FROM PERCEIVED TRUST TO PROVABLE SECURITY: A MULTI-TIERED FRAMEWORK FOR THE DESIGN, ASSESSMENT, AND VERIFICATION OF SECURITY IN RESOURCE-CONSTRAINED IOT ECOSYSTEMS

by

VICTORINE CLOTILDE WAKAM YOUNANG

A dissertation submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE AND INFORMATICS

2026

Oakland University
Rochester, Michigan

Doctoral Advisory Committee:

Amartya Sen, Ph.D., Chair
Huirong Fu, Ph.D.
Sunny Raj, Ph.D.
Dorin Drignei, Ph.D.

To those who perceive technology not merely as a tool, but as a bridge to a safer and more empowered world. May your innovations be guided by a commitment to integrity and the betterment of the global community.

And to the persistent dreamers: those who meet every setback with quiet resilience and every challenge with renewed determination. May your journey inspire others to persevere, and may your successes pave the way for generations to come.

ACKNOWLEDGMENTS

The completion of this dissertation marks the culmination of a journey defined by intellectual growth and perseverance. This achievement was made possible through the guidance and support of a dedicated community of mentors, family, and friends.

My sincerest gratitude goes to my Academic Advisor and Committee Chair, Dr. Amartya Sen. His guidance and unwavering mentorship provided the bedrock for this research, profoundly shaping my development as a researcher.

I am equally indebted to the members of my Dissertation Committee: Dr. Dorin Drignei, Dr. Huirong Fu, and Dr. Sunny Raj. I want to thank them deeply for their technical insights and steadfast encouragement throughout this journey. Their constructive feedback and expertise were instrumental in refining the technical depth of this work, and I am truly grateful for the personal and academic guidance they provided to help me navigate the complexities of this research.

I am grateful to Oakland University, its faculty, and the Department of Computer Science and Engineering for providing the academic environment and resources necessary for this undertaking. I extend a special thank you to the International Students and Scholars Office (ISSO); their support was instrumental in helping me navigate the challenges of being an international student. I also acknowledge the financial support provided by the university through Teaching Assistantships, which allowed me to focus on my research.

This journey would have been impossible without my family's steadfast love. To my father, thank you for your sacrifices and for your constant encouragement. To my siblings and extended family, thank you for your prayers and for being my constant source of strength.

To my friends, and research colleagues, thank you for the intellectual camaraderie, the listening ears, and the support during the most demanding periods of this work. Our shared experiences made the long working hours much more bearable.

Finally, to everyone who offered a kind word or a helping hand along the way, I am deeply grateful. This milestone is as much a result of your kindness as it is of my own efforts.

Victorine Clotilde Wakam Younang

ABSTRACT

FROM PERCEIVED TRUST TO PROVABLE SECURITY: A MULTI-TIERED FRAMEWORK FOR THE DESIGN, ASSESSMENT, AND VERIFICATION OF SECURITY IN RESOURCE-CONSTRAINED IOT ECOSYSTEMS

by

Victorine Clotilde Wakam Younang

Adviser: Amartya Sen, Ph.D.

The widespread adoption of Internet of Things (IoT) devices and Autonomous Vehicles (AVs) has created a significant assurance gap between what users expect and what is technically feasible in resource-limited environments. As these systems transition from prototypes to critical infrastructure, it becomes essential not only to make them efficient, but also to ensure they are demonstrably secure and capable of earning public trust. This dissertation introduces a multi-layered research framework that tackles this gap across four stages of the technology lifecycle: socio-technical, operational, security-evaluative, and formal-foundational.

The first phase situates the problem by examining the disconnect between governmental policies and end-user priorities related to AV safety and privacy. After identifying that public trust is frequently weakened by opaque automated decision-making, the work moves in the second phase, which focuses on security, especially assessing security risk in IoT networks, by leveraging complex numbers Bayesian Attack Graphs to model and quantify unpredictable attacker behavior. Once the risk can be assessed, the most common step is to mitigate the risk. The work focuses on assessing how realistically security models can be deployed on edge devices, especially Intrusion Detection Systems (IDS)

models, on vehicles' CAN buses, introducing new feasibility metrics, that account for the IDS models' accuracy, as well as the CAN bus resources restrictions. The final phase directly addresses the core of the security assurance gap by connecting imperative software implementations to rigorous mathematical proofs. It introduces the Tiered Isomorphic Alignment (TIA) framework, which helps automate the translation of Python programs into Isabelle/HOL formal specifications, and introduces CryptoGraph, a neuro-symbolic system that employs Graph Attention Networks to analyze the security of non-standard lightweight cryptographic ciphers. Taken together, these contributions establish a scalable, end-to-end methodology for designing the next generation of IoT ecosystems, one that is grounded in mathematical correctness and automated security assessment, capable of maintaining and enhancing public trust.

TABLE OF CONTENTS

TABLE OF CONTENTS—Continued

TABLE OF CONTENTS—Continued

TABLE OF CONTENTS—Continued

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

The Internet of Things (IoT) has evolved from an abstract paradigm to a widespread technology infrastructure composed of physical devices equipped with sensors, actuators, and software [2]. By connecting the digital and physical worlds, IoT has enabled major advances in areas such as healthcare [3], agriculture [4], transportation [5], Industry 4.0 [6], and smart cities [7]. A prominent example of this evolution is the rise of Autonomous Vehicles (AVs), representing a key role in the development of Intelligent Transportation Systems (ITS) [8]. Despite these advances, the transition toward fully autonomous, IoT-driven infrastructures is constrained by a multidimensional security gap. This gap appears when technological innovation outpaces our ability to build public trust, efficiently manage limited hardware resources, and rigorously prove the security of the underlying software logic. This dissertation addresses this gap using a multi-layered workflow that begins at the socio-technical level, focusing on people, policy, and technology, and extends down to the formal foundational level, focusing on mathematical models and proofs.

1.1 Phase I: Socio-Technical Trust and Policy Alignment on AVs

The fully AVs, once successfully tested and implemented, will allow efficient ways of transporting people and goods, managing the supply chain, and providing a unique user experience by accommodating a user's customized needs and preferences for provisioned services [9] [10]. Further, fully AVs will reduce carbon emissions and improve traffic management through carpooling and ride-sharing services [11]. Despite this potential, public skepticism remains high; a survey from 2019, 71% of participants expressed fear to ride in an AV. Another study from 2018 reveals that 60% of participants would not feel safe sharing the road with AVs [12]. A major source of this distrust is the opaque nature of these

systems, where users have little insight into or control over the internal decision-making processes of such systems.

The motivation for this phase arises from a key limitation in the existing literature. Current research tends to examine public sentiment through various methods such as surveys [13] [14] [15], or social media analysis [16] [17]. As a result, they mainly focus on the user concerns, lacking comparative studies that directly analyze the alignment (or misalignment) between what governments prioritize in policy and the users' concerns in practice. This phase therefore, seeks to determine whether legislative bodies adequately recognize and address the public's safety and privacy concerns on social media platforms such as Twitter and Reddit.

While identifying misalignment between policy and public sentiment offers a roadmap for societal acceptance, the technical basis of trust ultimately depends on system resilience to attacks. Policy frameworks can assign liability but cannot prevent cyber-physical breaches. Accordingly, Phase II shifts from the socio-technical to focus on risk assessment, necessary to rigorously model and quantify the complex, interdependent attack paths that threaten network integrity.

1.2 Phase II: Security Risk Assessment in IoT Networks

Securing resource-constrained infrastructures requires rigorously characterizing how vulnerabilities interact before designing defenses. In large-scale IoT networks, risk interdependence is critical; for instance, compromising a single node in a Connected and Autonomous Vehicle (CAV) network can trigger cascading attacks. This phase evaluates the limitations of traditional Risk Assessment (RA) methods for capturing such dynamics. Although Bayesian Networks and Bayesian Attack Graphs (BAGs) with real-valued probabilities are widely used to quantify vulnerabilities [18] [19] [20] [21], they are often

inadequate in IoT settings because they cannot model adversarial backtracking or the constructive and destructive interference among concurrent exploits. To address these gaps, we introduce complex-valued probabilities to mathematically capture these interactions, enabling more rigorous risk quantification and a stronger foundation for strategic security planning.

While Phase II focuses on strategically identifying and modeling risk, the third phase addresses the tactical challenge of deploying countermeasures to mitigate those risks. Identifying a threat is only part of the solution; ensuring that the resulting countermeasure is practically deployable without crippling the system is equally critical. Phase III focuses on the feasibility gap inherent in proposed Intrusion Detection Systems (IDS) for In-Vehicle Networks.

1.3 Phase III: Operational Model Feasibility and Deployment

While Machine Learning (ML) enables powerful anomaly-based IDS to mitigate cyber-physical risks, these models frequently impose substantial computational overhead. In the highly resource-constrained environment of an In-Vehicle Network (IVN), an IDS model that achieves high accuracy but introduces excessive latency becomes a liability rather than an asset. To reconcile this trade-off, we introduce the Custom Feasibility Score (CFS). This metric serves as a necessary bridge between theoretical security models and physical system constraints, allowing engineers to determine strictly whether an IDS can operate within the CPU and memory limits of a vehicle's Electronic Control Unit (ECU) without degrading mission-critical performance.

Although resource allocation optimization ensures that a system is available and efficient, it does not guarantee that the system is secure. As we maximize the operational

capacity of these networks, we simultaneously expand the attack surface or entry points, necessitating a shift from management to rigorous security risk assessment.

While the development of feasible, real-time Intrusion Detection Systems (IDS) provides a robust layer of defense for in-vehicle networks, they primarily operate at the message and traffic levels, and are designed to detect anomalies in systems that are already deployed and executing. However, the foundational security of an IoT ecosystem, including the encrypted telemetry and command structures within a vehicle, ultimately depends on a deeper layer, namely the underlying cryptographic primitives. Even with an effective IDS in place, the block ciphers responsible for securing data must be both correctly implemented and structurally sound, as minor design flaws like improper constant propagation or flawed key updates can lead to catastrophic vulnerabilities [22]. Two challenges exist in the current literature, notably a Formalizing Gap and an Evaluation Bottleneck. First, there is often insufficient expertise and tooling to manually verify that a cipher's implementation in Python faithfully adheres to its mathematical specification. Second, existing methods are inadequate for rapidly assessing the security of novel, non-standardized lightweight ciphers. Consequently, achieving genuine end-to-end security requires moving beyond perimeter and network-level defenses to the cryptographic core itself. This entails automating both the formalization of implementation code and the structural analysis of the ciphers, thereby enabling systematic reasoning about their correctness and security properties.

1.4 Phase IV: Formal Foundations and Neuro-Symbolic Logic for Block Ciphers

The final phase of this dissertation focuses on the foundational security of IoT ecosystems by examining the automated lifecycle of lightweight block ciphers. These ciphers, spanning Feistel, Substitution-Permutation Network (SPN), and

Addition-Rotation-XOR (ARX) families [23], serve as core mechanisms for preserving privacy in resource-constrained environments. However, their security guarantees are highly sensitive to implementation errors that can compromise their underlying mathematical properties. This phase proposes a two-fold neuro-symbolic framework to strengthen these cryptographic foundations. First, it addresses the implementation gap through the proposed Tiered Isomorphic Alignment (TIA) framework. TIA is designed to mitigate the difficulty that current Large Language Models (LLMs) face with cross-paradigm induction—specifically, translating imperative program of block ciphers (e.g., in Python), into declarative formal logic. It does so by enforcing structural equivalence between imperative code and formal specifications across four semantic layers: lexical constants, mathematical primitives, structural round functions, and top-level orchestration. By curating aligned training data for code-specialized models such as DeepSeek-Coder and StarCoder2, TIA enables LLMs to move beyond superficial syntactic patterns toward genuine logical induction in Isabelle/HOL [24]. Second, this phase targets the evaluation bottleneck through the CryptoGraph framework. For non-standardized lightweight block ciphers, traditional manual cryptanalysis is both time-consuming and susceptible to human error. CryptoGraph leverages the formal specifications produced in the TIA pipeline and extracts Abstract Syntax Trees (ASTs) to capture the internal data flow of each cipher. Using a hybrid architecture that combines Graph Attention Networks (GATs) with Multi-Layer Perceptrons (MLPs), CryptoGraph conducts a glass-box structural analysis to classify the security strength of previously unseen cipher architectures (e.g., HIGHT, SKINNY) with high fidelity. By integrating the rigor of formal proof assistants with the adaptivity of Machine Learning, this phase ensures that the cryptographic primitives underpinning IoT ecosystems are not merely operational, but are subject to systematic, provable security guarantees. Figure s

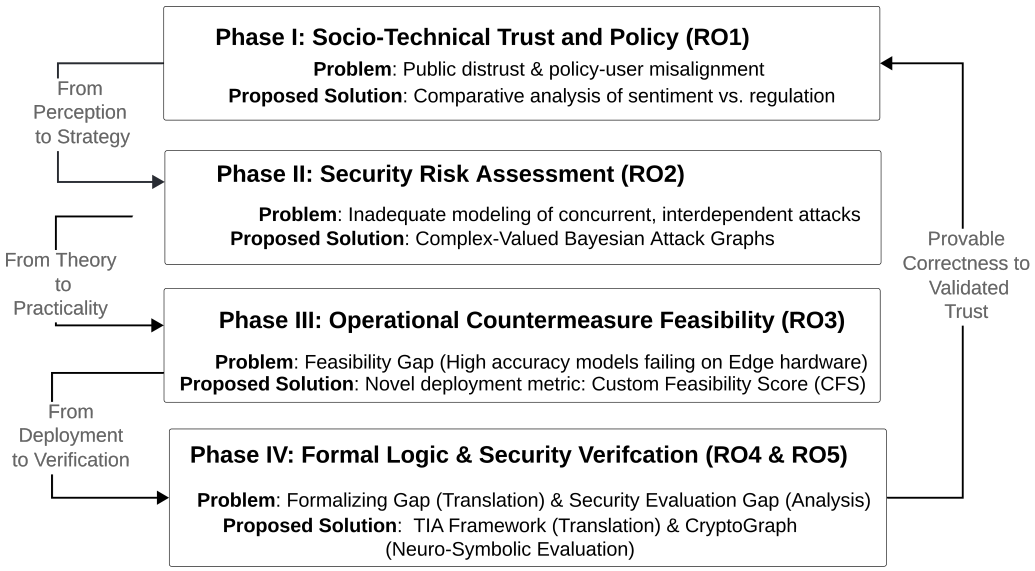

Figure 1.1: From Perceived Trust to Provable Security: A Framework for the Design, Assessment, and Verification of Resource-Constrained IoT Ecosystems

1.5 RESEARCH OBJECTIVES (RO)

The overarching goal of this dissertation is to close the assurance gap in autonomous IoT ecosystems by developing a multi-tiered framework that aligns technological innovation with societal trust, optimizes IoT network risk assessment, evaluates the feasibility of security countermeasures, and is grounded in formal mathematical correctness and automated security assessment. To realize this, the research is structured around five specific objectives, each corresponding to a critical layer in the system lifecycle.

