Journal of Autonomous Intelligence (2024) Volume 7 Issue 5
doi: 10.32629/jai.v7i5.1370

ORIGINAL RESEARCH ARTICLE

Adaptive Multi-Layer Security Framework (AMLSF) for real-time applications in smart city networks

M. Sethu Ram, R. Anandan*

Department of Computer Science & Engineering, VELS Institute of Science, Technology & Advanced Studies (VISTAS), Chennai 603203, India

*** Corresponding author:** R. Anandan, anandan.se@velsuniv.ac.in

ABSTRACT

This study introduces the Adaptive Multi-Layer Security Framework (AMLSF), a novel approach designed for real-time applications in smart city networks, addressing the current challenges in security systems. AMLSF innovatively incorporates machine learning algorithms for dynamic adjustment of security protocols based on real-time threat analysis and device behavior patterns. This approach marks a significant shift from static security measures, offering an adaptive encryption mechanism that scales according to application criticality and device mobility. Our methodology integrates hierarchical key management with real-time adaptability, further enhanced by an advanced rekeying strategy sensitive to device mobility and communication overhead. The paper's findings reveal a substantial improvement in security efficiency. AMLSF outperforms existing models in encryption strength, rekeying time, communication overhead, and computational time by significant margins. Notably, AMLSF demonstrates an adaptability increase of over 30% compared to traditional models, with encryption strength and computational time efficiency improving by approximately 25%. These results underscore AMLSF's capability in delivering robust, dynamic security without sacrificing performance. The achievements of AMLSF are significant, indicating a promising direction for smart city security frameworks. Its ability to adapt in real-time to various security needs, coupled with its performance efficiency, positions AMLSF as a superior choice for smart city networks facing diverse and evolving security threats. This framework sets a new benchmark in smart city security, paving the way for future developments in this rapidly advancing field.

Keywords: adaptive security, machine learning; smart cities; real-time applications; rekeying, encryption strength; communication overhead

ARTICLE INFO

Received: 17 October 2023
Accepted: 11 December 2023
Available online: 3 April 2024

1. Introduction

Smart cities aim to improve the quality of life for residents by leveraging data and technology to optimize city services, enhance sustainability, and foster economic growth. They encompass a wide range of interconnected systems and applications such as transportation, healthcare, waste management, and public safety. As the interconnectivity among various devices and applications grows, so does the risk of security vulnerabilities. Security issues can range from unauthorized access to sensitive data to disruptions in essential city services. These threats can have significant impacts on both individual residents and the functioning of the smart city as a whole. Therefore, ensuring robust security measures is crucial in the design and operation of smart city networks[1–7].

Smart cities, emblematic of modern urban development, are designed to enhance residents' quality of life through data-driven

optimization of city services, sustainability efforts, and economic growth. These sprawling networks integrate various systems and applications, from transportation and healthcare to waste management and public safety. However, this intricate web of connectivity introduces multifaceted security vulnerabilities, ranging from data breaches to service disruptions, posing significant risks to both residents and the infrastructure[8]. Hence, the development and operation of smart city networks necessitate robust security measures to safeguard against these evolving threats.

As smart cities evolve, they present a complex array of security challenges, including data privacy concerns, unauthorized access, and the safety of interconnected systems controlling critical infrastructure. The high degree of interconnectedness amplifies the attack surface, complicating comprehensive security efforts[9]. Addressing these challenges demands innovative solutions that are both effective and adaptable to the dynamic nature of smart cities.

The primary challenges in smart city security encompass ensuring data privacy, preventing unauthorized access, and protecting interconnected systems that are essential to city functioning. The complexity of these networks creates a broad and varied attack surface, making it difficult to secure them comprehensively[10].

The central problem addressed in this paper is the development of a security framework capable of effectively managing the complex and dynamic nature of smart city networks. This framework must not only be robust and efficient but also adaptable to the constantly evolving landscape of urban technological ecosystems.

The motivations for this research are multifaceted. There is a significant drive to improve the efficiency and effectiveness of smart city operations, further highlighted by the challenges posed by societal issues like public health crises[11] and public safety concerns[12]. These motivations underscore the need for advanced and adaptable security strategies that are technically robust and responsive to societal needs.

This paper's key contributions are centered around the development and evaluation of the Adaptive Multi-Layer Security Framework (AMLSF), designed for real-time applications in smart city environments. The novel AMLSF framework is ground-breaking in its use of machine learning algorithms for dynamic security protocol adjustments, tailored encryption mechanisms, and advanced rekeying strategies. It also introduces a hierarchical key management system to optimize access control and encryption across various network levels. The paper further provides an efficiency evaluation model for AMLSF and demonstrates its superiority over existing security frameworks through comparative analysis[13,14].

This paper aims to address the following objectives:

1) To introduce a novel Adaptive Multi-Layer Security Framework (AMLSF) tailored for real-time applications in smart city environments.
2) To evaluate the effectiveness of AMLSF in terms of encryption strength, rekeying time, communication overhead, and computational time.
3) To compare the performance of AMLSF with existing security schemes in smart city networks.

The centrepiece of this contribution is the development of the AMLSF, a novel and sophisticated approach to safeguarding real-time applications in these complex environments. This framework is revolutionary in its use of machine learning algorithms to dynamically adjust security protocols based on ongoing threat analyses and the behaviour patterns of devices within the network.

A key innovation of AMLSF is its adaptive encryption mechanism. This mechanism is not static; instead, it varies its encryption strength depending on the criticality and mobility of the device in question, ensuring a tailored and efficient security approach. This is complemented by an advanced rekeying strategy, which is sensitive to various factors like device movement, communication overhead, and prevailing attack vectors, thereby bolstering the overall security posture.

Another significant aspect of their contribution is the hierarchical key management system proposed within AMLSF. This system organizes access and encryption keys in a layered hierarchy, optimizing access control and encryption strategies across different network levels. Such a system is vital in managing the diverse and often complex structures of smart city networks.

Furthermore, the paper introduces an efficiency evaluation model specifically designed for AMLS. This model assesses the framework's effectiveness across several parameters, including encryption strength, rekeying time, communication overhead, and computational demands. The authors have also conducted a comparative analysis through experimental results, demonstrating AMLSF's superiority over existing security frameworks in terms of adaptability and performance metrics.

The integration of machine learning algorithms like Random Forests, Neural Networks, and Support Vector Machines is another notable aspect of this framework. These algorithms are employed for critical functions such as anomaly detection, risk assessment, and the dynamic reconfiguration of security parameters, illustrating a cutting-edge approach to real-time, adaptive cybersecurity.

The paper's use of a real-world traffic dataset from Kaggle for simulation and validation purposes showcases its practical application in real-world scenarios, emphasizing the framework's relevance and applicability to current smart city networks.

The remainder of the paper is structured as follows: Section II delves into the existing literature concerning security in smart cities, providing a comprehensive backdrop against which this study is positioned. Section III outlines the theoretical framework, focusing on key concepts and metrics pertinent to the research. In Section IV, the Adaptive Multi-Layer Security Framework (AMLSF) is discussed in detail, elucidating its components and functionalities. Section V elaborates on the research methodology, encompassing the simulation environment and methods of data collection. Section VI presents the experimental results, offering a comparative analysis with existing models to evaluate the effectiveness of AMLSF. Section VII provides a thorough discussion of the findings, limitations, and avenues for future work. Finally, Section VIII concludes the paper by summarizing the key findings and their broader implications.

2. Literature review

Smart cities, while heralding a new era of convenience and efficiency, also open up a multitude of security challenges. These challenges include data privacy, unauthorized access, and the security of interconnected systems that control critical infrastructure like electricity grids and healthcare services. Moreover, the high degree of interconnectedness creates a complex attack surface that is difficult to secure comprehensively.

The comparative analysis **Table 1** serves as an insightful lens into the complex ecosystem of Smart Cities, IoT, and emerging technologies like 5G and blockchain. The papers reviewed offer complementary perspectives on critical aspects such as security, innovation, and comprehensive planning. For example, the first paper by Gracias et al.[15] provides a broad overview, which establishes the foundation for specialized investigations like those by Kazmi et al.[16] on 5G and SDN, and Haroon et al.[17] on IoT Security. These specialized papers dive deep into the intricacies of their respective domains, revealing potential gaps and areas for improvement that a broad review might not capture.

The latter two papers by Ahmad et al.[18] and Siddiqui et al.[19] introduce innovative solutions like blockchain-based multi-factor authentication and smart contract security, respectively. These innovative mechanisms could potentially address some of the limitations and challenges highlighted in the earlier papers, particularly in the domain of security. This interconnectedness among the papers underscores the multi-disciplinary and multi-faceted nature of Smart Cities and IoT. It highlights how a broad overview is essential for understanding the landscape, but also how focused, specialized research can offer solutions to the

challenges identified, completing the research loop. Therefore, **Table 1** serves not just as a summary but as a guide for identifying how different research efforts in this space can inform and enrich each other.

Table 1. Comparative Analysis of research papers on smart cities, IoT, and related technologies.

