sort (89.5%). False Positive Rate (FPR) and False Negative Rate (FNR) were also reduced by 9% and 6%, respectively. These improvements demonstrate that efficient data ordering enhances both computational performance and analytic accuracy in cybersecurity systems.

Computational Efficiency and Comparative Performance Analysis

The hybrid Quick Sort implemented in the QS-CF framework demonstrated strong computational efficiency, achieving an average sorting time of 740 ms for 10^5 records, outperforming Merge Sort (980 ms) and Insertion Sort (1,125 ms). This performance gain is attributed to adaptive partitioning and optimized pivot sampling, which reduced recursion depth and redundant comparisons. With CPU utilization remaining below 68%, the framework maintained efficient resource consumption without excessive processing overhead.

These results align with findings by Gu et al. (2023), who reported significant throughput improvements (1.32×–28.8×) using Flexway O-Sort in enclave systems. Although QS-CF achieved a slightly lower improvement margin (1.30×), its effectiveness is notable because it operates without specialized hardware support.

Compared to enclave-based and privacy-focused sorting models (Gu et al., 2025) and secure multi-party mechanisms with higher latency overheads (Chen et al., 2023), QS-CF prioritizes throughput, detection accuracy, and practical

integration within existing IDS pipelines. While it does not yet incorporate full privacy-preserving features, it provides a scalable and extensible foundation for future enhancements.
Furthermore, addressing bottlenecks identified in CTI frameworks (Santos et al., 2025), QS-CF improves log correlation speed by 22% through efficient ordering and redundancy reduction, reinforcing its applicability in Security Operations Centers (SOC) and cybersecurity analytics environments.

CONCLUSION

This study introduced the Quick Sort–Cybersecurity Framework (QS-CF), a computational model that integrates an optimized Quick Sort algorithm into the preprocessing pipeline of cybersecurity systems such as Intrusion Detection Systems (IDS) and Cyber-Threat Intelligence (CTI) platforms. The framework was designed to examine how algorithmic ordering affects processing efficiency and detection accuracy.
Experimental results demonstrated that QS-CF significantly improves system performance compared to Merge Sort, Insertion Sort, and unsorted baselines. Specifically, the framework achieved a 25% reduction in sorting time, a 1.30× increase in throughput, and a 12% improvement in detection accuracy, alongside reductions in false positive and false negative rates. These findings confirm that algorithmic optimization at the data-ordering level can provide a low-cost yet high-impact enhancement to cybersecurity analytics.
Unlike prior studies that emphasize hyperparameter tuning or hardware-level acceleration, QS-CF offers a purely software-based optimization strategy capable of delivering comparable or superior performance gains with minimal resource overhead. Its modular architecture ensures scalability and adaptability, enabling seamless integration with existing IDS and SIEM systems for real-time traffic prioritization, faster alert generation, and reduced analyst fatigue. Additionally, the deterministic sorting structure reduces temporal variability, indirectly strengthening resistance to timing-based side-channel attacks.
Nevertheless, the study has limitations. The current implementation operates in a single-node simulation environment, and although optimized, the Quick Sort algorithm does not yet incorporate privacy-preserving or multi-party computation mechanisms. These constraints may affect performance and applicability in distributed or encrypted environments.

Future Work

Future research will focus on enhancing QS-CF's scalability, privacy, and performance. Key directions include integrating privacy-preserving techniques such as secure multiparty computation and differential privacy, implementing GPU-based parallelization for faster processing of high-volume traffic, and extending deployment to distributed and cloud environments. Additional improvements involve incorporating advanced machine learning models, developing adaptive pivot-selection strategies, and validating the framework in real-world SOC and enterprise settings. These advancements will strengthen QS-CF as a scalable, privacy-aware, and high-performance cybersecurity solution.

REFERENCES

Ala'Anzy, M. A., Mazhit, Z., Ala'Anzy, A. F., Algarni, A., Akhmedov, R., & Bauyrzhan, A. (2024). Comparative Analysis of Sorting Algorithms: A Review. *2024 11th International Conference on Soft Computing & Machine Intelligence (ISCMI)*, 88–100. https://doi.org/10.1109/iscmi63661.2024.10851593

Ayodele, O. S., & Oluwade, B. (2019). *A Comparative Analysis of Quick, Merge and Insertion Sort Algorithms using Three Programming Languages I: Execution Time Analysis*. Google.com; African Journal of Management Information System. https://afrjmis.wordpress.com/2019editions/

Chethan , H. N. (2017). *Network Intrusion dataset (CIC-IDS-2017)*. Kaggle.com. https://www.kaggle.com/datasets/chethuhn/network-intrusion-dataset

Deshmukh, P., & Carter, B. (2023). Secure Multi-Party Computation Protocols for Privacy-Preserving Data Analysis. *International Journal of Recent Advances in Engineering and Technology*, *12*(2), 1–6. https://journals.mriindia.com/index.php/ijraet/article/view/120