1.5.1 RO1: Analyzing the Alignment between Public Concerns and Policy on AVs

The initial objective is to investigate the disconnect between the stakeholders of autonomous vehicle (AV) technology. Despite the promise of Intelligent Transportation Systems (ITS), public skepticism remains a barrier to adoption. This objective performs a comprehensive social media analysis of Twitter and Reddit, to categorize and quantify

non-event-driven safety and privacy concerns of end-users. Additionally, it synthesizes publicly available policy documents from global governing bodies (e.g., U.S. DOT, European Commission) to determine the focus of current regulatory efforts. Lastly, it conducts a comparative analysis to identify the similarities and differences between legislative directions and public concerns, to better inform future software and hardware requirements.

1.5.2 RO2: IoT Network Risk Assessment with BAGs and Complex Probabilities

To develop a strategic risk assessment framework for IoT networks, we are enhancing Bayesian Attack Graphs (BAGs) with complex-valued probabilities. This objective aims to mathematically quantify the constructive and destructive interference of concurrent attacks and model cyclic backtracking, providing a rigorous foundation for identifying vulnerabilities in interdependent systems.

1.5.3 RO3: Deployment Feasibility Analysis of ML-Based Intrusion Detection Systems (IDS)

Evaluating the operational feasibility of deployed countermeasures is essential for securing IoT Networks. This objective introduces the Custom Feasibility Score (CFS) to bridge the critical gap between theoretical model accuracy and the physical constraints (CPU, memory, latency) when deploying ML-based IDS on automotive Electronic Control Units.

1.5.4 RO4: Inducing formal grammars from imperative code

To provide the highest level of security assurance, this objective focuses on imperative versus declarative implementation gaps of lightweight cryptographic block ciphers, in order to strengthen cryptographic foundations. Specifically, this objective aims to design the Tiered Isomorphic Alignment (TIA) framework to enable Large Language Models (LLMs)

to perform hierarchical logic induction by automating the translation of imperative Python code into formal Isabelle/HOL specifications for cryptographic block ciphers.

1.5.5 RO5: Neuro-Symbolic Structural Security Evaluation

The final objective seeks to provide the highest level of security assurance, by improving early security assessment of cryptographic primitives. It creates a Glass-Box security evaluation framework, CryptoGraph, that assesses the cryptographic strength of lightweight block ciphers. This objective leverages Graph Neural Networks (GNNs) to analyze the Abstract Syntax Trees (ASTs) of formal specifications, enabling the automated, structural classification of security levels for non-standardized primitives.

The remainder of this dissertation is organized as follows. Chapter 2 provides the technical background necessary to understand the interdisciplinary nature of this research, covering IoT architectures, formal verification principles (Isabelle/HOL), and Graph Neural Networks. Chapter 3 provides the existing literature for each research objective, necessary to understand the current state of the art. Chapter 4 addresses RO1 and details the socio-technical analysis of trust in Autonomous Vehicles; based on our work in [25], it presents the methodology for social media mining and policy synthesis used to identify the trust gap. Chapter 5 addresses the RO2 by introducing a complex-valued Bayesian Attack Graph (BAG) framework to quantify cyclic risks in IoT networks [26], with an application on Connected and Autonomous vehicles (CAVs). Additionally, Chapter 6 introduces the Custom Feasibility Score (CFS) to evaluate the deployment readiness of ML-based Intrusion Detection Systems [27] on vehicles' CAN buses, addressing the RO3. Chapter 7 presents the Tiered Isomorphic Alignment (TIA) framework ([28]), which describes the methodology for automating the translation of imperative Python code into formal Isabelle/HOL specifications, using LLMs to address the RO4. While Chapter

8 presents CryptoGraph, detailing the hybrid GNN architecture used for the structural security evaluation of block ciphers [29], addressing RO5. Finally, Chapter 9 concludes the dissertation by summarizing the research findings and outlining future directions for automated formal verification in critical infrastructure.

CHAPTER TWO

TECHNICAL BACKGROUND

This chapter defines the theoretical and technical foundations, mathematical models, and logical frameworks needed to navigate the interdisciplinary scope of this research.

2.1 Autonomous Vehicles (AV) and Vehicle Networks

This section elaborates on Autonomous Vehicles (AVs), their components, networks, and levels.

2.1.1 Controller Area Network (CAN) bus Protocol

The CAN bus plays a vital role in the communication infrastructure of a vehicle, making it a key target for security attacks. Different countermeasures like encryption schemes and authentication methods have been proposed to mitigate these security attacks. These countermeasures prevent unauthorized access to the CAN bus and ensure that data confidentiality and integrity are uncompromised. However, these countermeasures only prevent known attacks and do not operate in real time, making these approaches reactive. Hence, there is a need for a real-time Intrusion Detection System (IDS). The integration of IDS in the CAN bus of a vehicle has facilitated the detection of known and unknown attacks in real time, making it widely adopted for securing the CAN bus. In this section, we present background information about the CAN bus protocol and IDS.

In a vehicle, an Electronic Control Unit (ECU) is an embedded system that controls a specific function, like the Engine Control Module (ECM) manages the engine operation, and the Brake Control Module (BCM) handles the braking system of a vehicle. The CAN bus is a communication protocol that enables ECUs to transmit and receive data with each other through message broadcasts. It provides a communication architecture that is easily

accessible and scalable, as it provides a single point of entry for communication with all the ECUs in the vehicle. The use of an arbitration mechanism (priority IDs) in the CAN bus ensures that real-time communication is supported. Additionally, it also ensures that message collision is avoided, such that safety-critical messages like airbag deployment data and brake system data are prioritized in the event of a vehicle collision.

The Controller Area Network (CAN) is made up of two layers: the physical layer and the data link layer. The physical layer defines the physical requirements, like the electrical signal level. The data link layer is responsible for data transmission over the physical layer. Error handling, message arbitration, and message formats are defined in the data link layer. In the CAN bus, messages are referred to as frames, and there are four types of frames: data frames, remote frames, error frames, and acknowledgment frames. The data frame is used to transmit data among ECUs. The remote frame is used to request data from an ECU. The error frame is used when an ECU detects an error, and the overload frame is used to provide a delay between data or remote frames when additional processing time is required. The commonly used frame in the CAN bus is the data frame.

The CAN bus data frame, as shown in Fig. 2.1, consists of the following fields: Start of Frame (SoF), arbitration field, control field, data field, Cyclic Redundancy Check (CRC) field, Acknowledge (ACK) field, and End of Frame (EoF). SoF denotes the start of a data frame. The arbitration field is made up of frame identifier bits that are used for unique message identification and message prioritization. The control field has the Data Length Code (DLC) that denotes the amount of data in the data field, and the data field has the data bytes (payload) that can be decoded as the message content. The CRC field checks for error detection during message transmission. The ACK field includes information that indicates a message has been received by an ECU, and the EoF marks the end of the data frame. The integration of CAN bus in vehicles ensures reliable and accurate transmission of data

between the ECUs in a vehicle. However, it is highly susceptible to security attacks, such as replay [30], Denial of Service (DoS) [31], diagnostic [32], bus injection [33] and spoofing attacks [34] due to physical access vulnerability, as it can be accessed through physical connection to devices [35], lack of built-in security features like message authentication and encryption [36], and a lack of intrusion detection mechanism. These attacks can lead to unstable communication in the CAN bus, which leads to an attacker taking control of safety-critical functions related to the performance of the vehicle, leading to physical harm to the users. Therefore, to address these security attacks and enable real-time detection of malicious activities, an Intrusion Detection System (IDS) has been introduced.

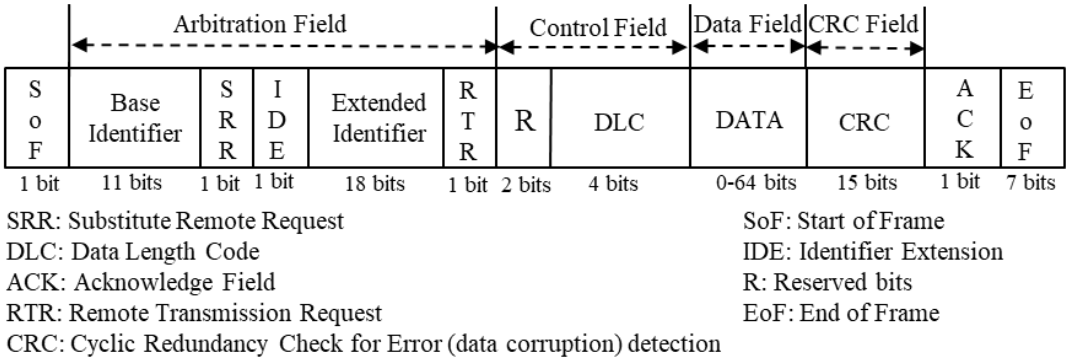

Figure 2.1: A CAN bus dataframe. This figure shows the structure of a CAN bus dataframe, which consists of various fields (SOF, arbitration field, control field, data field, CRC field, ACK, EOF) used for data transmission in the CAN network.

The implementation of an IDS in the CAN bus network provides real-time monitoring of the network traffic, making it easier to detect malfunctions or malicious activities.

2.1.2 Intrusion Detection Systems (IDS)

The IDS directly communicates with the CAN bus data frame to analyze the data for intrusion, and the authors in [1] have identified these anomaly detection sensors as a reliable way of identifying anomalies, as shown in Table 2.1. These detection sensors rely on information derived from defined collaborative device communication, network protocol rules, and multiple vehicle data sources, ensuring accurate and reliable information. Therefore, a deviation from the normal behavior of these sensors is a sign of an intrusion in a vehicle.

Sensor	Description
Consistency	Data from redundant sources is consistent
Correlation	Message correlation on different bus systems adheres to the specification
Formality	Correct message size, header and field size, field delimiters, checksum, etc.
Frequency	Timing behavior of messages is approved
Location	Message is allowed with respect to the dedicated bus system
Range	Compliance of payload in terms of data range
Plausibility	Message payload is plausible, no infeasible correlation with previous values
Protocol	Correct order, start-time, etc. of internal challenge-response protocols

Table 2.1: Anomaly Detection Sensor [1]. This table provides information about the different sensors in a vehicle that can be utilized to detect an intrusion in the CAN bus.

Based on the detection methodology adopted, an IDS can be categorized as signature-based (misuse) IDS and anomaly-based IDS [37]. A signature-based IDS involves monitoring the CAN bus network for known patterns of security attacks. An anomaly-based IDS monitors the CAN bus network to detect anomalies from the normal network behavior. The architecture of an IDS involves the following layers: data collection, data preprocessing, traffic analysis, decision, and reporting layer, as shown in Fig. 2.2. The data collection layer is directly connected to the CAN bus interface and involves the acquisition of data like message IDs, payloads, and data lengths that will be used for intrusion detection at the traffic analysis layer. Once the data is collected, it is routed

to the data preprocessing layer, where irrelevant data is removed, data is normalized for consistency, and important features like payload size and message frequency are extracted for intrusion detection. The extracted features are passed to the traffic analysis layer. In the traffic analysis layer, depending on the detection methodology adopted (signature-based or anomaly-based), the implemented algorithms process the data to identify deviations. Once a deviation has been detected, it is sent to the decision layer, where it is further observed to determine if the detected anomaly is a benign variation in traffic or an intrusion. Once the decision is made, the detected anomaly and decision are logged for further analysis, and the incident is reported.