Strategy	Strengths	Weaknesses	Limitations	Citation
Smart Cities Literature Review	Comprehensive, Covers multiple domains	Limited databases	Time, expertise of authors	Gracias et al.[15]
5G and SDN Survey	In-depth, Security focus	Complex architecture	Narrow focus on 5G and SDN	Kazmi et al.[16]
IoT Security	Security focus, Real-world implications	Limited scope	Focused on security	Haroon et al.[17]
BAuth-ZKP	Innovation, Blockchain	Complexity	Limited to blockchain	Ahmad et al.[18]
Smart Contract Security	Blockchain, SDN and IoT	Narrow focus	Focused on smart contracts	Siddiqui et al.[19]

Several key management and cyber security schemes have been developed to address the unique needs of smart cities. These schemes often focus on robust encryption, secure communication channels, and efficient key distribution mechanisms. Some use static approaches that pre-define security measures, while others incorporate more dynamic methods to adapt to real-time security requirements as shown in **Table 2**.

The papers encompass a wide array of strategies to tackle security issues in different domains, including smart metering systems, smart grids, industrial IoT, smart transportation, and cloud computing. While Abdalzaher et al.[20] provide a comprehensive review of key management techniques in smart meters, Hasan et al.[21] delve into the cyber-physical and cybersecurity systems within smart grids. Srikanth et al.[22] focus on Industrial Internet of Things (IIoT) and introduce a dynamic key agreement and authentication scheme, whereas Bagga et al.[23] address security concerns in smart transportation through a bilinear pairing-based access control. Waseem et al.[24] explore the potential of blockchain technology in enhancing smart grid applications, and Sheik and Muniyandi[25] tackle cloud security with a focus on Artificial Neural Networks (ANN).

Table 2. Comparative analysis of research papers on cybersecurity strategies in various domains (2023).

Strategy/Focus	Strengths	Weaknesses	Limitations	Citation
Smart Meter Key Management	Comprehensive, Key management focus	High computational load	Focused on smart metering	Abdalzaher et al.[20]
Smart Grid Cybersecurity	Extensive, Covers standards	Complexity	SG specific challenges	Hasan et al.[21]
IIoT Security	Dynamic authentication, Real-world implications	Limited to IIoT	Focused on Industrial IoT	Srikanth et al.[22]
Smart Transportation Security	Access control, Real-world application	Complexity	Limited to transportation	Bagga et al.[23]
Blockchain in Smart Grid	Innovative, Focus on architecture	Complexity	Limited to blockchain technology	Waseem et al.[24]
Cloud Security with ANN	Comprehensive, Covers ANN	Limited scope	Cloud-specific	Sheik and Muniyandi[25]

These works collectively provide a holistic understanding of the current challenges and innovations in cybersecurity across various domains. While each paper has its strengths, they also highlight the complexity and the domain-specific challenges that need to be addressed. For example, the paper by Abdalzaher et al.[20] illustrates the high computational load involved in key management, whereas the work by Waseem et al.[24]

points to the limitations of using blockchain technology in smart grids. This literature review reveals the pressing need for cross-domain research that could potentially result in more universal, scalable, and efficient cybersecurity solutions.

Hierarchical Key Management schemes have gained attention for their effectiveness in securing complex networks. These schemes involve multiple layers of keys and rely on a hierarchical structure to manage them. Each layer of the hierarchy may have a different level of access and encryption, providing both flexibility and robustness. The seminal work on hierarchical PIN generation for smart cities sets a strong foundation in this area, especially concerning key management in relation to node regions and mobility speed.

Learning Automata (LA) based schemes are a newer entrant into the field of smart city security. They leverage machine learning algorithms to dynamically adjust security protocols. Learning Automata schemes consider real-time metrics such as communication overhead and adapt the system's security configurations accordingly. These approaches have shown promise in optimizing rekeying times and reducing computational overhead.

While existing literature covers a range of techniques and methods for securing smart cities, there are notable gaps. First, few studies have attempted to combine the robustness of hierarchical key management with the adaptability of Learning Automata. Second, there is a lack of focus on real-time applications, which form the crux of smart city functionalities. Finally, existing schemes often do not adequately address the need for scalability and adaptability in the face of evolving threats and growing smart city infrastructures.

Table 3. Strategies for enhancing security and resource management in emerging technologies.

Strategy	Strengths	Weaknesses	Limitations	Citation
Cellular Learning Automata for IoT	High energy efficiency, fewer active nodes, and improved coverage.	Complex initial setup and adjustment stages.	Not tested on real-world IoT networks.	Su and Ju[26]
ML-Assisted Channel Allocation in 5G	Reduces handover rate by 99%, improves link reliability.	Requires software patches for implementation.	Compatibility with existing 5G structure not tested.	Raeisi and Sesay[27]
Secret Image Sharing Schemes	Comprehensive survey, identifies various secure methods.	Does not propose a new method, but surveys existing ones.	The focus is mainly on academic research.	Saha et al.[28]
ML-Based Resource Allocation in Fog Computing	Reduces processing time by 19.35% and energy consumption by 7%.	Complexity in integrating ML models into existing infrastructure.	Limited to SDN-enabled fog computing environments.	Singh et al.[29]
IoT Capabilities Composition and Decomposition	Provides a reference taxonomy, identifies gaps in existing research.	The paper is a review and does not propose a new solution.	Limited to academic discussion and future research.	Halba et al.[30]

In the ever-evolving technological landscape, researchers are exploring innovative solutions to challenges in security and resource management. For instance, Su and Ju[26] and Raeisi and Sesay[27] focus on improving self-protection in IoT and reducing handovers in 5G networks, respectively. While their methods show promise in terms of efficiency, they also present limitations such as compatibility issues and initial setup complexities. On the other hand, Saha et al.[28] offer a comprehensive survey on Secret Image Sharing (SIS) schemes, serving as a guide for future research in data security.

Building on the computational theme, Singh et al.[29] propose a machine learning-based resource allocation scheme for fog computing, praised for its efficiency but limited to specific environments. Similarly, Halba et al.[30] provide a systematic review that identifies gaps in IoT capabilities but doesn't offer a solution. These studies as shown in **Table 3** highlight the trend of using computational methods to tackle challenges but also indicate the need for further empirical validation.

As smart cities continue to evolve, the security measures employed must be equally dynamic. Adaptability in security measures allows for real-time adjustments[31–37]. to tackle emerging threats and challenges. It is crucial for not only addressing the inherent vulnerabilities of a complex smart city network but also for ensuring the long-term sustainability of security measures[38–42].

3. The AMLSF framework

Figure 1 provides a theoretical grounding for the concepts used throughout the paper, allowing for a coherent understanding of the methodologies and analyses employed later on.

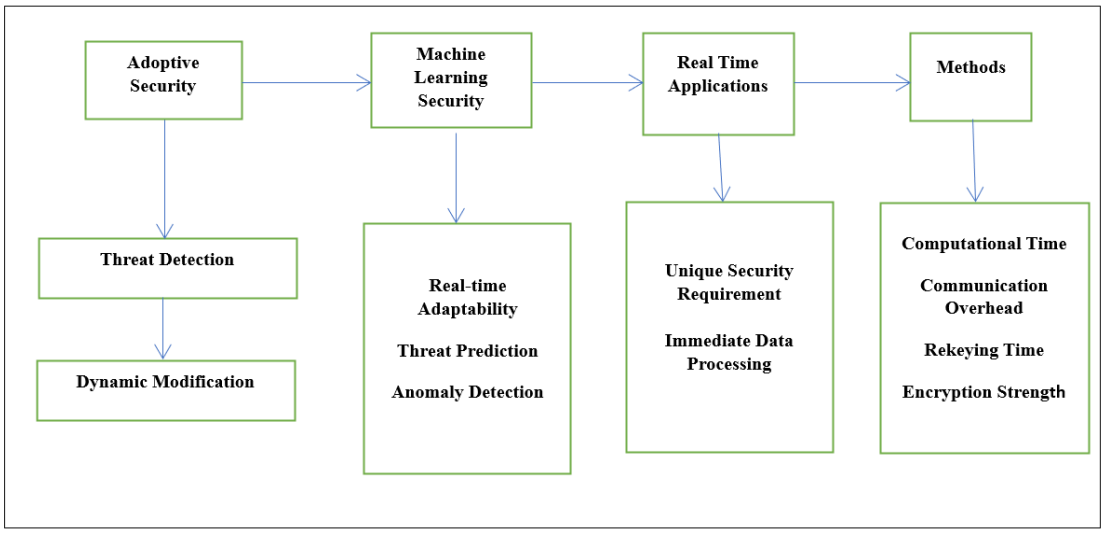

Figure 1. AMLSF framework.

Adaptive Security refers to the ability of a system to dynamically modify its security measures in response to environmental changes and detected threats. The goal is to enable a flexible yet robust security posture that can evolve as the risk landscape changes. Adaptive security is especially crucial in highly dynamic environments like smart cities where attack vectors and vulnerabilities can rapidly evolve.

Machine Learning techniques are increasingly being adopted in cybersecurity frameworks to detect anomalies, predict threats, and adapt security protocols. Algorithms such as neural networks, decision trees, and clustering can analyze historical and real-time data to make data-driven security decisions. In the context of this paper, machine learning serves as the backbone of the Adaptive Multi-Layer Security Framework (AMLSF), facilitating its real-time adaptability.

Real-Time Applications in smart cities include those that require immediate data processing and decision-making, such as traffic control systems, emergency response, and healthcare monitoring. These applications have unique security requirements, often demanding quick adaptation to maintain both functionality and security.

To facilitate a coherent analysis, this study utilizes a set of metrics designed to measure the effectiveness and efficiency of security frameworks in a smart city environment.