Diana, L., Dini, P., & Paolini, D. (2025). Overview on Intrusion Detection Systems for Computers Networking Security. *Computers*, *14*(3), 87. https://doi.org/10.3390/computers14030087

Gu, T., Wang, Y., Cloud, A., Tinoco, A., Shi, E., Chen, B., & Yi, K. (2025). *Flexway O-Sort: Enclave-Friendly and Optimal Oblivious Sorting Flexway O-Sort: Enclave-Friendly and Optimal Oblivious Sorting*. https://www.usenix.org/system/files/usenixsecurity25-gu-tianyao.pdf

Gu, T., Wang, Y., Tinoco, A., Chen, B., Yi, K., & Shi, E. (2023). *Flexway O-Sort: Enclave-Friendly and Optimal Oblivious Sorting*. Cryptology EPrint Archive. https://eprint.iacr.org/2023/1258

Haseeb-ur-rehman, R. M. A., Aman, A. H. M., Hasan, M. K., Ariffin, K. A. Z., Namoun, A., Tufail, A., & Kim, K.-H. (2023). High-Speed Network DDoS Attack Detection: A Survey. *Sensors*, *23*(15), 6850. https://doi.org/10.3390/s23156850

Jia, W., Sun, M., Lian, J., & Hou, S. (2022). Feature Dimensionality reduction: a Review. *Complex & Intelligent Systems*, *8*, 2663–2693. https://doi.org/10.1007/s40747-021-00637-x

Jose, J., & Jose, D. V. (2019). Impact of Distributed Denial of Service attack in Internet of Things Applications-An Overview. *ResearchGate*, *28*(17), 201–205. https://www.researchgate.net/publication/378315828_Impact_of_Distributed_Denial_of_Service_attack_in_Internet_of_Things_Applications-An_Overview

Khalil, A., & Abu-Naser, S. S. (2025, June 6). *AI-Driven Sorting Algorithms: Innovations and Applications in Big Data*. https://doi.org/10.13140/RG.2.2.19176.74245

Lima, M. G., Carvalho, A., Gabriel, Á. J., Escouper, C., & Goldschmidt, R. R. (2024). *Impacts of Data Preprocessing and Hyperparameter Optimization on the Performance of Machine Learning Models Applied to Intrusion Detection Systems*. ArXiv.org. https://arxiv.org/abs/2407.11105

Maithri Bairy, Pai, P., Kini, N. G., & K. Jyothi Upadhya. (2025). Parallelized Hybrid Sorting Using Quick and Insertion Sort for Big Data. *Lecture Notes in Electrical Engineering*, 503–513. https://doi.org/10.1007/978-981-96-9203-3_41

Merlano, C. (2024). Enhancing Cyber Security through Artificial Intelligence and Machine Learning: A Literature Review. *Journal of Cyber Security*, *6*(1), 89–116. https://doi.org/10.32604/jcs.2024.056164

Mohamed, N. (2025). Artificial intelligence and machine learning in cybersecurity: a deep dive into state-of-the-art techniques and future paradigms. *Knowledge and Information Systems*, *67*, 6969–7055. https://doi.org/10.1007/s10115-025-02429-y

Mohammad Kamrul Hasan, Rabiu Aliyu Abdulkadir, Islam, S., Thippa Reddy Gadekallu, & Nurhizam Safie. (2024). A review on machine learning techniques for secured cyber-physical systems in smart grid networks. *Energy Reports*, *11*, 1268–1290. https://doi.org/10.1016/j.egyr.2023.12.040

Munappy, A. R., Bosch, J., Olsson, H. H., Arpteg, A., & Brinne, B. (2022). Data management for production quality deep learning models: Challenges and solutions. *Journal of Systems and Software*, *191*, 111359. https://doi.org/10.1016/j.jss.2022.111359

Nagi. (2025). *NSL-KDD Dataset (filtered version of KDD)*. Kaggle.com. https://www.kaggle.com/datasets/primus11/nsl-kdd-dataset-filtered-version-of-kdd

Rao, C. K., Singh, K., & Kumar, A. (2021). Oblivious stable sorting protocol and oblivious binary search protocol for secure multi-party computation. *Journal of High Speed Networks*, *27*(1), 67–82. https://doi.org/10.3233/jhs-210652

Santos, P., Abreu, R., Reis, M. J. C. S., Serôdio, C., & Branco, F. (2025). A Systematic Review of Cyber Threat Intelligence: The Effectiveness of Technologies, Strategies, and Collaborations in Combating Modern Threats. *Sensors*, *25*(14), 4272. https://doi.org/10.3390/s25144272

wenbosun703. (2025). *Flexway O-Sort Enclave-Friendly and Optimal Oblivious Sorting*. Scribd. https://www.scribd.com/document/944603571/Flexway-O-Sort-Enclave-Friendly-and-Optimal-Oblivious-Sorting