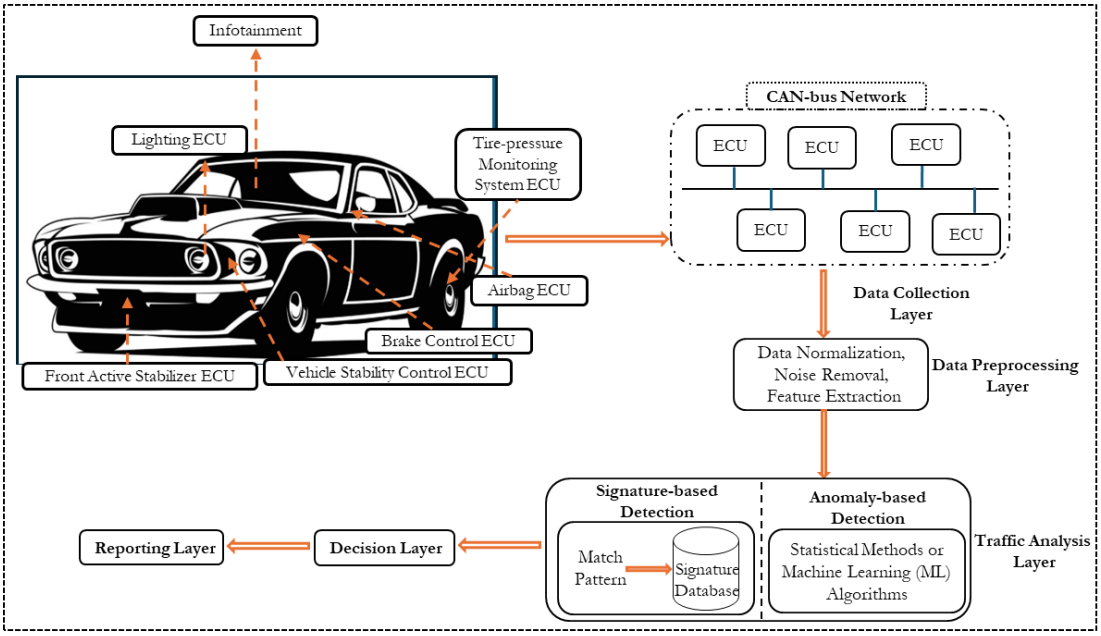

Figure 2.2: An architecture of Intrusion Detection System (IDS). This figure shows the different layers of an IDS and how the data flows between the different layers.

Signature-based IDS: In a signature-based IDS, the CAN bus network traffic is monitored and compared to a predefined list of attack patterns to identify intrusion in the

CAN bus. The predefined list of attack patterns consists of a comprehensive list of security attack signatures like irregular timing, specific message IDs, and message sequences that are stored in a database [38]. Once a signature is matched, intrusion has been detected in the CAN bus network traffic.

Anomaly-based IDS: In an anomaly-based Intrusion Detection System (IDS), the CAN bus network traffic is observed to identify deviations from its normal communication behavior. These deviations are considered signs of malicious activities by the anomaly-based IDS until proven otherwise. This makes it effective at identifying novel or zero-day attacks and highly adaptable to vehicle traffic patterns in an IVN. It is currently the most common approach for detecting intrusion in CAN bus [39].

2.1.3 Autonomous Vehicles (AV)

An Autonomous Vehicle (AV) is defined as a system capable of sensing its environment and operating without human involvement. In the perception stage, the AV uses sensors such as LiDAR, RADAR, and cameras to map the environment; in the planning stage, it processes this sensor data to generate trajectories and make decisions (e.g., yielding to a pedestrian); and in the control stage, it executes physical actions via actuators such as steering and braking.

2.1.4 Connected and Autonomous Vehicles (CAVs)

While an Autonomous Vehicle (AV) operates independently, a Connected and Autonomous Vehicle (CAV) integrates network connectivity to communicate with external entities, providing the primary vector for the remote attacks analyzed in the Risk Assessment (RO3) phase. CAVs utilize Vehicle-to-Everything (V2X) communication standards, including Vehicle-to-Vehicle (V2V) links for the direct exchange of kinematic

data (such as speed and location) to prevent collisions, Vehicle-to-Infrastructure (V2I) communication with traffic signals, road sensors, and edge servers, and Vehicle-to-Network (V2N) cellular connectivity (LTE/5G) for cloud services and over-the-air (OTA) updates. The convergence of autonomy (internal decision making) and connectivity (external data input) significantly expands the attack surface, motivating the proposed complex probability approach.

2.1.5 Autonomous Vehicle Levels

To contextualize the Trust Gap (RO1), we utilize the SAE J3016 taxonomy [40] to classify driving automation into three groups: Levels 0–2 (Driver Support), where the human performs part of the Dynamic Driving Task (DDT); Level 3 (Conditional Automation), where the system performs the entire DDT but requires human fallback; and Levels 4–5 (High/Full Automation), where the system performs the entire DDT and fallback without human intervention.

2.2 Risk Assessment Models: Bayesian Attack Graphs (BAGs)

We leverage Bayesian Attack Graphs to model probabilistic attack paths in order to quantify the likelihood and impact of vulnerabilities in an IoT network. A BAG is a Directed Acyclic Graph (DAG) where nodes represent security states and edges represent exploit probabilities [41]. The joint probability distribution is given by:

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | \text{Parents}(X_i))$$

This dissertation extends this model using Complex Probabilities to account for the constructive and destructive interference of concurrent attacks, addressing the limitations of static probability values in dynamic IoT environments.

2.2.1 Common Vulnerability Scoring System (CVSS):

Additionally, risk values in our framework are determined using the CVSS v3.1 specification [42], where the Base Score is derived from two main components: Exploitability—capturing Attack Vector (AV), Attack Complexity (AC), Privileges Required (PR), and User Interaction (UI), and Impact, which reflects the attack's effects on Confidentiality (C), Integrity (I), and Availability (A) of the targeted network.

2.3 Cryptographic Primitives and Formal Verification

This section establishes the mathematical definitions of block ciphers and the formal logic systems used to verify their correctness.

2.3.1 Block Ciphers

A block cipher is a deterministic, symmetric-key primitive that operates on fixed-length groups of bits, called blocks. Formally, a block cipher is a function $E : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$ that takes a k-bit key K and an n-bit plaintext block P to produce an n-bit ciphertext block C.

$$C = E_K(P)$$

The inverse function $D = E^{-1}$ allows for decryption such that $P = D_K(C)$. Modern block ciphers rely on Shannon's principles of Confusion (obscuring the relationship between the key and ciphertext) and Diffusion (dissipating the statistical structure of the plaintext over the ciphertext).

2.3.2 Lightweight Cryptography (LWC)

Lightweight cryptography refers to primitives optimized for constrained environments (e.g., RFID tags, sensor nodes) where standard primitives like AES are too

resource-intensive. As outlined in NIST [43], LWC aims to minimize Gate Equivalents (GE) in hardware and RAM/ROM usage in software. We focus on three specific architectural families, notably Feistel, ARX, and SPN. A Feistel network partitions the input block into two halves, (L_i, R_i). In each round i, the *right* half is transformed by a round function F and XORed with the *left* half. The halves are then swapped. The defining characteristic is that F does not need to be invertible for the cipher to be reversible.

$$L_{i+1} = R_i$$

$$R_{i+1} = L_i \oplus F(R_i, K_i)$$

Unlike Feistel networks, the Substitution-Permutation Networks (SPN) operate on the entire state in parallel using three distinct layers, the substitution layer (S-box), a non-linear bijection $S : \{0,1\}^m \to \{0,1\}^m$ providing confusion, the permutation layer (P-layer), a linear transformation (bit-shuffling or matrix multiplication) providing diffusion; and the key addition, an XOR operation with the round subkey. Lastly, the Add-Rotate-XOR (ARX) networks rely on the non-linearity of modular addition ($+$) combined with bitwise rotation ($\lll$) and XOR ($\oplus$). These operations are extremely efficient in software but challenging to analyze using linear cryptanalysis.

2.3.3 Isabelle/HOL Proof Assistant

Isabelle/HOL is a generic proof assistant based on Higher-Order Logic [24]. It serves as the formal language used for the declarative implementation of the selected block ciphers. Isabelle follows the Logic for Computable Functions (LCF) approach, where all theorems must be derived from a small, trusted logical kernel, ensuring that if the kernel is correct, no false theorems can be proven. In this work, we utilize inductive predicates to model the reachable states of a cipher; for example, a valid encryption trace is defined as the set of all

states reachable from the plaintext by applying the round function r times. This subsection directly supports our approach in RO4 and RO5, where we introduce the machine learning based methodologies, TIA and CryptoGraph, to automate the formalization and structural security evaluation of ligthweight block ciphers.

CHAPTER THREE

LITERATURE REVIEW

The pursuit of a secure Internet of Things (IoT) network entangles multiple interconnected research domains. This chapter reviews the key bodies of work that form the basis of this dissertation. We move from the socio-technical dimensions of user trust through to the mathematical foundations of formal verification, highlighting how each layer contributes to closing the security assurance gap.

This chapter is structured around the four research objectives (RO) mentioned in Chapter 1, notably the socio-technical trust analysis, examining the gap between user sentiment and policy, the operational resource management, focusing on optimization in constrained edge environments; then the security assessment, evaluating quantitative risk models and deployment feasibility; and lastly, the formal foundations, investigating neuro-symbolic methods for code translation and cryptographic evaluation.

3.1 RO1: Analyzing the Public's Trust on AVs

Researchers have tried to determine end-users' sentiment towards AVs using traditional survey and polling methods on different populations. [44] surveyed several residents in Qatar to understand their perceptions towards AVs in regards to reducing human error in Human-Driven Vehicles. [14] surveyed 391 participants to examine the public's trust and sustainability concerns regarding AVs. [15] surveyed almost 1000 participants for their perceptions and acceptance of AVs. These studies present various views about technology knowledge, the acceptance of AVs, and demographic information. Results showed that participants with higher knowledge of the technology have better perceptions on AV eliminating human error in regards to safety and performance [44]. While [14] showed

that the participants willing to use an AV considered how useful and convenient they are rather than how it technically performs. [44] showed that around 86.7% of the participants would not prefer switching to AVs, while [15] found out that overall, AVs are perceived to be "low risk". However, the risk also varied depending on whether they are a passenger or pedestrian, interacting with the AV. It was perceived to be riskier being the passenger than being a pedestrian around AVs. They also found out that younger adults compared to other age range had a higher acceptance of the technology, and the demographic of respondents were mostly 37.9% men [14].

[16] analyzes the role social media plays in shaping online opinion by researching language bias and trending social events to explain how sentiments on AV can be influenced. For the analysis of trending social events, they found that bursts of overall positive sentiment occur during the announcements of new AV technologies, and bursts of negative sentiment occur when news regarding user concerns surfaces. [17] researched developing a comprehensive Twitter search algorithm and conducting an analysis of Twitter conversations about AV. They developed a supervised model called Constrained Label Learning (CLL) to identify tweets centered around certain events. Their results show that the average sentiment of AV on Twitter is neutral and only negative when there are words describing crashes or any major news events. Furthermore, recent studies have found that 60% of Twitter users use Twitter as a regular source of news, and 40% of Reddit users use Reddit as a source of news [45]. Therefore, as social media platforms offer the opportunity to both obtain information and distribute personal opinions, it would also be worthwhile to analyze them for the user conversations before and after major news events are released.

[46] analyzed online conversations before and after a 2019 Tesla autopilot incident, finding frequent mentions of “blame” and “confusion.” Although post-crash sentiment was largely negative, many tweets remained positive, indicating the need for further study. [47]

conducted a similar before-and-after social media analysis during COVID-19 to understand how AV-related discussions shifted and which demographics, manufacturers, and policymakers should be targeted to encourage adoption. Their dataset was predominantly male (82% Twitter, 86% Reddit). By occupation, engineers and entrepreneurs were more positive about AVs pre-pandemic, while those in entertainment, education, and writing were more negative; after the pandemic, positive sentiment in these latter groups increased, likely due to perceived convenience. Overall, they found age to be the strongest predictor of the public sentiment towards AVs, followed by education, engineering, and entrepreneurship occupations.

[48] analyzed Twitter conversations about AVs and, from 7k tweets, found that overall sentiment was predominantly positive, with frequent positive topics including "excited," "faster," and "cool," and negative topics centering on "liable," "crash," and "difficult." [49] applied an attention-based LSTM model to classify AV-related Twitter sentiment, reporting an approximate 3:1 ratio of positive to negative tweets and demonstrating that deep learning techniques are effective for this classification task.

Although prior studies largely focus on users, key concerns remain around trust, safety, transparency, and privacy. It is therefore crucial to examine how governments and other regulators approach AV development and whether they adequately address these documented user concerns.

3.2 RO2: Security Risk Assessment in IoT networks

Securing an IoT networks require a dual capability: the ability to strategically quantify risk across the network and the ability to tactically deploy feasible countermeasures on constrained devices. This section reviews the state of the art for quantitative risk assessment in IoT networks, with a use case on CAV networks.

The purpose of performing RA in an IoT network is to identify and understand threats, vulnerabilities, and their relationships in the network. It is essential to examine the interdependence between risks because assessing network risks in isolation might limit the awareness of potential vulnerabilities and overlook the complex dependencies between them. For instance, consider an IoT network of three CAVs (AV1, AV2, AV3) and the respective identified vulnerabilities: weak authentication, lack of sensor validation, and unencrypted communication. Evaluating these threats in isolation might neglect the potential constructive or destructive effects the resulting attacks have on each other. Constructive relationships occur when the success of one attack enhances the likelihood and impact of another attack, while the destructive effect involves interference, where the success of one attack hinders another. A constructive relationship can be seen in a scenario where, with AV1 compromised, the attacker would take advantage and exploit the lack of sensor validation to gain unauthorized access and engage in sensor spoofing with AV2. Concurrently, the compromised AV1 could intercept unencrypted communications with other vehicles like AV3.

Bayesian Attack Graphs (BAGs) are used to account for dependencies between attacks. BAGs offer a holistic representation of vulnerabilities and attack paths, accounting for attacks' dependencies and their potential consequences. Using BAGs for Risk Assessment (RA) can pose challenges, requiring a thoughtful approach to overcome potential obstacles.

Early studies have explored Risk Assessment (RA) through Attack Graphs (AGs), showing their effectiveness in revealing complex attack interdependencies within networks. [50] introduced a graph-based method to assess both external and internal threats, identifying high-probability attack paths to help test security measures. Moreover, [51] enhanced the understanding of attack trees during their construction and analysis by providing denotational semantics for them. Additionally, [52] developed a framework for

detecting insider threats by generating customized attack trees for each user, monitoring activities, and raising alarms when actions align with the attack tree towards potential network compromise.