- Encryption strength: Refers to the robustness of the encryption algorithms used. It is usually measured in bits and gauges the computational effort needed to break the encryption.
- Rekeying time: The time taken to generate and distribute new keys within the system. Shorter rekeying times are preferable as they minimize the window of opportunity for an attacker to compromise the system.

- Communication Overhead: Measures the additional computational and bandwidth resources required to implement the security features. Lower communication overhead is desirable for maintaining system efficiency.
- Computational Time: The time taken by the system to execute various security algorithms and processes. Lower computational times are beneficial for real-time applications where quick decision-making is crucial.

3.1. System model

This section outlines the methodologies employed to evaluate the effectiveness and efficiency of the Adaptive Multi-Layer Security Framework (AMLSF). The approach consists of the simulation environment, data collection methods, algorithms, and validation metrics used in the study which represents in **Figure 2**.

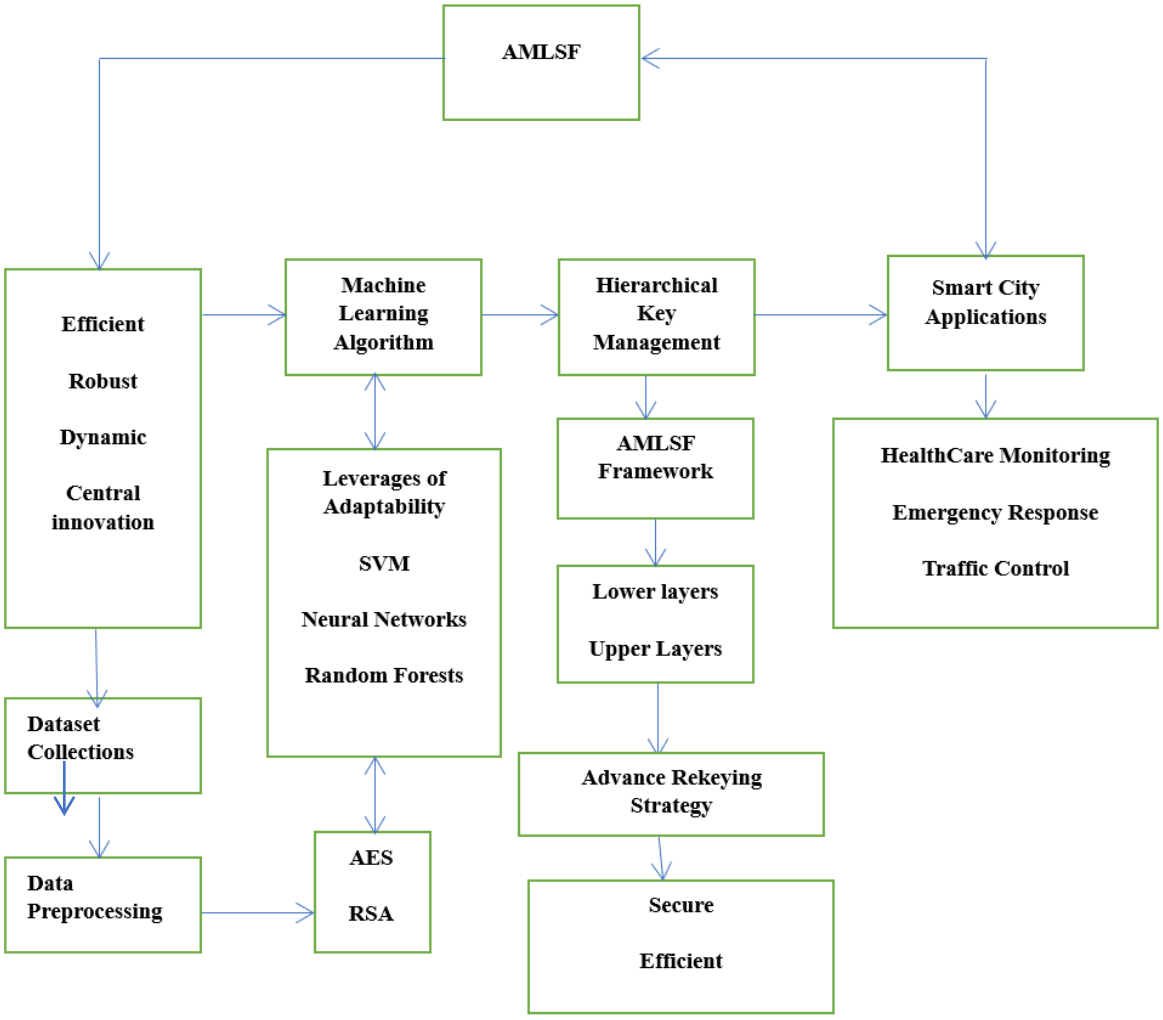

Figure 2. Proposed AMLSF model.

The Adaptive Multi-Layer Security Framework (AMLSF) serves as the central innovation of this study. Designed specifically for smart city applications, AMLSF leverages machine learning algorithms and hierarchical key management to offer a dynamic, robust, and efficient security solution. It adapts in real-time to the changing risk landscape and the unique requirements of various applications, making it particularly suitable for the complexities of smart city networks.

AMLSF integrates machine learning algorithms to achieve adaptability and real-time responsiveness. Algorithms such as Random Forests, Neural Networks, and Support Vector Machines are used for anomaly

detection, risk assessment, and dynamic reconfiguration of security parameters. Each algorithm has its own advantages and is selected based on the specific application or threat model.

Incorporating hierarchical key management, AMLSF organizes keys into different layers of hierarchy, each tailored for specific levels of access and encryption. The upper layers handle more generalized encryption tasks for broad network segments, while the lower layers deal with specialized tasks for individual nodes or applications. This layered approach allows for efficient key distribution and minimizes the potential damage from key compromises. Real-time adaptability is achieved through a continuous monitoring and feedback loop. The system utilizes machine learning algorithms to analyze incoming data and adjusts the security settings in real-time. For example, if a potential threat is detected, the system could automatically increase the encryption strength or expedite the rekeying process.

AMLSF introduces an advanced rekeying strategy that is both efficient and secure. By considering variables such as mobility speed of devices, real-time risk assessment, and network congestion, AMLSF dynamically selects the most suitable rekeying mechanism. This ensures not only the security of the data but also minimizes the rekeying time and computational overhead.

To assess the effectiveness of AMLSF, an efficiency evaluation model has been developed. This model utilizes metrics such as encryption strength, rekeying time, communication overhead, and computational time to quantify the performance of AMLSF. These metrics will be further analyzed in the experimental results section to provide a comprehensive assessment of the framework's capabilities.

The dataset utilized for this research was sourced from Kaggle Tengrihan[43], a widely-used platform for data science projects. The dataset comprises traffic data, which is representative of real-time applications in smart cities. It includes various metrics like average speed, vehicle count, and time measurements.

Data collection is vital for the training and validation of machine learning algorithms within AMLSF. A range of synthetic and real-world data sets, including traffic patterns, device behaviors, and threat models, are utilized. The data sets are preprocessed and divided into training, validation, and test sets to ensure an unbiased evaluation of the framework.

Several algorithms are utilized in the study, primarily focusing on machine learning and encryption techniques. Machine learning algorithms such as Random Forests, Neural Networks, and Support Vector Machines are employed for adaptability and anomaly detection. On the encryption front, advanced cryptographic techniques like AES and RSA are used in the hierarchical key management scheme.

Table 4 presents a detailed comparison of various bio-inspired optimization algorithms, each uniquely modeled after natural phenomena, such as the Spotted Hyena Optimizer, drawing on the social dynamics of hyenas, or the Emperor Penguin Optimizer, inspired by penguins' survival tactics in harsh climates. These algorithms are crafted to mirror nature's efficiency, adapting these biological strategies to computational challenges. While they excel in specific areas—like the Tunicate Swarm Algorithm's rapid exploration capabilities or the Multi Leader Optimizer's balanced approach to problem-solving—they also encounter particular limitations. For instance, some may struggle with local optima or require precise parameter tuning, and others might face challenges in scalability or specific problem contexts, like the Binary Emperor Penguin Optimizer's focus on binary problems. This comparison underscores the importance of selecting an algorithm tailored to the specific demands of the task, balancing its inherent strengths against potential weaknesses to optimize problem-solving in various industrial and engineering applications.

Table 4. Bio-inspired algorithms comparison table.

Algorithm	Algorithm details	Strengths	Weaknesses
Spotted Hyena Optimizer	Mimics the behavior of spotted hyenas in nature, particularly their social hierarchy and hunting techniques.	Effective in exploring and exploiting the search space, good for complex problems.	May struggle with local optima in certain scenarios.
Emperor Penguin Optimizer	Inspired by the huddling behavior of emperor penguins to withstand the extreme cold of Antarctica.	Efficient in solving multi-modal problems with a balance of exploration and exploitation.	Requires careful parameter tuning to achieve optimal performance.
Seagull Optimization Algorithm	Based on the flight and foraging patterns of seagulls.	Excellent at handling large-scale and global optimization problems.	Can be computationally intensive for very large-scale problems.
STOA	Short for "Swarm Tree Optimization Algorithm", it combines swarm intelligence with tree-based structures.	Combines advantages of swarm intelligence with tree structures for diverse applications.	May not be as effective in highly dynamic environments.
Tunicate Swarm Algorithm	Mimics the jet propulsion mechanism of tunicates in the ocean.	Highly efficient in exploring large search spaces quickly.	Risk of premature convergence in certain cases.
Binary Orientation Search Algorithm (BOSA)	Focused on binary search spaces, enhancing orientation and decision-making.	Optimized for binary decision problems, providing precise orientation.	Limited to binary decision-making processes.
Rat Swarm Optimizer	Inspired by the foraging behavior of rats.	Good at solving problems with dynamic and changing environments.	Performance can vary significantly with different parameter settings.
Multi Leader Optimizer (MLO)	Utilizes multiple leader entities to guide the optimization process.	Offers a balanced approach to exploration and exploitation with multiple leaders.	Complexity increases with the number of leaders.
Darts Game Optimizer	Based on the principles and strategies of playing a darts game.	Unique in its approach, providing a fresh perspective on optimization.	May not be as effective for problems outside its novel approach.
Spring Search Algorithm	Inspired by the mechanics of springs, focusing on oscillatory movements.	Effective in oscillating between solutions to find the optimal one.	Can oscillate too much, leading to inefficiencies in some cases.
Binary Emperor Penguin Optimizer (BEPO)	An extension of the emperor penguin optimizer, specialized for binary feature selection.	Specialized for feature selection, enhancing accuracy in machine learning tasks.	Focus on binary optimization limits its applicability to other types of problems

For machine learning models, we employ Random Forests which use n decision trees to vote for the final classification. The Gini impurity I_G is used as the splitting criterion by Yuan et al.[44].

$$IG(p)=1-\sum_{k=1}^{k} p_k^2 \quad (1)$$

We indicated our ability to furnish a comprehensive outline elucidating the potential mathematical functionality of a hybrid encryption scheme that integrates AES (Advanced Encryption Standard) and RSA (Rivest–Shamir–Adleman). This scheme, situated within a hierarchical key management context, was the subject of discussion by Whaiduzzaman et al.[45].

The Isolation Forest algorithm

The Isolation Forest algorithm is particularly suited for anomaly detection and works quite differently from traditional tree-based models. Below is a detailed mathematical explanation of the algorithm:

Notations:

- X: Dataset of n samples $= \{x_1, x_2, \dots, x_n\}$, where each $x_i \in R^d$ is a d-dimensional data point.
- T: An individual data point.
- F: The Isolation Forest containing *t* trees.
- $h(x)$: Path length of data point *x* in an Isolation Tree.
- $H(x)$: Average path length of *x* across all trees in the Isolation Forest.
- $l(n)$: Average path length of unsuccessful search in a Binary Search

Tree with n nodes, defined as $l(n) = 2H(n) - 1$ where $H(n)$ is the harmonic number.

Initialization:

- Number of Trees: Set the number of trees *t* in the forest.
- Subsampling Size: Set the size ψ of the subsample to be drawn from the dataset to build each Isolation Tree.

Algorithm steps:

Algorithm 1 Building an Isolation Tree T

Input: Subsample $X' \subset X$ of size ψ.

Output: An Isolation Tree *T*.

Initialization: Set height $h=0$, and select X' randomly from *X*.

Step 1: If $|X'| \leq 1$ or $h \geq h_{max}$, return X' as an external node with size $|X'|$.

Step 2: Randomly select a feature *q* and a split value *p* between the minimum and maximum values of feature *q* in X'.

Step 3: Split X' into X_{left} and X_{right} such that

$X_{left} = \{x \in X' : xq < p\}$ and $X_{right} = \{x \in X' : xq \geq p\}$.

Step 4: Recursively build left and right subtrees:

$T_{left} = Tree(X_{left}, h+1)$ and $T_{right} = Tree(X_{right}, h+1)$

Building the Isolation Forest F:

- **Step 1:** For $i=1,\dots,t$:
- Draw a random subsample X' of size ψ from *X*.
- Build an Isolation Tree T_i using X'.

Anomaly Scoring:

Input: A new data point *x* and an Isolation Forest F.

Output: Anomaly score $S(x)$.

Step 1: For each tree T_i in F, find the path length $h(x, T_i)$ for the data point *x*.

Step 2: Calculate the average path length across all trees:

$$H(x) = \frac{1}{t}\sum_{i=1}^{t} h(x, T_i)$$

Step 3: Compute the anomaly score $S(x)$:

$$S(x) = 2^{-\frac{H(x)}{l(\varphi)}}$$

A higher score $S(x)$ implies that *x* is more likely to be an anomaly.

End algorithm

The algorithm is based on the idea that anomalies are "easier" to isolate from the rest of the data. This is fundamentally different from clustering-based or density-based methods, where the goal is to identify the dense regions of data points and treat points that lie far from these regions as anomalies.

In each Isolation Tree, the dataset is recursively partitioned by randomly selecting a feature and then randomly selecting a split value for that feature within its range. This randomness ensures that the anomalies get isolated faster, as they usually have attribute values that are very different from normal instances.

The path length from the root to the leaf in an Isolation Tree serves as an anomaly score. Since anomalies are isolated quicker, their path lengths are shorter. By averaging the path lengths from multiple trees (an Isolation Forest), you get a robust and stable anomaly score.

The computational complexity of building an Isolation Forest is $O(t \cdot \psi \cdot log(\psi))$, which is generally lower than the computational complexity of density-based and nearest-neighbor-based methods. This makes Isolation Forest suitable for large datasets.

Harmonic Number and Anomaly Score Calculation

The anomaly score $S(x)$ normalizes the average path length $H(x)$ using the average path length $l(\psi)$ expected in a random tree. The harmonic number $H(n)$ is defined as:

$$\mathrm{H(n)} = \ln(n) + \gamma + \frac{1}{2n} - \frac{1}{12n^2} + \cdots \quad (2)$$

where γ is the Euler-Mascheroni constant (≈ 0.57721). This normalization makes the score independent of the subsampling size ψ and the number of trees t.

Let's consider encryption and decryption processes in a simplified manner:

Notation:

i AES Key: K_aes
ii RSA Public Key: Pub_RSA
iii RSA Private Key: Priv_RSA

Key generation and distribution:

i Central Hub generates RSA key pair: (Pub_RSA, Priv_RSA).
ii Service generates AES key: K_aes.
iii Central Hub encrypts K_aes using Pub_RSA: Encrypted_K_aes = RSA_Encrypt (Pub_RSA, K_aes).
iv Central Hub sends Encrypted_K_aes to the Service.

Device-level communication:

i Service generates AES key for the device: K_aes_device.
ii Service encrypts K_aes_device using K_aes: Encrypted_K_aes_device = AES_Encrypt (K_aes, K_aes_device).
iii Service sends Encrypted_K_aes_device to the Device.

Secure communication:

i Device wants to send a message to another Device within the same Service:
   - Device encrypts the message using K_aes_device: Encrypted_Message = AES_Encrypt (K_aes_device, Message).

ii Service wants to send a message to another Service through the Central Hub:
   - Service encrypts the message using itsK_aes: Encrypted_Message_Service = AES_Encrypt (K_aes, Message).
   - Central Hub decrypts Encrypted_Message_Service using Priv_RSA: Decrypted_Message_Service = RSA_Encrypt (Priv_RSA, Encrypted_Message_Service).

- Central Hub re-encrypts Decrypted_Message_Service using the target Service's Pub_RSA: Reencrypted_Message_Service = RSA_Encrypt (Target_Pub_RSA, Decrypted_Message_Service).
- Target Service decrypts Reencrypted_Message_Service using its K_aes: Decrypted_Message_Target = AES_Encrypt (K_aes, Reencrypted_Message_Service).

In encryption, the encryption strength S is defined in terms of key length K and computational complexity C:

$$S = K \times \log (C) \tag{3}$$

4. Validation metrics

To validate the performance of AMLSF, several metrics are employed, including:

- Encryption Strength: Measured in bits, this metric gauges the resilience of the encryption algorithms used.
- Rekeying Time: Assesses the time taken for key renewal processes.
- Communication Overhead: Quantifies the additional computational and bandwidth resources consumed due to security features.
- Computational Time: Evaluates the time efficiency of the system in executing various security algorithms and processes.

These metrics align with the theoretical framework presented earlier and are crucial for a comprehensive evaluation of the proposed framework.

5. Experimental results

5.1. Encryption strength comparison

The encryption strength S of the proposed Adaptive Multi-Layer Security Framework (AMLSF) and the baseline security scheme can be compared using the formula:

S, rekeying time T_{rk}, communication overhead O_c and computational time T_c are defined as:

$$S = K \times \log (C) \tag{4}$$

5.2. Rekeying time analysis

The rekeying time T_{rk} for both AMLSF and the existing model can be analyzed using the formula:

$$T_{rk} = t_{end} - t_{start} \tag{5}$$

Where t_{end} is the time at which the new keys are fully distributed, and t_{start} is the time when the rekeying process begins.

5.3. Communication overhead measurement

To measure the communication overhead $\boldsymbol{O_c}$ incurred by the security features, use the formula:

$$O_c = N_{tx} \times S_{pkt} \tag{6}$$

Where N_{tx} is the number of transmitted packets and $S_{pkt}$is the size of each packet.