Further studies introduced quantitative approaches for RA using AGs. [53] proposed managing security in large networks by breaking down complex AGs into smaller sub-graphs, enhancing efficiency and scalability in risk analysis. [54] integrated AGs with the CVSS framework [55] to evaluate host and network damage from attacks. [56] enhanced risk assessment by adding countermeasures and consequences to AGs for a more comprehensive and dynamic vulnerability analysis. [57] introduced a model for simulating attacks, using AGs to compute potential attack paths from confirmed security events. [58] leveraged CVSS metrics to prioritize vulnerabilities and attack paths, validated through the Markov model. [59] introduced a Bayesian Network (BN)-based method as a more flexible alternative to the FAIR model for quantitative cybersecurity risk assessment, achieving greater accuracy and adaptability. Additionally, the approach in [60] focused on the use of BN to model uncertainty in enterprise networks, demonstrating its effectiveness in enhancing security analysis for intrusion response by incorporating attack semantics and experimental evidence in uncertainty modeling.

Several risk assessment models have integrated AGs with BNs for enhanced probabilistic analysis. [61] adapted Bayesian Attack Graphs (BAGs) to cover a broader range of threats, including insider threats, and introduced concepts like security control coverage and risk strategy. The approach recommended unique security measures such as IT training and employee satisfaction surveys within the BAG framework. [62] developed a BAG-based RA model to calculate exploit probabilities and assess static security risks, helping in selecting optimal security strategies for small networks. [63] employed Genetic Algorithms to independently explore multiple attack paths as individual

scenarios, enhancing network security by offering a range of solutions for network security assessment. [64] presented a method using BAGs to model probability metrics, the cumulative effects of vulnerabilities in a network. [65] focused on risk levels of critical resources by considering attacker behavior and emerging vulnerabilities through behavior-based AGs and Bayesian techniques. [66] developed a method for efficiently modeling attack paths and assessing vulnerabilities using Bayesian inference, validated by experimental testing. The Hybrid Risk Assessment Model (HRAM) introduced by [67] extends AGs with BNs to evaluate dynamic cybersecurity risks. The HRAM includes Dynamic Risk Correlation Models for analyzing ongoing attacks, and Future Risk Assessment Models for predicting future threats. [68] presented a risk assessment framework for Sensor Cloud networks using BAGs to analyze attacks and estimate their impact on network security parameters. This work mainly influenced this research framework's design, especially the construction of three BAGs focused on Confidentiality, Integrity, and Availability parameters.

Additionally, other studies have used Artificial Intelligence (AI) and Machine Learning (ML) for RA and risk management within IoT networks. [69] examined key weaknesses in the four main Industrial IoT (IIoT) protocols and conducted a vulnerability analysis in IIoT systems, by identifying prevalent threats and evaluating their severity as well as performing intrusion detection with ML algorithms like SVM, KNN, and Naive Bayes. Additionally, [70] proposed a hybrid method that employs deep learning architectures, specifically a multi-connect variational auto-encoder and Probabilistic Bayesian networks, to enhance risk management and attack detection in physical networks, outperforming traditional centralized systems in accuracy and sensitivity. Authors in [71] developed a RA framework that captures IoT component interconnections and uses AI for real-time and interpretable risk estimation. The research examined risk transference strategies like cyber

insurance and demonstrated the model's effectiveness in identifying IoT cybersecurity risks with the BoT-IoT dataset. The study highlights the need for standardized IoT risk assessment approaches given the dynamic nature of cyber threats.

In the field of CAVs, an emerging area where cybersecurity is crucial, many studies have applied AGs and/or BNs to assess security risk. [72] integrated Bayesian Network (BN) modules into real-time decision-making for unmanned aerial vehicles. [73] created a RA framework using CVSS to evaluate attack success likelihood and their potential impact, helping in secure autonomous driving system design. Authors in [74] introduced a forward-thinking probabilistic model based on variables and causal relationships derived from CVSS to represent CAV cyber risks. Similarly, [75] utilized BNs to determine if a vehicle is under attack and assess the nature and extent of the attack. Authors in [76] developed a security model for AVs that quantitatively evaluated threat likelihood and associated risks using defense graphs and BN analysis, demonstrated through a case study on GPS spoofing attacks.

3.3 RO3: Feasibility Analysis of the Deployment of ML-based IDS on CAN bus

Once the risk assessment is performed and the vulnerabilities of a network are identified, there is a need for defensive measures. For In-Vehicle Networks (IVN) for example, to detect intrusion in the Controller Area Network (CAN) bus of a vehicle in an In-Vehicle Network (IVN), ML models [77] [78] have been proposed to identify anomalies in real time. These models can provide real-time CAN bus traffic monitoring to identify anomalies, making it effective at detecting known and zero-day attacks, hence enhancing the security of the vehicle. However, these models require a lot of computational resources for optimal performance, which are not present in an IVN. This shows that though these

models are effective at performing their specified tasks, little consideration is given to their practical deployment feasibility in terms of their resource consumption and cost.

For signature-based Intrusion Detection Systems (IDS), the authors in [79] proposed a specification-based IDS that utilizes hosting a HIDS on each ECU to detect drop, replay, and spoofing attacks. This detection is done by comparing messages on the CAN bus against the defined protocol specification in the HIDS. In [80], the authors proposed a novel windowed hamming distance-based IDS that detects fuzzing, spoofing, and Denial of Service (DoS). This proposed IDS uses sliding window characteristics and hamming distance to extract the sequential flow of data from the CAN bus time series. It is made up of two modules: a calculation of the threshold and an intrusion detection module. In the calculate threshold module, window size similarity in comparison to data in a normal state is used to determine the threshold ranges. Using these ranges, the intrusion detection module identifies the different types of attacks present in the data.

On the other hand, the detection methodology in an anomaly-based IDS can be categorized into statistical methods and Machine Learning (ML) models. In statistical anomaly-based IDS, statistical methods such as density estimation [81], information entropy [82], and message frequency [83] are used to detect anomalies. In [84], a Wavelet-Based Intrusion Detection System (WINDS) was proposed to identify behavior anomalies in CAN bus traffic through analysis of network transmission patterns. Different statistical detection methods (Z-score, Autoregressive Integrated Moving Average (ARIMA), and supervised threshold) were implemented to comparatively analyze the best detection method for intrusion detection using CAN bus messages [85]. The use of statistical methods in anomaly-based IDS has proven to be effective at detecting anomalies based on the features being analyzed and does not require prior knowledge of the attack. Additionally, it is a lightweight approach compared to ML anomaly-based IDS. However,

statistical anomaly-based IDS suffers from difficulty in identifying novel or sophisticated attacks as it relies heavily on historical data, takes a longer time to detect attacks, and is limited to a specific attack based on the feature(s) being considered. Therefore, there is a need to incorporate ML algorithms that can learn from the data and dynamically adapt to new scenarios. In ML anomaly-based IDS on the other hand, ML algorithms such as K-Nearest Neighbors (KNN) [86], ensemble learning [87], and Convolutional Neural Networks (CNN) [88] are used to detect anomalies on the CAN bus. For a multiclass intrusion detection in CAN bus, the authors in [89] implemented a Deep Neural Network (DNN). An unsupervised ML algorithm, Ensemble Hierarchical Agglomerative Clustering (E-HAC), was implemented to detect zero-day attacks using majority votes to detect intrusion [90]. The authors in [91] integrated a novel distance-based algorithm with an unsupervised Kohonen Self-Organizing Map (SOM) network and k-means algorithm to identify malicious messages in the CAN bus. The application of ML algorithms in anomaly-based IDS has led to improved accuracy and precision in detecting attacks in CAN bus, dynamic ability to identify novel or sophisticated attacks, and real-time detection. However, these models require a large amount of data to train the models before they can be implemented, lack transparency in how they work, and can be resource-intensive. Furthermore, these models are deployed in a resource-constrained environment, IVN, which can render them ineffective due to the limited amount of resources required to run these models in an IVN. Therefore, there is a need to evaluate the feasibility of these ML anomaly-based IDS models so they can be deployed in real-world IVNs.

3.4 RO4: Cross-Paradigm Logic Methodologies for Automating Formal Specifications

While RO3 addresses network defense, true security assurance requires that the underlying software is mathematically proven to be correct and secure. This section

examines the existing literature regarding the transition from imperative (e.g., Python, C++) to declarative (e.g., Isabelle/HOL) languages, to improve and automate cross-paradigm translations, and correctness verification.

Most industrial systems are written in imperative languages (e.g., C, Python, Rust) that focus on state manipulation, while formal verification requires declarative specifications that describe program behavior mathematically. Bridging these two paradigms manually is labor-intensive, error-prone, and demands specialized expertise in both systems engineering and formal logic. Large Language Models (LLMs) show great potential for automating translation tasks, especially between similar high-level imperative languages (e.g., C++ to Java). However, they struggle with the cross-paradigm induction required for formal methods. Transitioning from imperative bit manipulation to declarative formalisms like Isabelle/HOL [24] requires understanding the underlying mathematical grammar, a task where current LLMs often fail, as they mostly rely on syntactic mapping. This emphasizes the need for a scalable, learnable bridge between the corpus of imperative implementations and formal proofs that moves beyond pattern matching to hierarchical logic induction.

Existing works have proposed various solutions to bridge the gap between cross-paradigm translations. Recent research have focused on the transition from natural language to formal specifications. The foundational feasibility of using large-scale pre-training to translate mathematical problems into Isabelle/HOL was established by [92], and built upon by [93], which introduced the Draft, Sketch, and Prove (DSP) framework, using an informal draft to guide the formalization process. Additionally, ALPHAGEOMETRY [94] leveraged a natural language model and neuro-symbolic to improve success rates for mathematical theorem proving. While these approaches achieve formal verification, they mainly focus on mathematical theorems and use natural language

as an input source, and might struggle to induce formal specifications from stateful imperative code.

Some studies have focused on the intra-paradigm translation (e.g., Python-to-Java). For example, [95] demonstrated that unsupervised translation on a neural transcompiler could bridge syntactic differences between high-level programming languages. [96] provided a rigorous taxonomy of code translation (token, syntactic, library, and algorithm levels). It argues that current models fail at tasks where deep knowledge of algorithms is required. In the cryptography domain, the verification of cryptographic primitives is the gold standard for system security. Historically, this has required extreme manual effort through frameworks like HACL* [97] and FIAT-CRYPTO [98], which utilize verified compilers to synthesize provably correct C code from high-level mathematical specifications in Coq. Recently, AWS [99] discusses the use of automated provers to verify cryptographic libraries, highlighting the manual effort required to write formal specifications as the primary bottleneck.

Moving beyond syntactic pattern matching, [100] highlighted the use of intermediate scaffolding to improve neural reasoning through Chain-of-Thought. [101] introduced the Tree of Thoughts to handle complex multi-step logical tasks. While these methods provide general reasoning hints, they lack a hierarchical alignment mechanism that maps the specific semantic tiers of an algorithm (e.g., constants vs. state-flow) to the target logic. Our approach fills this gap by decomposing the translation task into a four-tier semantic hierarchy (Lexical, Functional, Structural, Orchestration). This forces the model to align logic at the component level, successfully inducing the semantic grammar required for Isabelle/HOL where end-to-end approaches fail.

3.5 RO5: Automated Security Evaluation of Lightweight Block Ciphers

The final layer of the literature review focuses on existing approaches regarding the early assessment of the security of the lightweight block ciphers. This section focuses on existing approaches for early security assessment

Researchers have proposed various non-standardized lightweight block cipher implementations optimized for resource-constrained IoT environments [23]. However, the proposed ciphers differ in resource costs, leading to tradeoffs among various security properties such as block size, key size, number of rounds, S-box complexity, and software implementations, which can result in inefficiencies [102] [23]. Although standardization efforts have been made to define secure primitives for IoT environments [103], the diversity in cipher implementations (e.g., key schedules), security properties, cost, energy, and hardware compatibility, leads security administrators to use custom, non-standardized ciphers. Thus, there is a need to evaluate ciphers' attack resistance and security level.

Cryptanalysis helps evaluate a cipher's security. However, cryptanalysis can be resource-intensive (cost and time), requires expert knowledge, and can be prone to human error. It may not be feasible for all new cipher designs, especially custom or modified designs, which may slightly differ in round count or operations, potentially compromising the environment's security. Without proper assessment, security administrators could either over- or under-protect IoT environments. Therefore, *an early evaluation method is needed to assess the security of new ciphers and reduce the risks associated with vulnerable cipher designs*. This evaluation can help security administrators understand the strengths and implications of lightweight block ciphers before deployment, and potentially reduce cryptanalysis costs. To enhance the security evaluation of block ciphers, researchers have focused on formal verification and applied ML.

Cryptographic Primitives Formal Verification: In the field of cryptography, formal verification methods employ logical systems to establish mathematical assurance regarding the correctness of software systems by verifying the algorithms' properties during implementations. Many studies have focused on ensuring that implemented cryptographic code is free from side-channel attacks. [104] developed a toolchain that automatically verifies C code implemented for the constant-time property, by translating it into the Isabelle/HOL theorem prover. This research ensures that cipher's execution time does not depend on secret data, and highlights the capabilities of formal tools such as Isabelle in providing guarantees about the correctness of implementations. Similarly, in Project Everest [105], to mitigate timing side-channel attacks, [97] introduced the HACL* (High-Assurance Cryptographic Library) project, using the F* proof assistant to develop a complete cryptographic library that is formally verified for functional correctness, memory safety, and constant-time execution, showing the practicality of creating high-performance, provably secure cryptographic code. Additionally, [106] also developed an approach that combines verified high-level programming, verified compilation, and verified optimizations to deliver both correctness and high performance for cryptographic code. This demonstrates the feasibility of a verified engineering pipeline from high-level specification to optimized assembly. Similarly, [107] used the Coq proof assistant and the Verified Software Toolchain to formally verify the functional correctness of the SHA-256 implementation within OpenSSL, proving that the cryptographic implementation correctly performed its specified algorithm.