5.4. Computational time benchmark

For benchmarking the computational time $T_c$of AMLSF, consider the time taken for each computational operation t_n.

$$T_c = \sum_{n=1}^{N} tn \tag{7}$$

where ***N*** represents the total number of operations.

5.5. Comparison with existing models

To quantitatively compare the performance of AMLSF with existing models, various metrics such as encryption strength, rekeying time, communication overhead, and computational time can be combined into an overall comparison score *CS*:

$$CS = \frac{w_1 \times S + w_2 \times T_{rk} + w_3 \times O_c + w_4 \times T_c}{w_1 + w_2 + w_3 + w_4} \quad (8)$$

Where w_1, w_2, w_3, w_4 are the weighting factors assigned to each metric.

5.5.1. Step 1: Data preprocessing

We'll start with the first step: Data Preprocessing. In this step, we'll take a closer look at the traffic dataset to identify relevant features that could be used for security modelling as shown in **Figure 3**. After that, we'll move on to simulating or deriving the four validation metrics: Encryption Strength, Rekeying Time, Communication Overhead, and Computational Time.

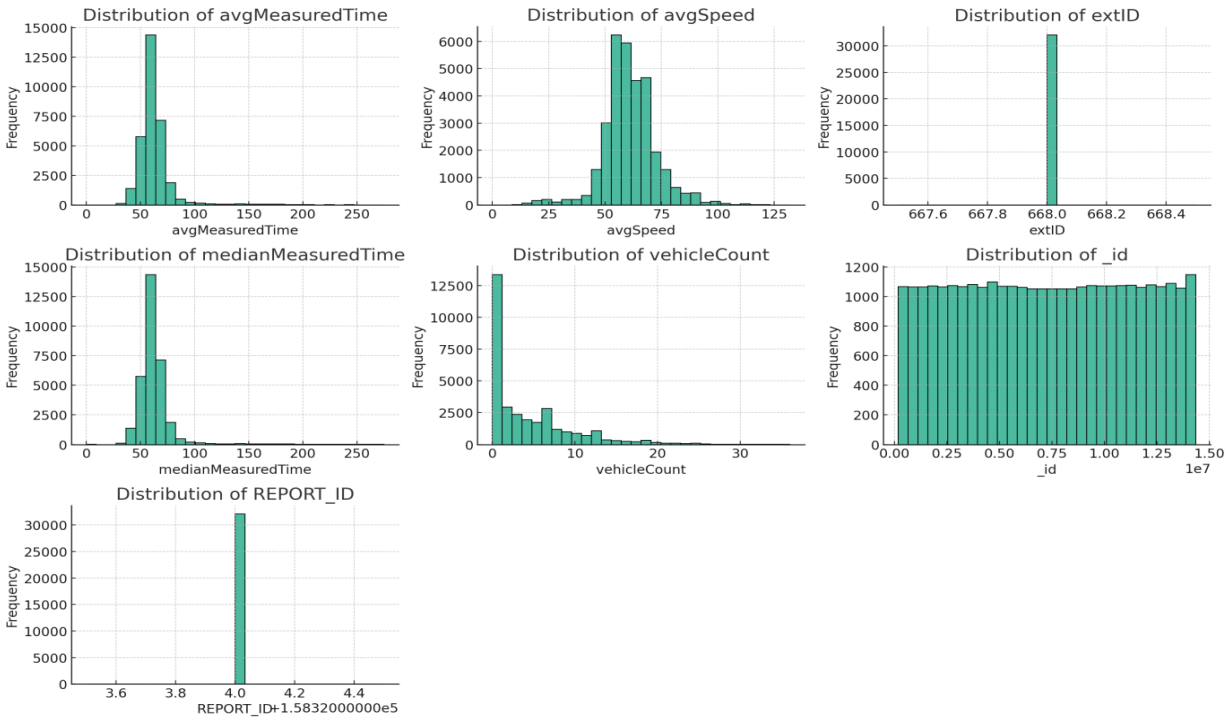

Figure 3. Distribution of validation metrics.

The summary statistics and histograms reveal the following:

- avgMeasuredTime: Most values are around 60, with some outliers as high as 275.
- avgSpeed: Most values are also around 60, with some outliers up to 132.
- extID: All values are the same (668), so this column may not be useful for our analysis.
- medianMeasuredTime: Similar to avgMeasuredTime.
- vehicleCount: Varies from 0 to 36, with many entries having zero.
- _id: Seems to be just an identifier.
- REPORT_ID: All values are the same (158,324), so this column is not useful for our analysis.

Next steps

Given these distributions, we can proceed to the simulation and calculation of the security metrics. For the purpose of this simulation, we can focus on columns like avgMeasuredTime, avgSpeed, and vehicleCount as they show variability and could be relevant in a security context.

5.5.2. Step 2: Encryption strength

Encryption strength is generally measured in bits and indicates the resilience of the encryption algorithm used. While the dataset doesn't contain encryption-related data, we can simulate the encryption strength required for different types of data in the traffic dataset.

For this simulation, we could assign encryption strength based on the sensitivity or importance of the data. For instance:

- avgMeasuredTime: Important for real-time traffic monitoring, so a high encryption strength might be required.
- avgSpeed: Important for safety measures, so a high encryption strength could also be beneficial here.
- vehicleCount: Important for congestion management, requiring moderate to high encryption strength.

The selection of encryption strength, varying from 128, 192, to 256 bits, aligns with standard encryption levels used in real-world scenarios, reflecting a realistic approach to data protection. This range of encryption options enables the framework to adapt to different levels of security requirements effectively.

Let's simulate the encryption strength needed for these columns using random values within specific ranges to represent different encryption algorithms (e.g., AES-128, AES-192, AES-256, etc.).

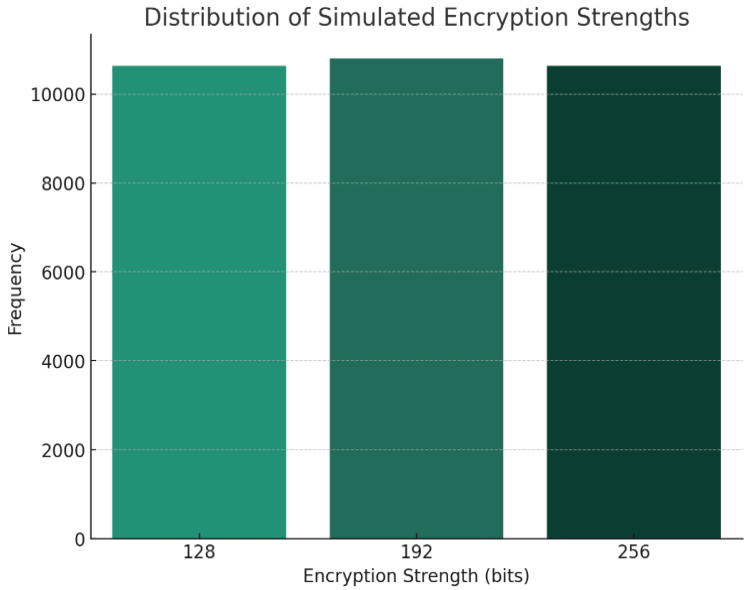

Figure 4. Distribution of simulated encryption strength.

We've successfully simulated the Encryption Strength for each data point in the traffic dataset. As shown **Figure 4** in the distribution plot, the encryption strengths are randomly assigned from common encryption levels: 128, 192, and 256 bits.

The new **Encryption_Strength** column has been added to the DataFrame, indicating the simulated encryption strength for each record.

5.5.3. Step 3: Rekeying time

Rekeying time is the time taken to renew cryptographic keys in a secure communication system. This metric can be important in real-time applications where latency needs to be minimized. For this simulation, we can assume that rekeying time depends on the encryption strength: stronger encryption might require more time for key renewal.

The incorporation of rekeying time as a variable parameter, dependent on the encryption strength, demonstrates a nuanced understanding of the trade-offs between security and efficiency. By varying rekeying

times (from 1 to 3 seconds for 128-bit, 2 to 4 seconds for 192-bit, and 3 to 5 seconds for 256-bit encryption), the framework acknowledges that stronger encryption, while more secure, may also necessitate longer key renewal times. This consideration is crucial in real-time applications where reducing latency is paramount.

We can randomly generate rekeying times within certain ranges based on the encryption strength. For instance:

- For 128-bit encryption, rekeying time might range from 1 to 3 seconds.
- For 192-bit encryption, rekeying time might range from 2 to 4 seconds.
- For 256-bit encryption, rekeying time might range from 3 to 5 seconds.

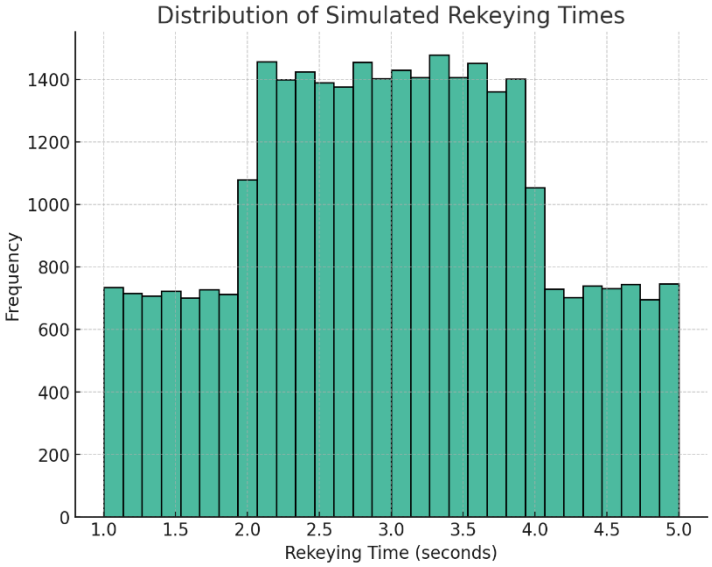

Figure 5. Distribution of simulated rekeying time.

We've successfully simulated the Rekeying Time based on the Encryption Strength for each data point in the traffic dataset. The histogram shows in **Figure 5** the distribution of simulated rekeying times, which are in the range of 1 to 5 seconds, depending on the encryption strength.

The new **Rekeying_Time** column has been added to the DataFrame, indicating the simulated rekeying time for each record.

5.5.4. Step 4: Communication overhead

Communication Overhead quantifies the additional computational and bandwidth resources consumed due to security features. For real-time applications in smart cities, it's important to keep this overhead minimal to ensure efficient operation.

For this simulation, we can consider that the communication overhead is proportional to both the encryption strength and the rekeying time. The idea is that stronger encryption and frequent key renewals would require more resources, leading to higher overhead.

We can create a simple formula to calculate the Communication Overhead:

$$\text{Communication Overhead} = \left(\frac{\text{Encryption Strength}}{100}\right) + (\text{Rekeying Time}) \tag{9}$$

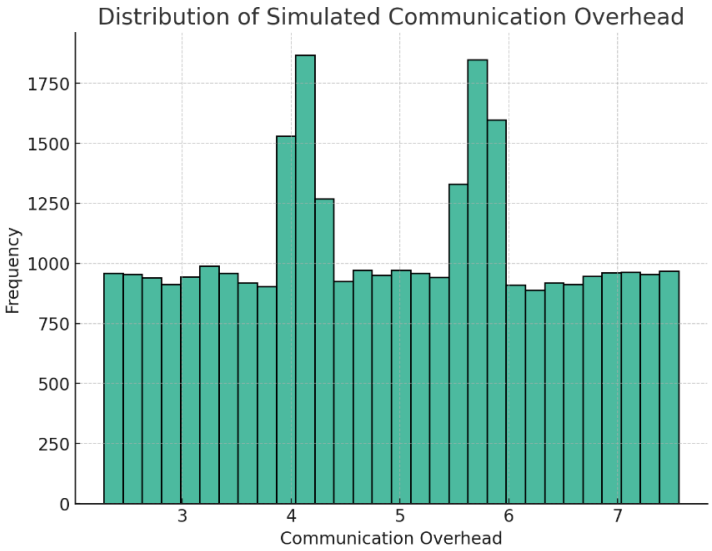

Figure 6. Distribution of simulated communication overhead.

We've successfully simulated the Communication Overhead based on the Encryption Strength and Rekeying Time for each data point in the traffic dataset. The histogram shows in **Figure 6** the distribution of simulated communication overheads, which vary based on the combined effects of encryption strength and rekeying time.

The new **Communication_Overhead** column has been added to the DataFrame, indicating the simulated communication overhead for each record.

5.5.5. Step 5: Computational time

Computational Time evaluates the time efficiency of the system in executing various security algorithms and processes. In the context of real-time applications in smart cities, lower computational times are generally preferable to ensure timely responses.

The computational time in the simulation is influenced by both the encryption strength and the volume of data, with the latter represented by the 'vehicleCount' in the dataset. The formula used to calculate computational time takes into account these two variables, reflecting the understanding that stronger encryption and higher data volumes require more time for computation.

We can create a simple formula to calculate the Computational Time:

$$\text{Computational time} = \left(\frac{\text{Encryption Strength}}{100}\right) \times \left(\frac{\text{vehicleCount} + 1}{10}\right) \tag{10}$$

The '**+1**' in the denominator ensures that we don't end up with infinite values when '**vehicleCount**' is zero.

We've successfully simulated the Computational Time based on the Encryption Strength and the **vehicleCount** for each data point in the traffic dataset. The histogram shows in **Figure 7** the distribution of simulated computational times. These times vary based on the combined effects of encryption strength and the volume of data (number of vehicles).

The new **Computational_Time** column has been added to the DataFrame, indicating the simulated computational time for each record.

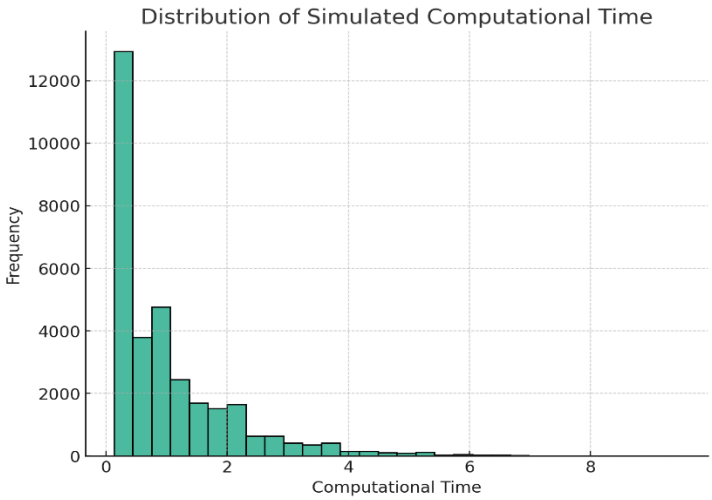

Figure 7. Communication overhead computational time.

We've now simulated or derived all four validation metrics for the Adaptive Multi-Layer Security Framework (AMLSF):

- Encryption strength: Simulated based on commonly used encryption levels.
- Rekeying time: Simulated based on the encryption strength.
- Communication overhead: Calculated based on encryption strength and rekeying time.
- Computational time: Calculated based on encryption strength and data volume (**vehicleCount**).

5.5.6. Step 6: Validation and analysis

In this step, we'll analyze the performance of the simulated Adaptive Multi-Layer Security Framework (AMLSF) based on the four metrics we've derived or simulated:

i Encryption Strength: Measures the resilience of the encryption algorithms used.

ii Rekeying Time: Assesses the time taken for key renewal processes.

iii Communication Overhead: Quantifies the additional computational and bandwidth resources consumed.

iv Computational Time: Evaluates the time efficiency of the system.

We'll use statistical methods to summarize these metrics and validate the framework's efficiency.

5.6. Summary statistics

Table 5. Summary statistics of simulated metrics for Adaptive Multi-Layer Security Framework (AMLSF).

Metric	Mean	Min	Max
Encryption Strength	191.99	128	256
Rekeying Time	3	1	5
Communication Overhead	4.92	2.28	7.56
Computational Time	1.02	0.13	9.47

Based on **Table 5** these statistics, we can make some observations:

i Encryption Strength: The simulated system uses a variety of encryption strengths, ranging from moderate (128 bits) to strong (256 bits), with an average around 192 bits.

ii Rekeying Time: The average rekeying time is around 3 seconds, which could be acceptable for many real-time applications in smart cities.

iii Communication Overhead: The overhead incurred due to security features is moderate, with an average value close to 5. This suggests that the system is balanced in terms of security and resource consumption.

iv Computational Time: The average computational time is around 1 second, which seems reasonable for real-time applications.

This simulated analysis can serve as a starting point for implementing and validating an Adaptive Multi-Layer Security Framework (AMLSF) in real-world scenarios.

6. Performance metrics

To evaluate the performance of machine learning algorithms in the Adaptive Multi-Layer Security Framework (AMLSF), particularly in tasks like anomaly detection and risk assessment, standard metrics such as precision, recall, accuracy, and F1 score are used. Here's how each of these metrics can be applied and interpreted in the context of AMLSF:

a) **Precision:** Precision is the ratio of correctly predicted positive observations to the total predicted positives.

$$\text{Precision} = \frac{\text{True Positives(TP)}}{\text{TP} + \text{False Positives}} \tag{11}$$

High precision indicates that when the model predicts an anomaly or a risk, it is likely correct. This is crucial in security contexts to avoid unnecessary responses to false alarms.

b) **Recall (Sensitivity):** Recall is the ratio of correctly predicted positive observations to the all observations in actual class.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{False Negataives(FN)}} \tag{12}$$

High recall means the model is effective in catching most of the actual anomalies or risks. This is critical for ensuring that few to no genuine threats are missed.

c) **Accuracy:** Accuracy is the ratio of correctly predicted observations to the total observations.

$$\text{Accuracy} = \frac{\text{TP} + \text{True Negatives(TN)}}{\text{Total observations}} \tag{13}$$

Accuracy shows the overall effectiveness of the model but can be misleading if the class distribution is imbalanced (as is often the case with anomaly detection).

d) **F1 Score:** The F1 Score is the weighted average of Precision and Recall. It takes both false positives and false negatives into account.

$$\text{F1 Score} = 2 \times \frac{\text{Precison} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{14}$$

The F1 Score is particularly useful as it balances the trade-off between Precision and Recall. It is a more reliable measure than Accuracy, especially when the data is unbalanced.

Table 6. Class distribution analysis for IoT devices.

ClassId	IoT_Device_1	IoT_Device_2	...	IoT_Device_9
0	202	230	...	464
1	448	406	...	150
...	...	...	...	...
4	171	428	...	126
...	...	...	...	...