Beyond implementation correctness, formal methods can be used to verify abstract cryptographic properties. [108] leveraged Isabelle/HOL to formalize the concept of indifferentiability, an important security notion for hash functions, applied to verify the security of various sponge-based constructions like SHA-3. This shows the ability of

proof assistants in reasoning about the theoretical security of cryptographic designs. Additionally, many modern security proofs on cryptographic properties are probabilistic. [109] introduced a form of probabilistic relational Hoare logic, a mathematical framework used to reason about the behavior of programs that involve randomness, for proving differential privacy in interactive systems. This work provides a foundational approach to formally verifying privacy properties, which can complement security analyses of cryptographic primitives and protocols. In a similar probabilistic setting, [110] developed EasyCrypt, a tool-aided framework for formally verifying cryptographic proofs, in a probabilistic setting, making it easier to construct complex, game-based proofs. While these works lay a strong foundation for the correctness of cryptographic properties and primitives through verification, they do not offer automated mechanisms for quantifying or evaluating the security levels for new or unverified cipher designs.

ML for Cryptanalysis and Cipher Security: To address the aforementioned challenge, some studies have applied ML to improve security in general, and specifically for cryptanalysis and code representation and analysis. [111] introduced a novel approach to differential cryptanalysis through the application of neural networks. By training a neural distinguisher on pairs of plaintext and ciphertext, the model effectively identified a differential characteristic for the lightweight cipher Speck, which could be used to execute a key-recovery attack. This illustrated the capacity of deep learning methodologies to uncover complex statistical patterns that were traditionally found through manual analysis. A more rigorous analysis of [111]'s approach was later done in [112], exploring a wider range of neural network architectures. They confirmed the validity of the technique on multiple ciphers and offered deeper insights into why and how these models work. They concluded that these networks mainly learn linear approximations of a cipher—combining linear cryptanalysis with differential properties—rather than new non-linear techniques.

Despite their power, such neural cryptanalysis methods remain black boxes that model only external behavior, not internal structure, and thus lack interpretability and verifiable foundations.

Regarding cipher security evaluation, [113] demonstrated the effectiveness of deep learning for profiling side-channel attacks against the Speck cipher. They employed a sequential divide and conquer ensemble of deep learning models to attack software implementations of the SPECK-32/64 cipher, successfully recovering its 8-byte secret key in fewer than 250 traces. Additionally, [114] proposed a method for evaluating the security of lightweight block ciphers using ML classifiers trained on features like the properties of their number of active S-boxes, the number of rounds, and the permutation pattern. The results showed that non-linear ML models outperform linear models, achieving a prediction accuracy of up to 93% on data seen during training and up to 71% on unseen ciphers. This work shares our high-level goal of automated security evaluation, but it relies on a predefined, hand-crafted set of features rather than learning from the complete structural representation of the cipher.

Some approaches have focused on using ML models to understand the syntax and semantic code itself. [115] established the principle of learning code from their rich structural information within an Abstract Syntax Tree (AST), rather than a flat sequence of text. The approach aimed to represent a code snippet as a single fixed-length code vector, that are then decomposed to a collection of paths in its ASTs, and used to train a neural network model. This provides a foundation for using ASTs to represent the structural information of the dataset in this work. Additionally, [116] demonstrated that a GNN trained on Simplified Code Property Graphs, representing syntactic and semantic information of software functions, could effectively detect vulnerabilities in C/C++ source code. Furthermore, [117] firstly applied GNNs to the domain of higher-order theorem

proving. Firstly, they converted mathematical theorems present in HOL light theorem prover into graph representations, and then successfully trained a GNN to predict the next logical step in a proof search, significantly outperforming previous methods. This work established that GNNs can extract meaningful, semantic information from the ASTs of formal logic. While focused on proof guidance, this work inspired our methodology of using GNNs on ASTs derived from formal models for security classification. These works show that ML, in particular GNNs, can learn the representations of code and formal logic. However, they neither use formal verification techniques nor predict a system's properties, such as security level. Our proposed framework addresses the gap between formal verification techniques and ML for block ciphers' security evaluation. By using Isabelle/HOL to create verified cryptographic representations, it extracts ciphers' structured representations to train an ML model for security assessment.

CHAPTER FOUR

RO1: ANALYZING THE ALIGNMENT BETWEEN PUBLIC CONCERNS AND EXISTING POLICY ON AUTONOMOUS VEHICLES

The transition to fully Autonomous Vehicles (AVs) marks a major shift in transportation, not only advancing technology but also reshaping how society engages with intelligent systems [8]. Although the technological roadmap for AVs is clear, adoption is hindered by user alienation and resulting deficits in public trust. This chapter argues that trust depends not only on engineering reliability but also on alignment among stakeholders. For AVs to be successfully integrated into the supply chain and urban infrastructure [9, 10], the development process must be transparent and inclusive of all key actors: manufacturers, government bodies, and the public. A harmonious ecosystem requires that the laws governing AVs directly address the users' concerns.

This chapter addresses Research Objective 1 (RO1) through a comprehensive, non-event-driven comparative study of unstructured user data from social media (Twitter and Reddit) and formal government policy documents from U.S. and international bodies [25]. We quantify the trust gap, the structural mismatch between users' concerns and what government regulations on AVs. Specifically, the contributions of this work are: (i) conducting non-event-driven sentiment analysis to isolate persistent safety and privacy concerns about fully autonomous vehicles, (ii) extracting policy focus from publicly available documents (e.g., U.S. DOT, European Commission) to identify current regulatory priorities; and (iii) comparing these two perspectives to locate points of convergence and divergence, thereby outlining a roadmap for future legislation better aligned with public needs and concerns.

4.1 Data Acquisition and Pre-processing

For user concern analysis, data is acquired from social media platforms Twitter and Reddit owing to their suitability to gauge public sentiment and their ability to provide timely feedback concerning various events. Aside from analyzing social media for public opinion, this study analyzes government policy documents made publicly available on the Web by the United States of America and foreign entities like the European Union to regulate autonomous vehicles.

4.1.1 Social Media Scraping

Data was scraped from Twitter and Reddit and saved as `.CSV` files to be analyzed and modeled for user concerns regarding AVs. Separate modules for Twitter and Reddit were utilized to scrape the platforms by keyword. The scraping module Python Reddit API Wrapper (PRAW) along with a developer account was used for Reddit and the Twitter Intelligent Tool (twint) was used to scrape Twitter. Before using the scraping tools, a list of common AV-related words was created, derived from journal articles, papers, and policy papers available in the literature and on the Web. These keywords consisted of a combination of phrases like autonomous vehicle, driverless car, and self-driving car with several related topics as shown in Fig. 4.1. It should be noted that there exist other keywords that are synonyms to "autonomous vehicles" or related topics of AV that were not included in the scraping process. However, the three selected terms quantify a well representative sample with 6,497 tweets and 13,350 Reddit comments.

4.1.2 Government Policies Scraping

To obtain government regulations concerning AVs, `.PDF` files of several downloaded government papers were scraped with PyPDF2. From U.S.A, 17 papers from the National

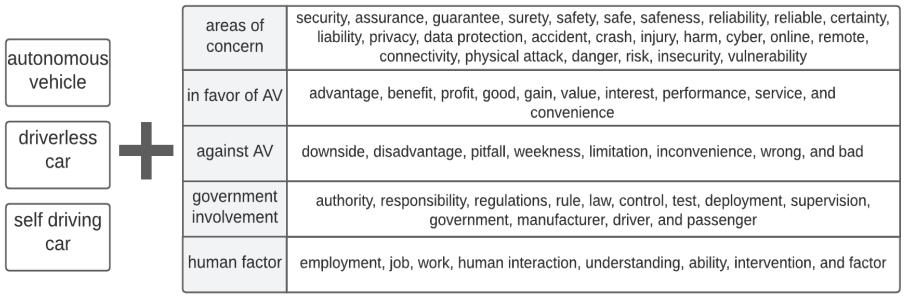

Figure 4.1: List of combinations of keywords used for the social media scraping

Department of Transportation, the National Highway Traffic and Safety Administration, and various state governments [45] were analyzed and saved as `.CSV` files. This process was then repeated with the papers from foreign entities including Australia, Canada, Dubai, the EU, France, Germany, India, Japan, New Zealand, and Singapore, taken from various sources, yielding a total of 25 papers. Policy papers from China and Russia were also looked into, but their regulation papers could not be used either because of a language barrier or because they were American articles written about them in which bias could exist, and therefore was excluded from the process. The foreign policies that were utilized can be found in [45].

4.1.3 Data Pre-processing

Natural Language Processing techniques were used to clean and prepare the Reddit comments and Tweets by removing duplicates, blank spaces, special characters, numbers, URLs, and Twitter handles and transforming into lowercase. Thereafter lemmatization process was applied to change the words to their base forms. Stopwords (*'and'*, *'the'*) are regular common words that usually do not add meaningful information to text and were removed for text analysis. For sentiment analysis, a separate dictionary was created to remove only minimal words like articles and conjunctions. This allowed us to keep

negation, and modal verbs and provide more context for the analysis in contrast to using a default dictionary. For all other data analysis methods we used the English stopwords dictionary from the `nltk` library. After the cleaning process, the dataset decreased to 4,642 tweets and 13,337 Reddit comments. For the government policy papers, the aforementioned preprocessing techniques were applied. Moreover, since most PDFs had extraneous data like *summary*, *appendixes*, *title* on every page, those information were manually removed. Country names were also removed from the dataset, to prevent them from affecting the analysis.

4.1.4 Data Categorization

To obtain a more comprehensive understanding of the data, both the user concerns and government policy datasets were categorized. For the users' concerns, the categories were classified using the keywords proposed to scrape the data (Fig. 4.1) compiled through the literature survey of researchers like [13], [14], [15], where it was found that users have a different kind of concerns related to *trust*, *safety*, *control of the vehicle*. For government policies, a preliminary analysis of a few government policy documents helped in understanding their focus on issues like liability, accident responsibility, and vehicle control. These categories were then compared to the user concern categories, to match similarities and categorize both datasets along similar categories as shown in Fig. 4.2.

Following the preliminary analysis, a Python script was written to iterate through the datasets and search for keywords related to different categories, then save the data into respective .CSV files. For example, in accident category, the synonyms and related topics such as crash, fatality, death, and collision were used to collect tweets, comments, or policies that mention these areas of concern. This was done to analyze each category individually and to get a complete understanding of the public's perception and policies for

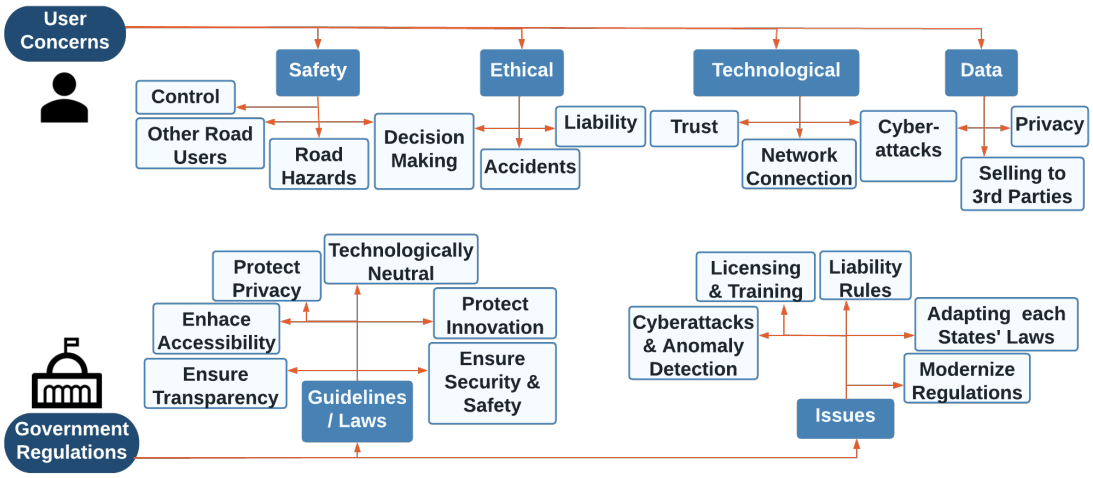

Figure 4.2: Categorized user concerns and government policies

each category. To have a similarity between the users' concerns and government policies categories, government categories and the keywords related to them were used to infer the same categories as for the users' concerns. For example, *transparency* would be useful for deducting *trust* and overlap with *safety* categories.

4.2 Data Analysis: Methods and Results

After data pre-processing, several data analysis methods were applied to analyze datasets for insights. The outlines of the methods applied and their results are reported in the following sections.

4.2.1 Tri-gram and Word Frequency Analysis

4.2.1.1 Method: Both the user concerns and government policies datasets were analyzed using tri-grams. To prevent synonyms of *autonomous vehicles* overpopulating the most frequent words list, they were removed from the datasets. Besides, the word frequency analysis was applied to different categories in Fig. 4.2, to have a better understanding of the existing keywords in both datasets.

Results on User Concerns: Fig. 4.3 shows results from the tri-gram analysis. Analyzing the two platform's trigrams, Twitter conversations are more focused on the area of crashes and accidents compared to Reddit. Twitter has four phrases in the top 10 that mentions crashes or accidents, Reddit has only one. Further, the number one most frequently appearing trigram for Reddit is *safer human driver*, which overall has positive connotations without further context.

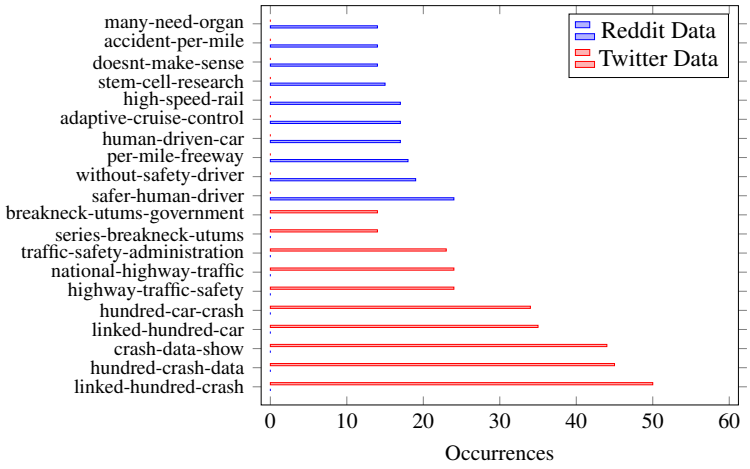

Figure 4.3: Top 10 Trigrams for Reddit and Twitter user concerns

Thereafter, word frequency analysis and percent proportion computation was performed on each of the user concern categories. Table 4.1 shows that *accidents* and *road hazards/users* are the most commonly mentioned topics. Whereas, *cyberattacks/cybersecurity* are the least mentioned topics. However, this can be rationalized as an average user currently do not consider cybersecurity as an issue for AVs. However, it is an imperative area of concern since connected and AVs can increase the attack surface for rogue actors.

Table 4.1: Word frequency for each user concern category

Category	Reddit Frequency (13.3k)	Twitter Frequency (4.6k)
Control | Accidents	400 (3%) | 900 (6.8%)	132 (2.8%) | 333 (7.2%)
Road hazards/users	1000 (7.5%)	233 (5.1%)
Decision making	200 (1.5%)	37 (0.8%)
Liability | Trust	200 (1.5%) | 150 (1.1%)	254 (5.5%) | 62 (1.3%)
Network connection	100 (0.75%)	47 (1%)
cybersecurity | Privacy	2 (0.01%) | 150 (1.1%)	48 (1%) | 102 (2.2%)
Sell to 3rd parties	70 (0.5%)	49 (1%)

Results on Government Policies: Fig. 4.4 shows the trigrams with words like *federal safety standard* and *road traffic law* frequently mentioned, revealing how government policy across nations is focusing on implementing safety laws and standards on AVs.

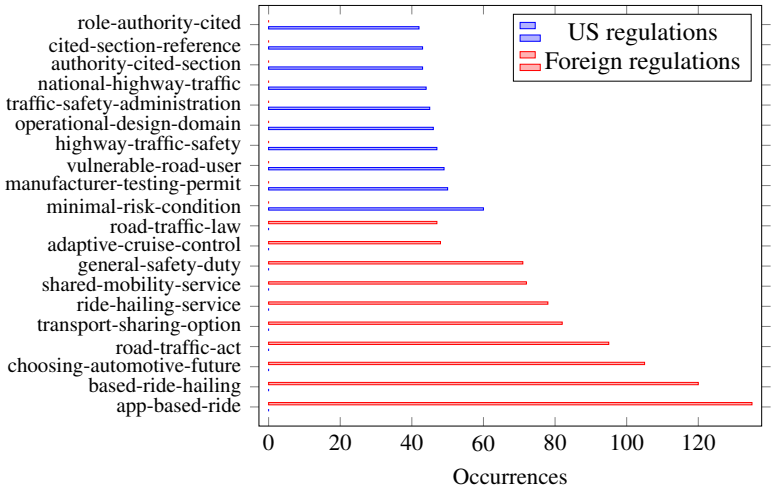

Figure 4.4: Top 10 Trigrams for U.S. and foreign government policies

A word frequency analysis was also performed on the government policy categories (Table 4.2). Percent proportions of each category were unable to be obtained for regulation papers, given that they were scraped by page, one page containing various information. In Table 4.2, similar to user concerns, *safety* and *security* are two of the most mentioned topics. *Liability* is also a very frequently mentioned topic in government regulations. However, when analyzing the liability regulations from the papers, there

were disagreements regarding the responsibility for accidents or data access. Although the topics of *cybersecurity/cyberattack* are mentioned more in government regulations than user concerns, there is still a very lacking amount of regulations especially on the U.S. side with only 20 mentions in all U.S. regulations, requiring a need for more cybersecurity regulations.

Table 4.2: Word frequency (W.F.) for each government policy categories

Category	U.S W.F.	Foreign W.F.	Category	U.S W.F.	Foreign W.F.
Transparency	25	43	Security	400	646
Privacy	120	387	Innovation	50	177
Accessibility	200	62	Safety	1500	1518
Liability	170	619	Tech. neutral	400	1
Cyberattack	10	29	Cybersecurity	10	192
Anomaly detection	10	2	Licensing	200	154
Laws	30	322	Modern regulations	0	14

4.2.2 Sentiment Analysis

4.2.2.1 Method: For sentiment analysis we applied the Bidirectional Encoder Representations from Transformers (BERT) model, a machine learning model utilized for NLP tasks. BERT was trained on an AV-related labeled dataset [45], to better learn to identify different sentiments when it comes to our dataset.

BERT model training data and process: First, a labeled dataset with people's opinions on AV was used. This dataset contains over 7,000 labeled tweets and 10 columns among which *text* and *sentiment* were extracted to train the BERT model. The *text* column represents the tweet and the *sentiment* is a rating given from 1 to 5, the most negative to the most positive, respectively. The sentiments were divided into three main categories: negative (from 1 to 2), positive (from 4 to 5), and neutral (3). As shown in Fig. 4.5, the dataset was highly unbalanced, with a predominance of neutral sentiments. This led to the

model classifying most of the data as neutral sentiment. Data augmentation with the help of the `nplaug` library was applied to identify an additional 2,500 positive tweets and 3,000 negative tweets.

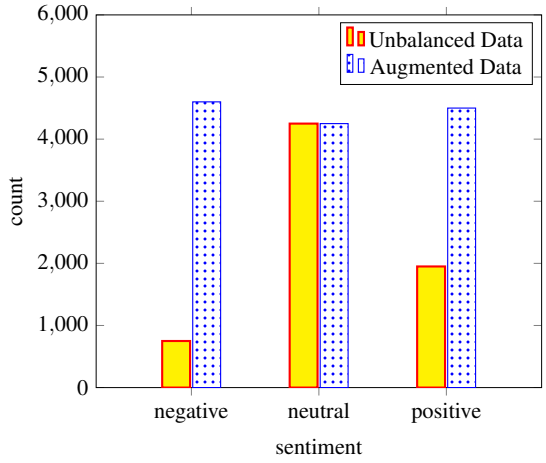

Figure 4.5: BERT training dataset before and after data augmentation

To train the BERT model, 90% of the augmented dataset was used. For the test and validation sets, 50% of the original non-augmented dataset was used with the *train_test_split* function from `sklearn` library to split the different sets. Moreover, the BERT model's parameters were fine-tuned to obtain the best possible accuracy. The final accuracy of the model on the test set is 96%.

Results and Discussion - User Concerns: Compared to other already pre-trained models, our BERT model performed better. This is attributed to the fact that BERT performs well in identifying context and was pre-trained on a labeled dataset related to AVs. On the user concerns, the overall tweets and comments were classified as negative with BERT. Moreover, the previously categorized data were each run through the BERT model and the outcome was predominately negative sentiment across all categories. From

the results, it is also apparent that Reddit had exceedingly more negative sentiments than Twitter (Fig. 4.6). This could be explained by the fact that Reddit users are all anonymous and therefore may express themselves more freely.

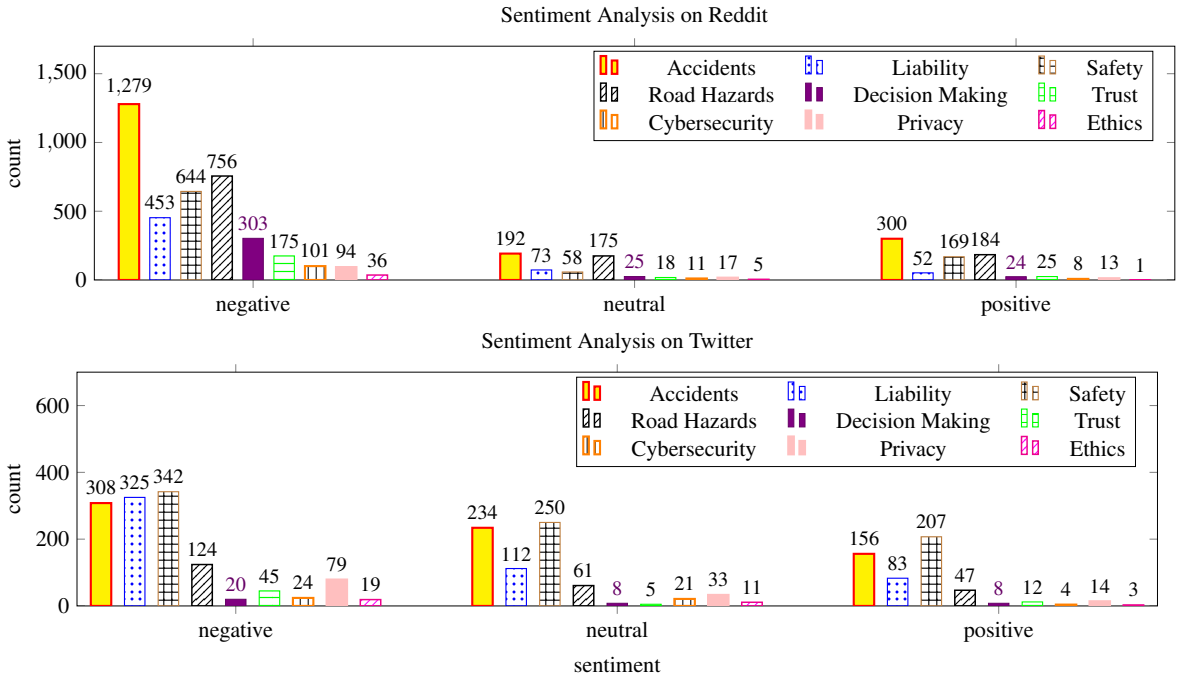

Figure 4.6: BERT sentiment analysis on user concern categories on Reddit and Twitter

Further, certain categories such as accident or safety have significantly higher numbers of positive sentiment on Twitter compared to Reddit. This could be due to the many promises AV brings towards safer and more efficient transportation. An example tweet with positive sentiment states, *'it doesn't get tired at night, it doesn't get drunk, it doesn't make those kinds of mistakes that lead to these terrible outcomes.'* A major aspect of positive sentiment towards the AV is based on the fact that AV will decrease the number of human error accidents.

4.2.3 Topic Modeling

Method: Topic modeling is an unsupervised machine learning technique capable of clustering word groups and similar expressions in documents and offering insights to better understand large collections of unstructured text. Topic modeling was applied to two specific categories: accidents and privacy. Those categories were chosen given the different engagements of the users in the data categorization and sentiment analysis sections, high and low engagement respectively.

Data Pre-processing: Using the previously processed dataset, all *autonomous vehicle* synonyms like *driverless car* or *self driving car* were replaced with *autonomous vehicle*. Then, *gensim's* phrases class was applied to group frequent related phrases into one token. As such, *traffic safety* will become *traffic_safety*. Further pre-processing was done to remove the most common and rarest words in the corpus for the Latent Dirichlet Allocation (LDA) model.

Model Used: The latent Dirichlet allocation (LDA) from the gensim library was applied, given its good performance in clustering the text by similar domains. To find the optimal number of topics for the LDA model in each category, the coherence scores for different topic numbers were considered, as well as applying the Hierarchical Dirichlet Process (HDP) class. HDP is similar to LDA, except it seeks to learn the correct number of topics from the dataset. While relying on coherence scores might not always be satisfactory given the context and the dataset, applying the HDP method would constantly give 20 as the optimal number of topics no matter the data source (Twitter, Reddit, or government policies), the category (accident, safety, or decision making, and so forth), or the number of instances (from 100 to 1000 tweets/comments). Further, it was noted that some topics had only one or two instances. Moreover, 20 would be the result for many other projects

using HDP with different and diverse datasets. Consequently, the coherence score was chosen as the model evaluation metric.

Results of User Concerns Analysis: For each category, the coherence scores of the LDA models were calculated with the number of topics starting from seven to thirty with a step of three. The number of optimal topics for the accident and privacy category is seven for Twitter and Reddit.

In Table 4.3, the primary focus on Twitter regarding accidents is strongly related to companies like Tesla and Waymo. While on Reddit, people are more concerned about crashes and pedestrians' lives. Regarding the privacy, the topics from Twitter concerns data privacy with terms like *datum*, *protection*, *concern*, *law*, *private* in the first two topics. While on Reddit, the first topic seems to translate the work that should still be done for people's privacy, with terms like: *people*, *privacy*, *company*, *work*. To better understand what the topics were about, samples of sentences that most represent given topics were extracted from the dataset. For example, for topic 2 in the accident category for Twitter (Table 4.3), the most representative sentence was raising concern about tesla advising drivers to stay aware of surroundings while driving a *much safer* autonomous vehicle. That sentence belongs with 93% to Topic 2. Another example of privacy for Reddit talks about the protection and requirement that people should have and follow while testing AVs. For example, testers should be an employee of the testing company and have specific safety training and coverage. That sentence belonged at 96% to topic 2.

Results of Government Policies Analysis: For each category, the optimal number of topics with the highest coherence scores was ten for the U.S. policies for both the privacy and accident categories, whereas it was ten for the foreign policies in the accident category and seven for the privacy category. For the top topics of the government policies regarding the accident category, the U.S. focuses on the performance of autonomous systems along

with guidance and law enforcement. One can also infer from Topic 1 in the accident category (Table. 4.3) that the foreign countries focus more on developing infrastructure for the future of AVs, as well as developing regulations.