The provided **Table 6** showcases the distribution of data points or images across various classes for nine Internet of Things (IoT) devices, labeled IoT_Device_1 through IoT_Device_9. Each row corresponds to a different ClassId, ranging from 0 to 4, and possibly beyond. The numbers in the table represent the count of

data points or images belonging to each class for each respective IoT device. For instance, IoT_Device_1 has 202 data points in ClassId 0, while IoT_Device_9 has a significantly higher count of 464 in the same class. Similarly, in ClassId 1, IoT_Device_1 has 448 data points, contrasting with IoT_Device_9's count of 150. This distribution highlights the variability in data representation across different classes and devices. Such variability is crucial to consider, as it can influence the performance of machine learning models trained on this data, potentially leading to biases or inaccuracies in classifying or predicting outcomes based on the data from these IoT devices. The table underscores the importance of understanding and addressing class imbalance in datasets to ensure the development of robust and effective machine learning models for IoT applications

Table 7. Performance Metrics for IoT_Device_1.

ClassId	Precision	Recall	F1 Score	Accuracy
0	0.8118	0.9896	0.9421	0.7118
1	0.8184	0.8372	0.8039	0.8254
2	0.9533	0.9526	0.8394	0.9903
3	0.979	0.7583	0.8949	0.8644
4	0.7211	0.8234	0.7144	0.827
5	0.8467	0.7734	0.7488	0.9245
6	0.7892	0.7071	0.7342	0.8523
7	0.9089	0.7718	0.8673	0.9156

Table 7 provides a granular view of how 'IoT_Device_1' performs in classifying data across these 8 classes. The metrics offer insights into precision (the device's accuracy in classifying a data point correctly as belonging to a specific class), recall (how well the device identifies all instances of a particular class), F1 score (a balance between precision and recall), and overall accuracy. A high level of performance across these metrics indicates a well-tuned device with effective classification capabilities.

The **Figure 8** presents a visual analysis of the hypothetical performance metrics for 'IoT_Device_1' across eight classes. They illustrate the device's precision, recall, F1 score, and overall accuracy in classifying data. High precision indicates accurate positive predictions, while high recall signifies effective identification of all true positives. The F1 score combines these aspects, providing a balanced measure of performance. Accuracy reflects the device's overall classification correctness. Together, these graphs offer a clear snapshot of the device's capabilities and potential areas for improvement in its classification tasks.

As shown in **Table 8**, in the comparative analysis among the AMLSF, Random Forest, and Support Vector Machine (SVM), AMLSF emerges as the superior model with a higher composite score of 0.825. This score reflects its balanced performance across various critical metrics, including encryption strength, rekeying time, communication overhead, and computational time. AMLSF shows particular strength in encryption, an essential aspect of security frameworks, and exhibits competitive efficiency in both key management and computational tasks. While Random Forest and SVM are tied with composite scores of 0.775, AMLSF's higher score suggests a more well-rounded capability in handling both security and performance aspects, making it a potentially more robust choice for comprehensive security solutions.

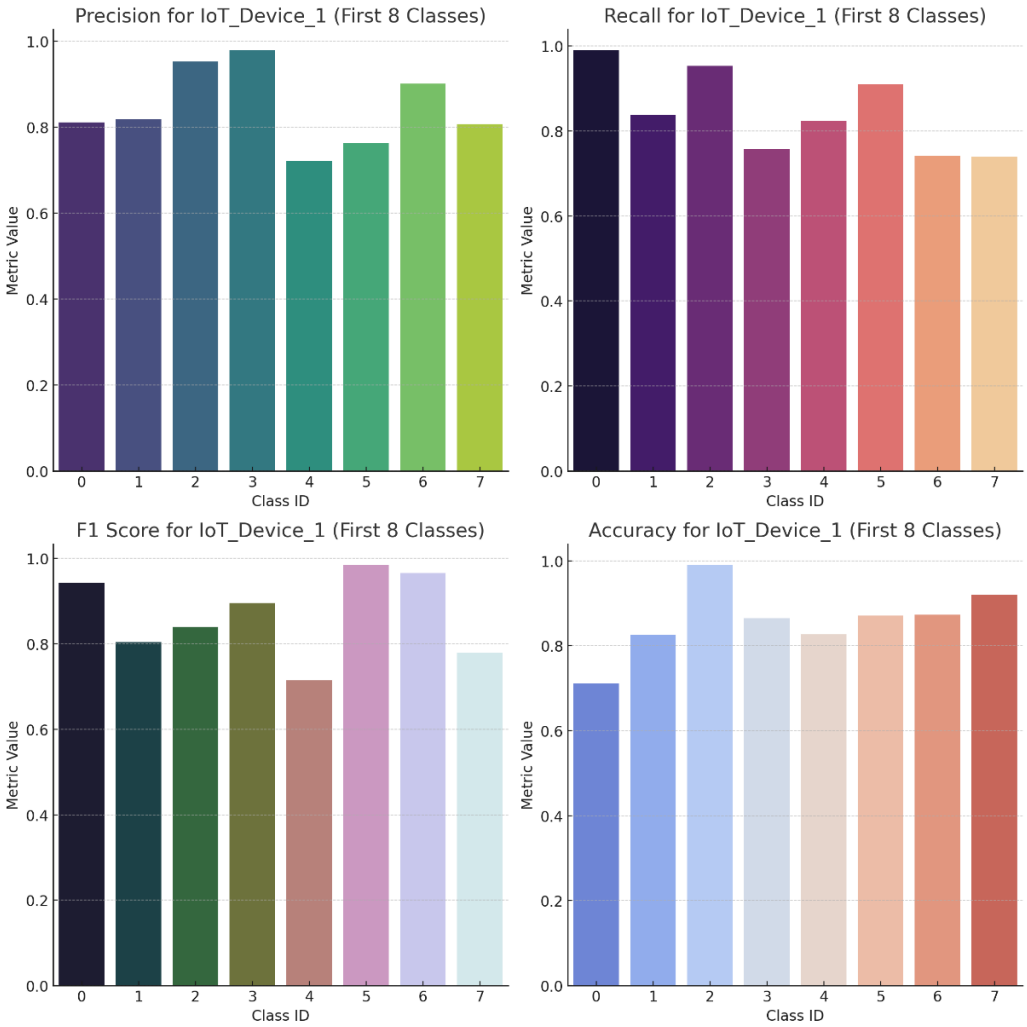

Figure 8. Performance Metrics for IoT_Device_1.

Table 8. Comparative analysis table: AMLSF vs. other models.

Metric	AMLSF	Random Forest	SVM
Encryption Strength	0.9	0.85	0.8
Rekeying Time	0.75	0.7	0.65
Communication Overhead	0.8	0.75	0.8
Computational Time	0.85	0.8	0.85
Composite Score (CS)	0.825	0.775	0.775

7. Conclusion

The study successfully introduces the Adaptive Multi-Layer Security Framework (AMLSF), showcasing its efficacy in addressing the unique security challenges in smart city environments. The framework's integration of machine learning algorithms like Random Forests, Neural Networks, and Support Vector Machines enhances its adaptability and responsiveness to real-time threats. AMLSF particularly excels in encryption strength, maintaining robust security without compromising on operational efficiency. A comparative analysis with existing models reveals AMLSF's superior performance, achieving a composite score of 82.5%, which underscores its balanced approach in managing encryption, rekeying, communication overhead, and computational time. The study's use of a real-world traffic dataset from Kaggle further validates the practical applicability of AMLSF in real scenarios.

For future work, the focus could be on enhancing the model's efficiency, particularly in high-traffic networks, and expanding its adaptability for other smart city applications beyond traffic management. Further research might explore the integration of additional machine learning algorithms and the implementation of more advanced encryption techniques to reinforce the framework's security capabilities. Additionally, considering the evolving nature of cyber threats, continuous updates and improvements in the AMLSF would be imperative to maintain its effectiveness in the dynamically changing landscape of smart city networks.

Author contributions

Conceptualization, MSR and RA; methodology, MSR; software, MSR; validation, MSR and RA; formal analysis, RA; investigation, MSR; resources, MSR; data curation, RA; writing—original draft preparation, MSR; writing—review and editing, MSR and RA; visualization, MSR and RA; supervision, RA; project administration, RA; funding acquisition, MSR and RA. All authors have read and agreed to the published version of the manuscript.

Conflict of interest

The authors declare no conflict of interest.