Table 4.3: Top 2 topics of user concerns and government policies in accident and privacy categories

Category	**Twitter**	**Reddit**	**U.S. Regulation**	**Foreign Regulation**
Accident Topic 1	company, link, law, New York, responsible, liable, happen, course, might, technology	fault, avoid, hit, blame, pedestrian, crash, brake, system, Uber, tech	automated drive, DOT, transportation, performance, support, safety administration, national highway, operation, guidance	moral, CAVs, pedestrian, infrastructure, communication, future, gap, CAV, obtain, different
Accident Topic 2	accident, Tesla, human, cause, driver, one, safe, use, even, drive	autopilot, crash, insurance, risk, company, test, datum, safe, like	scenario, real, section, define, automated drive, world, element, lead	framework, map, regulatory, panel, policy, begin, paper, February, industry
Privacy Topic 1	full, automated, Tesla, vehicle, driver, right, datum, private, think, drive	privacy, people, say, company, want, mean, really, work, way, know	automated drive, section, act, drive, bill, follow, relevant, occupant, practice, traffic	transport, time, travel, cost, share, Uber, vehicle share, report, people, mobility
Privacy Topic 2	concern, need, automate, protection, think, datum, vehicle, private, full, law	beta, public, road, use, automate, mean, test, Tesla, like, driver	program, Michigan, management, impact, transportation system, mobility, investment, connect, infrastructure, private	CAVs, user concern, respondent, personal datum, recommendation, right, individual, develop

4.2.4 Comparative Analysis: User Concerns VS Government Policies

In this subsection, this research reports the comparative analysis results in the accident and privacy categories for user concerns and U.S. government policy sets. Emphasis is placed on the U.S. policies only to better identify any existing gaps and raise awareness of the U.S. government about them as well as the concerns the public might have; This will be done by identifying the similarities and divergences between the two stakeholders. This analysis helps in understanding what may be preventing the total acceptance of AVs, thus providing indications on what should be done.

Accident: Analyzing the top 3 topics in the accident category, it was noted that users' concerns are evolving around various themes. One such frequent theme is about people being afraid of crashes regarding AVs. Although understanding AVs would reduce the car accident rate, it will not nullify the concern. This leads to other concerns like having the driver of an AV stay aware of the surroundings while the autonomous system is driving or the ability for AVs to adapt and learn to not cause problems on different and unknown road conditions. Moreover, another important concern includes the population wondering if non-self-driving cars would one day be eradicated for autonomous ones.

In the U.S. government policies, different themes emerge from the first few topics. The advantages of AVs especially in reducing the number of car accident death rates are brought forward as a big reason for implementing them. Besides, the policies also emphasize the development of safety design guidelines. The requirement to conduct tests are discussed, as well as the risk that might exist in AVs and how to quantify it. It was also noted from the top two topics on user concerns and government policies sides certain similarities like both parties agreeing on the benefits of AVs, especially reducing the accident rate. Users also raise various concerns like the reason why a driver should be aware of the surroundings while the autonomous system drives. Additionally, government policies do not always exhibit specific regulations or explanations on existing problems, like just emphasizing the fact that safety regulations must be developed.

Privacy: In the sample comments on social media in the two first topics regarding the privacy category, different concerns are emerging. Firstly, people are worried about privacy in general, and especially bio-metric violations in cars, after existing incidents with tesla. Furthermore, another common concern is related to the fact that people would like AVs to meet safety, and privacy real metrics, and others are talking about developing frameworks to evaluate implications for privacy and security in AVs.

Regarding the U.S. government policies in the privacy category, there is an emphasis on developing technical requirements for AVs connectivity. Other privacy matters are raised, like cyber criminality and the need to protect personal information. In addition, the need to increase public awareness of regulations is also mentioned, as well as the development of technical standards regarding data liability. While there exist common topics in both the users and government policies, like protecting personal information, the government also talks about increasing public awareness about regulations, which can be considered very important. The more the public is aware of regulations, the more likely it is to understand and accept the technology.

4.3 Future Directions

This work will be extended to use the data analysis as a base to develop an explainable AI (XAI) framework, thus increasing users' trust in adopting fully AVs. The XAI will aim to understand and answer to the best of its performance all end-user queries. The design will be supplemented by an interface with the following features. (1) Ask a question: user can express a concern and interface returns a mapping with the government policies mitigating the concern. (2) Visualize public sentiments towards AVs categories and their evolution on a given time period. To improve efficiency of learning and explanations, the framework through the interface will enable users to give feedback. Information about demographics, profession, education-level will be collected. The end-users will be able to give their assessment about the XAI framework and its interface on whether it helped them better understand the technology, change their perception of AVs positively or negatively and their overall experience and suggestions.

4.4 Conclusion

This paper discussed the analysis of various users' concerns regarding the skepticism of adopting fully autonomous vehicles by using extracted data from two social media platforms: Reddit and Twitter. It also discussed the analysis of current government policies that apply to AVs. While existing literature found that there is generally higher acceptance and more positive sentiment towards autonomous vehicles, the findings of our analysis differ. Across the board for user concerns on Reddit and Twitter, negative sentiment was predominately found.

Furthermore, the complete implementation and acceptance of AVs cannot be immediate, since there are still many uncertainties and questions on the user's side. The existence of an XAI framework to provide more information and answers to future AV users might become a key factor in the development and acceptance of fully AVs. It will be an opportunity to inform future users on the different aspects, advantages, and possible dangers of AVs as well, like cybersecurity, lacking in both the users' concerns and the government regulations.

CHAPTER FIVE

RO2: SECURITY RISK ASSESSMENT IN IOT NETWORKS, WITH BAYESIAN ATTACK GRAPHS AND COMPLEX PROBABILITIES

The consistent growth of the Internet of Things (IoT) is creating a vast, interdependent attack surface in critical domains such as Connected and Autonomous Vehicles (CAVs). Ensuring reliability requires not only robust security, but also optimized resources usage. This chapter addresses the challenge of mathematically modeling how attacks propagate across interconnected devices; it addresses Research Objective 2 (RO2), by bridging theoretical risk quantification and practical defense deployment in IoT networks. We introduce complex-valued Bayesian Attack Graphs that use complex numbers to model constructive and destructive interference among attacks, naturally handle cycles and attacker backtracking without modifying the network architecture, and validate the approach on a CAV use case that maps logical dependencies and quantifies multi-stage attack impacts [26].

5.1 Approach

This section outlines the architecture of an IoT-based Connected and Autonomous Vehicles (CAV) network to establish a threat model and assess its effectiveness. The framework addresses common vulnerabilities across various IoT domains, making it applicable to any IoT network. The following sections discuss the threat model and technical details of the proposed approach.

5.1.1 Network Architecture

This work considers an IoT network architecture, as described in Fig. 5.1, highlighting the four exploitable layers of a CAV network. This includes, V (Vehicle and its

sub-systems): communications among components inside an AV via protocols like FlexRay/Ethernet. V2V (Vehicle to Vehicle): communications from a vehicle to another via V2V protocols like DSRC/ITS-G5. V2I (Vehicle to Infrastructure): Vehicle to Roadside Units (RSU) like traffic lights through protocols like C-V2X, 5G New Radio. V2X (Vehicle to Everything): Vehicle to other devices connected via the Internet but not explicitly part of the CAV network like phones and smart grids. Details on these exploitations and assumed attacker capabilities are discussed in the next section.

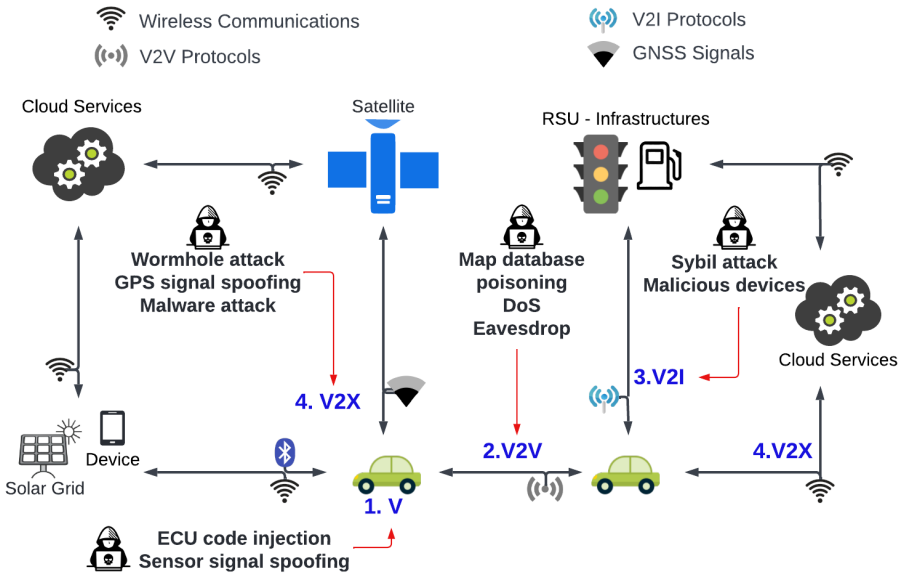

Figure 5.1: Large IoT network architecture use case scenario: CAV network

5.1.2 Threat Model

The threat model references an architecture composed of resource-constrained devices like ECUs, LiDAR, and Radar sensors, communicating through local (in-vehicle) and internet-enabled network layers (V2V, V2I, and V2X). The primary *threat agents* include

remote attackers, internal adversaries, or well-resourced individuals, with basic to high capabilities, exploiting known vulnerabilities (CVE/NVD database), weak authentication mechanisms, or physically compromising in-vehicle components like ECUs. The *attackers' motivations* entail privacy invasion, service disruption, full system compromise, financial gain, or espionage. The *attack methods* exploit attack surfaces such as open ports, exposed APIs, wireless channels, or interconnected devices' firmware and hardware (CAN bus or RSUs). These attack methods aim to compromise core security goals, like the network's Confidentiality, Integrity, and Availability. The *attack scenarios* outlined in this threat model are summarized in Table I, detailing the target security parameters and network components for compromise. The threat model aims to outline a comprehensive set of potential attacks in such networks and their interactions, focusing on both constructive and destructive behaviors. It operates under the assumption that any identified vulnerability is persistent, meaning once discovered, a security administrator cannot quickly or permanently eradicate a vulnerability. This approach encourages a long-term view in risk assessment, prompting security administrators to consider all valid attack paths and the implications of security threats beyond just patching vulnerabilities, especially when an attacker backtracks to a previously compromised state, to stay undetected or explore alternative attack paths.

5.1.3 Proposed Risk Assessment Framework

The proposed framework is based on Attack Graphs (AGs), to illustrate ways an attacker can compromise a network. They can be represented as nodes-as-attacks [19] or nodes-as-states [18]. In the nodes-as-attacks AGs, the nodes depict attacks, with edges indicating dependencies i.e. for an attack to succeed, it requires another attack to succeed. The root node is the goal state, which in the framework is the violation of the confidentiality

(C), integrity (I), and availability (A) of the network. However, in the nodes-as-states AGs, the nodes portray the system's state in the presence of an attack, and the edges denote state transition or exploits. The goal state may indicate achieving a specific network privilege like an attacker gaining root access to a system. Both representations have their advantages. Nodes-as-attacks are fixed-sized AGs and easier to understand but lack the ability to model system characteristics or network size. Nodes-as-states show the system's state, but face scalability issues in large networks.

In this research, the RA framework leverages BAGs by combining the nodes-as-attacks AGs with BN. BAGs introduce a probabilistic dimension to AGs, allowing the modeling of uncertainties and computing the likelihood of various attack paths [118]. They provide a more realistic representation of evolving threat, considering uncertainties, and enabling dynamic updates [119], therefore, enhancing the accuracy and adaptability of security analyses. To assess attacks' likelihood and estimate the net threat and impact levels, the RA framework adopted the scoring system and security metrics from NIST's RA guide [120] and the Common Vulnerability Scoring System (CVSS) v3.1 Calculator [42]. These resources provide the necessary background knowledge to create attack metrics and model the threat level. [120] explains the purpose, goals, and steps of RA, and [42] provides a way to quantify the severity in software vulnerabilities and can be customized by organizations and security administrators. These resources were adapted to determine the suitable CVSS metrics for the network architecture defined in Section 5.1.1. Fig. 5.2 shows the conceptual design of the proposed RA framework and implementation details in the following sections.