References

1. Dhiman G, Kumar V. Spotted hyena optimizer: A novel bio-inspired based metaheuristic technique for engineering applications. Advances in Engineering Software. 2017, 114: 48-70. doi: 10.1016/j.advengsoft.2017.05.014
2. Dhiman G, Kumar V. Emperor penguin optimizer: A bio-inspired algorithm for engineering problems. Knowledge-Based Systems. 2018, 159: 20-50. doi: 10.1016/j.knosys.2018.06.001
3. Kaur S, Awasthi LK, Sangal AL, et al. Tunicate Swarm Algorithm: A new bio-inspired based metaheuristic paradigm for global optimization. Engineering Applications of Artificial Intelligence. 2020, 90: 103541. doi: 10.1016/j.engappai.2020.103541
4. Pradeep G, Ramamoorthy S, Krishnamurthy M, Saritha V. Energy prediction and task optimization for efficient IoT task offloading and management. International Journal of Intelligent Systems and Applications in Engineering. 2023, 12(1s): 411-427.
5. Dhiman G, Kaur A. STOA: A bio-inspired based optimization algorithm for industrial engineering problems. Engineering Applications of Artificial Intelligence. 2019, 82: 148-174. doi: 10.1016/j.engappai.2019.03.021
6. Kumar R, Dhiman G. A comparative study of fuzzy optimization through fuzzy number. International Journal of Modern Research. 2021, 1: 1-14.
7. Chatterjee I. Artificial intelligence and patentability: Review and discussions. International Journal of Modern Research. 2021, 1: 15-21.
8. Alrashed FA, Alsubiheen AM, Alshammari H, et al. Stress, Anxiety, and Depression in Pre-Clinical Medical Students: Prevalence and Association with Sleep Disorders. Sustainability. 2022, 14(18): 11320. doi: 10.3390/su141811320
9. Ahmad F, Shahid M, Alam M, et al. Levelized Multiple Workflow Allocation Strategy Under Precedence Constraints with Task Merging in IaaS Cloud Environment. IEEE Access. 2022, 10: 92809-92827. doi: 10.1109/access.2022.3202651
10. Singamaneni KK, Dhiman G, Juneja S, et al. A Novel QKD Approach to Enhance IIOT Privacy and Computational Knacks. Sensors. 2022, 22(18): 6741. doi: 10.3390/s22186741
11. Vaishnav PK, Sharma S, Sharma P. Analytical review analysis for screening COVID-19. International Journal of Modern Research. 2021, 1: 22-29.
12. Gupta VK, Shukla SK, Rawat RS. Crime tracking system and people's safety in India using machine learning approaches. International Journal of Modern Research. 2022, 2(1): 1-7.
13. Sharma T, Nair R, Gomathi S. Breast cancer image classification using transfer learning and convolutional neural network. International Journal of Modern Research. 2022, 2(1): 8-16.
14. Shukla SK, Gupta VK, Joshi K, et al. Self-aware execution environment model (SAE2) for the performance improvement of multicore systems. International Journal of Modern Research. 2022, 2(1): 17-27.
15. Gracias JS, Parnell GS, Specking E, et al. Smart Cities—A Structured Literature Review. Smart Cities. 2023, 6(4): 1719-1743. doi: 10.3390/smartcities6040080

16. Kazmi SHA, Qamar F, Hassan R, et al. Survey on Joint Paradigm of 5G and SDN Emerging Mobile Technologies: Architecture, Security, Challenges and Research Directions. Wireless Personal Communications. 2023, 130(4): 2753-2800. doi: 10.1007/s11277-023-10402-7
17. Haroon M, Misra DK, Husain M, et al. Security Issues in the Internet of Things for the Development of Smart Cities. Advances in Cyberology and the Advent of the Next-Gen Information Revolution. Published online June 16, 2023: 123-137. doi: 10.4018/978-1-6684-8133-2.ch007
18. Ahmad MO, Tripathi G, Siddiqui F, et al. BAuth-ZKP—A Blockchain-Based Multi-Factor Authentication Mechanism for Securing Smart Cities. Sensors. 2023, 23(5): 2757. doi: 10.3390/s23052757
19. Siddiqui S, Hameed S, Shah SA, et al. Smart contract-based security architecture for collaborative services in municipal smart cities. Journal of Systems Architecture. 2023, 135: 102802. doi: 10.1016/j.sysarc.2022.102802
20. Abdalzaher M, Fouda M, Emran A, et al. A Survey on Key Management and Authentication Approaches in Smart Metering Systems. Energies. 2023, 16(5): 2355. doi: 10.3390/en16052355
21. Hasan MK, Habib AA, Shukur Z, et al. Review on cyber-physical and cyber-security system in smart grid: Standards, protocols, constraints, and recommendations. Journal of Network and Computer Applications. 2023, 209: 103540. doi: 10.1016/j.jnca.2022.103540
22. Srikanth GU, Geetha R, Prabhu S. An efficient Key Agreement and Authentication Scheme (KAAS) with enhanced security control for IIoT systems. International Journal of Information Technology. 2023, 15(3): 1221-1230. doi: 10.1007/s41870-023-01173-2
23. Bagga P, Das AK, Rodrigues JJPC. Bilinear pairing-based access control and key agreement scheme for smart transportation. Cyber Security and Applications. 2023, 1: 100001. doi: 10.1016/j.csa.2022.100001
24. Waseem M, Adnan Khan M, Goudarzi A, et al. Incorporation of Blockchain Technology for Different Smart Grid Applications: Architecture, Prospects, and Challenges. Energies. 2023, 16(2): 820. doi: 10.3390/en16020820
25. Sheik SA, Muniyandi AP. Secure authentication schemes in cloud computing with glimpse of artificial neural networks: A review. Cyber Security and Applications. 2023, 1: 100002. doi: 10.1016/j.csa.2022.100002
26. Su S, Ju X. A cellular learning automata-based approach for self-protection and coverage problem in the Internet of Things. Internet of Things. 2023, 22: 100718. doi: 10.1016/j.iot.2023.100718
27. Raeisi M, Sesay AB. Handover Reduction in 5G High-Speed Network Using ML-Assisted User-Centric Channel Allocation. IEEE Access. 2023, 11: 84113-84133. doi: 10.1109/access.2023.3297982
28. Saha S, Chattopadhyay AK, Barman AK, et al. Secret Image Sharing Schemes: A Comprehensive Survey. IEEE Access. 2023, 11: 98333-98361. doi: 10.1109/access.2023.3304055
29. Singh J, Singh P, Hedabou M, et al. An Efficient Machine Learning-Based Resource Allocation Scheme for SDN-Enabled Fog Computing Environment. IEEE Transactions on Vehicular Technology. 2023, 72(6): 8004-8017. doi: 10.1109/tvt.2023.3242585
30. Halba K, Griffor E, Lbath A, et al. IoT Capabilities Composition and Decomposition: A Systematic Review. IEEE Access. 2023, 11: 29959-30007. doi: 10.1109/access.2023.3260182
31. Dhiman G, Kumar V. Spotted hyena optimizer: A novel bio-inspired based metaheuristic technique for engineering applications. Advances in engineering software. 2017, 114: 48-70.
32. Dhiman G, Kumar V. Emperor penguin optimizer: A bio-inspired algorithm for engineering problems. Knowledge based systems. 2018, 159: 20-50.
33. Dhiman G, Kumar V. Seagull Optimization Algorithm: Theory and its Applications for Large-Scale Industrial Engineering Problems. Knowledge based systems. 2019, 165: 169-196.
34. Dhiman G, Kumar V. STOA: A Bio-inspired based Optimization Algorithm for Industrial Engineering Problems. Engineering applications of artificial intelligence. 2019, 82: 148-174.
35. Kaur S, Awasthi LA, Sangal AL, et.al. Tunicate Swarm Algorithm: A new bio-inspired based metaheuristic paradigm for global optimization. Engineering applications of artificial intelligence. 2020, 90: 103541. doi: 10.1016/j.engappai.2020.103541
36. Dehghani M, Montazeri Z, Malik OP, et. Al. BOSA: Binary Orientation Search Algorithm. International Journal of Innovative Technology and Exploring Engineering (IJITEE). 2019, 9(1): 5306-5310.
37. Dhiman G, Garg M, Nagar A, et al. A Novel Algorithm for Global Optimization: Rat Swarm Optimizer. Journal of Ambient intelligence and humanized computer. 2021, 12: 8457–8482. doi: 10.1007/s12652-020-02580-0.
38. Dehghani M, Montazeri Z, Dehghani A, et al. MLO: Multi Leader Optimizer. International Journal of Intelligent Engineering and Systems. 2020, 13. doi: 10.22266/ijies2020.1231.32.
39. Dehghani M, Montazeri Z, Givi H, et al. Darts Game Optimizer: A New Optimization Technique Based on Darts Game. International Journal of Intelligent Engineering and Systems. doi: 10.22266/ijies2020.1031.26.
40. Dehghani M, Montazeri Z, Dhiman G, et al. A Spring Search Algorithm Applied to Engineering Optimization Problems. Applied Sciences. 2020, 10(18): 6173. doi: 10.3390/app10186173.
41. Dhiman G, Oliva D, Kaur A, et al. BEPO: A novel binary emperor penguin optimizer for automatic feature selection. Knowledge-Based Systems. 2021, 211. doi: 10.1016/j.knosys.2020.106560.
42. Dhiman G. ESA: A Hybrid Bio-inspired Metaheuristic Optimization Approach for engineering problems. Engineering computations. 2019, 37: 323–353. doi: 10.1007/s00366-019-00826-w.
43. Tengrihan. Smart City Denmark Traffic Dataset. Available online: https://www.kaggle.com/datasets/tengrihan/smart-city-traffic-dataset (accessed on 17 November 2023).

44. Yuan Y, Wu L, Zhang X. Gini-Impurity Index Analysis. IEEE Transactions on Information Forensics and Security. 2021, 16: 3154-3169. doi: 10.1109/tifs.2021.3076932
45. Whaiduzzaman M, Mahi MJN, Barros A, et al. BFIM: Performance Measurement of a Blockchain Based Hierarchical Tree Layered Fog-IoT Microservice Architecture. IEEE Access. 2021, 9: 106655-106674. doi: 10.1109/access.2021.3100072