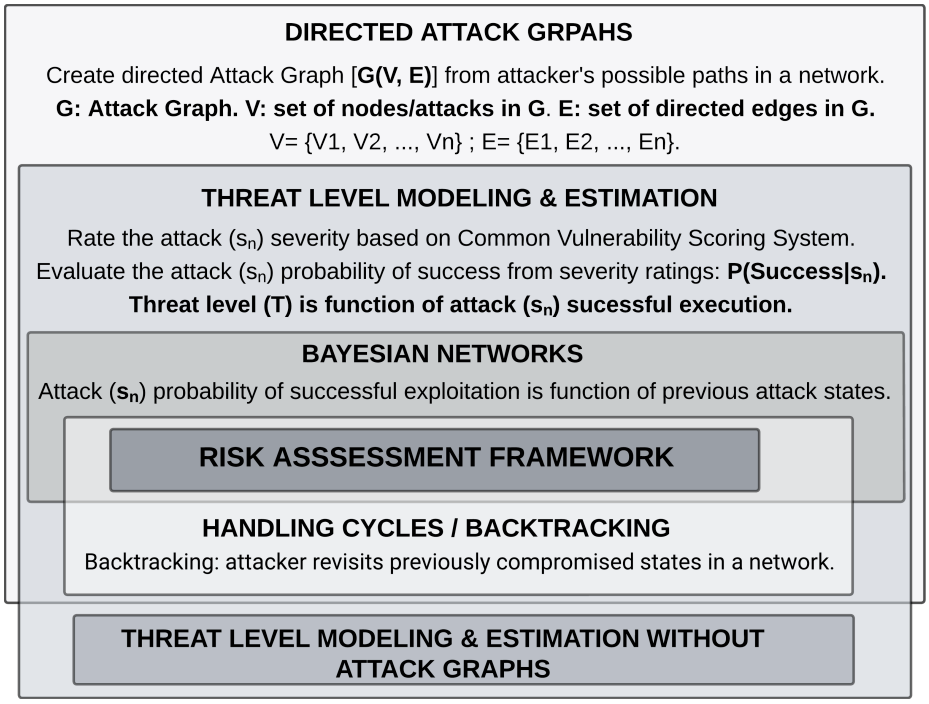

Figure 5.2: Conceptual overview of proposed risk assessment framework

5.1.4 Attack Graphs

Defining Attack patterns is the first step towards developing Attack Graphs (AGs). They describe a sequence of actions that attackers follow to achieve their objectives and give insight into their goals and the security parameter they intend to exploit i.e. Confidentiality, Integrity, or Availability (C, I, A). Attack patterns can be defined as follows.

Definition 1: Attack Pattern - *Attack pattern: a tuple* $P_n = (s_n, \phi)$ *where* s_n *is the attack* [$s_n = 1$ *for successful attacks,* $s_n = 0$ *for unsuccessful attacks] and* $\phi \in$ *[Confidentiality (C), Integrity (I), Availability (A)] is a security parameter.*

Table 5.1 lists potential attacks in a network of connected devices, their description, and associated attack patterns. Due to the dynamic and evolving nature of cybersecurity threats in IoT ecosystems, the list is considered non-exhaustive.

To carry out a specific attack, certain conditions must be met beforehand, also known as attack pre-conditions. Once an attack is successful, the specific outcomes are known as attack post-conditions. Additionally, when the outcomes of one attack meet the conditions for another, an edge is created in the attack graph, indicating a cause-consequence relationship. Moreover, the post-conditions of many attacks can contribute to the pre-conditions of another attack.

Definition 2: Join type (OR, AND) - *Attacks X, Y, Z have an AND join type if their combined outcomes are required to meet the pre-conditions of an attack s_n. Conversely, they have an OR join type if any one of their outcomes can satisfy the pre-conditions of attack s_n. These relationships are documented in an attack module, essential for building AGs.*

Definition 3: Attack Module - *An attack module is defined as a tuple (P_n, S_{pre}, S_{post}, ε), where P_n is the attack pattern, S_{pre} the required pre-conditions for the attack execution, S_{post}, the post-conditions after the successful attack, and ε the join type on the S_{pre}, $\varepsilon \in$ [{OR}, {AND}, {OR, AND}].* Table 5.2 shows attack modules with the causes-consequence relationships of attacks, the pre/post conditions, and the corresponding join types. Attack modules help implement AGs for each security parameter, showing how attackers can compromise the root nodes (C, I, A) and the intermediate nodes an attacker can navigate during their attack. Once an attack state is exploited, the threat model assumes it cannot be undone, and the attacker can visit previously compromised points. This phenomenon, expressed as backtracking is discussed in Section 5.2.4. With the attack modules identified, an AG is defined as follows.

Definition 4: Attack Graph (AG) - *AG is defined as a tuple (s_{root}, S, α, ω), with s_{root}, the ultimate goal of the attacker (a security parameter: C, I, A). S is the set of all possible attacks to reach the goal. α denotes the set of pre and post-conditions of all attacks in S*

Table 5.1: Attacks Definitions and Patterns

Attack	Description	Sec. Parameter	Target component/layer
Node subversion	Adversary taking over a genuine node	C, I	ECUs, RSU, cloud server
Signal jam- ming & DoS	Emits interference signals to disrupt communication	A	GPS receiver, sensors: LiDAR
Packet sniffing eavesdropping	Listens and logs data over a communication channel	C	CAN frames, V2V,V2I, V2X
Sybil attack	A malicious vehicle or node sends deceptive messages with fake identities to others	C, I	RSUs/cloud authentication protocols
Masquerading attack	A malicious node forges identity to access the system	C, I	V2V, V2I identity verification
Device/signal spoofing	Malicious device/signal copy real one, misleading target	C, I	GPS receiver, sensors: LiDAR
Data injection	Code/Data injected to network to compromise the system	C, I	CAN frames, V2V,V2I, V2X
Data modification	Software updated or modified to compromise system	C, I	CAN frames, V2V,V2I, V2X
Altering attack	Modify & send invalid data	I	ECU firmware
Replay attack	Repetitive packet re-routing	I	Telematic units
Wormhole; Blackhole	Deceptive routes diverting traffic through the adversary, enabling free packet dropping	C, I, A	V2V, V2I, V2X routing paths
Sinkhole/Selecti- ve forwarding	Selectively dropping packets after adversary rerouted traffic	C, I, A	V2V/V2I routing paths
Spamming attack	Sending harmful messages to a broad audience	C, I, A	communication layers: V2V, V2I
Flooding attack	Disrupting communication by saturating nodes with malicious signals	A	CAN, ECUs, V2V, V2I communication
Physical attack	Damaging network component through physical means	C, I, A	RSU, ECU hardware, OBD port
Malware attack	Malicious software injection to compromise devices & data	C, I, A	connected devices firmware
Node malfunction	Genuine node dysfunction due to data scarcity/malware	I, A	Components: RSUs, sensors
Node outage	Genuine node cease functions	A	Components: ECUs, sensors
Side channel attack	Collects system data from leaks or external sources	C	ECUs, telematic units

and ω is the list of all join type sets (ε); required to exploit the goal; $\varepsilon \in$ [{OR}, {AND}, {OR, AND}].

While AGs offer network security insights, they lack quantitative analysis for estimating attack likelihood and impact. However, when combined with Bayesian networks (BNs), they introduce the quantitative aspects needed to assess a network's security vulnerabilities and model threat level.

Table 5.2: Attacks Modules

Attack	Pre-condition	Post-condition	Join Type
Node subversion	Limited authentication; malicious code	Attacks requiring authentication bypass	OR
Jamming & DoS	Attacker within the network reach	Disrupt network communication	OR
Packet sniffing/ eavesdropping	(Malicious node; unencrypted communication) - side channel attack	Obtain unencrypted data	AND-OR
Sybil attack	Node subversion; Insufficient or No authentication	Bypass authentication; break key distribution	OR
Masquerading	Node subversion; Sybil	Bypass authentication	OR
Spoofing	Sybil; Node subversion	Degrade authentication	AND
Data injection/ modification	Adversary node; legitimate part of the network	Jamming; Falsified data in network	AND
Altering attack	Data injection; attacker within network	Integrity loss	AND
Replay attack	Sybil, Spoofing	Energy & integrity loss	OR
Wormhole	(Node subversion; Sybil); Spoofing	route control, communication & packet loss	AND-OR
Sinkhole; Blackhole	Wormhole	Network prone to replay, rerouting; loss of C-I-A	AND
Spamming; Flooding	Spoofing, Sybil, Malware Attack	Energy, integrity & availability loss	OR
Physical attack	Topology discovery	Node outage	OR
Malware attack	Attack surface architecture; User code execution	Node Subversion	OR
Node Outage	Low energy on node; malware attack	Faulty data in network	OR
Node Malfunction	Energy drain; Directed physical attack	Network disruption	OR
Side channel attack	Directed physical attack	Data leakage, network disruption	OR

5.1.5 Threat Level Estimation

To determine the threat level of a vulnerability in a network, it is important to quantify the potential risk posed by the vulnerability. The approach in this research implements a framework using the scoring metric established by CVSS for wired networks [42], to quantify vulnerabilities and thus determine the likelihood of success of an attack s_n: $\Pr(s_n)$ within a network of connected devices. The assessment of CVSS scores involves three factors: base, temporal, and environmental metrics. Base metrics measure the difficulty level in exploiting a vulnerability, the temporal metrics reflect the characteristics of a vulnerability that may change over time but not across user environments, and the environmental metrics assess how vulnerability exploitation affects the network's deployed environment [42].

The likelihood of success and impact of an attack can be assessed using the CVSS severity ratings. Table III details these metrics, their attributes, descriptions, severity, and values. However, given the specific threat landscapes, rating values may differ across organizations. The next section explains the use of CVSS metrics to quantify the aspects of a vulnerability.

5.1.5.1 Misuse Frequency and Misuse Impact To quantify vulnerability and attack likelihood, Houmb's model [121] and the CVSS metrics were used. [121] assesses the risk level of an attack by considering the probability over Misuse Frequency (MF) and Misuse Impact (MI). The Local Static (LS) or unconditional MF (MF_{LS}), represents the static likelihood of an attack, taking into account factors like attack attributes, network architecture, and security measures in place, and independent of the network environment or other attacks happening in the network. MF_{LS} (eq. 5.1-5.3) is computed using the Temporal (T) and Base Exploitability (BE) metrics from the CVSS metrics. The initial MF (MF_{init}) is defined by considering the exploitability sub-score within the base metrics (eq. 5.1). Similarly, the MF_{temp} of an attack, representing the characteristics of a vulnerability that may change over time, is computed using the temporal metrics (eq. 5.2). The MF_{LS} is then derived by averaging MF_{init} and MF_{temp} (eq. 5.3).

$$MF_{init} = \frac{5}{18} \times B_{Scope} \times \sum (BE_{AV}; BE_{AC}; BE_{PR}; BE_{UI}) \tag{5.1}$$

$$MF_{temp} = \frac{1}{3} \times \sum (T_E; T_{RL}; T_{RC}) \tag{5.2}$$

$$MF_{LS} = \frac{1}{2} \times \sum (MF_{init}; MF_{temp}) \tag{5.3}$$

Moreover, the LS impact or severity of a successful attack on a network is expressed in the MI_{LS}. Similarly to the MF_{LS}, the impact sub-scores of CVSS base metrics on (C, I, A) form the basis for the initial impact estimate (MI_{init} in eq. 5.4). The environmental impact MI_{env} is then determined based on the environmental metrics (eq. 5.5), and the final MI_{LS} estimate is derived by averaging MI_{init} and MI_{env} (eq. 5.6).

$$MI_{init} = \frac{11}{20} \times \sum (BI_C; BI_I; BI_A) \tag{5.4}$$

$$MI_{env} = \frac{2}{9} \times \sum (E_{CR}; E_{IR}; E_{AR}) \tag{5.5}$$

$$MI_{LS} = \frac{1}{2} \times \sum (MI_{init}; MI_{env}) \tag{5.6}$$

Given the upper bound Ratings and Values from the CVSS metrics (Table 5.3), each equation (eq. 5.1-5.6) is normalized with coefficients constraining the final values for the MF_{LS} and MI_{LS} within the range of 0 to 1. Although MF_{LS} and MI_{LS} characterize vulnerabilities, modeling risk requires a distribution of outcomes and therefore a probabilistic RA framework. The next section uses complex probabilities to asses attack likelihood and impact, beyond static MF and MI.

5.1.5.2 Complex Probabilities They are essential in quantum physics and computing due to interference, a unique property absent in real numbers [122]. With interference, adding two complex numbers can yield a smaller result than either original number. Mathematically, for the addition of two complex numbers to show interference, one or both components need to have opposing signs. In the proposed RA framework, complex probabilities and their two-dimensional nature help depict multiple aspects of the attacks and capture attackers hindering each other. A complex probability z_n of an attack can be defined as:

$zn = a_n + b_n i$, where a_n, the real component, represents the likelihood of occurrence of the attack, and b_n is the imaginary component representing the impact of the attack. a_n and b_n are functions of the MF_{LS} [in eq. (5.3)] and MI_{LS} [in eq. (5.6)] respectively, as defined in eq. (5.7-5.8).

$$a_n = \Re(z_n) = \mu \times MF_{LS} \tag{5.7}$$

$$b_n = \Im(z_n) = \pm\beta \times MI_{LS} \tag{5.8}$$

μ and β are constants defined by the security administrator, where $(\beta, \mu) \in [0.05, 1.0]$. μ represents the effectiveness of security measures, with higher values indicating lower

Table 5.3: CVSS Metrics and Attributes for MF and MI

Metric	Attributes	Description	Rating and Value	
Base Exploitability (BE) Metrics	Attack Vector (BE_{AV})	Ease with which a vulnerability can be exploited	Physical (P)	0.20
			Local (L)	0.55
			Adjacent network (A)	0.62
			Network (N)	0.85
	Attack Complexity(BE_{AC})	Difficulty of the attack	High (H)	0.44
			Low (L)	0.77
	Privileges Required(BE_{PR})	Privileges for a vulnerability exploit	High (H)	0.27
			Low (L)	0.62
			None (N)	0.85
	User Interaction (BE_{UI})	User interaction for exploit	Required (R)	0.62
			None (N)	0.85
Base Scope	Scope (B_{Scope})	Vulnerability security scope	Unchanged (U)	1.00
			Changed (C)	1.08
Temporal (T) Metrics	Exploit code Maturity (T_E)	Availability status of exploit techniques or code	Unproved (U)	0.91
			Proof-of-concept (POC)	0.94
			Functional (F)	0.97
			High/Not defined (H)(N)	1.00
	Remediation Level (T_{RL})	Solutions available to fix vulnerability	Official Fix (O)	0.95
			Temporary Fix (T)	0.96
			Workaround (W)	0.97
			Unavailable/Not defined (U)(N)	1.00
	Report Confidence (T_{RC})	Evidences on vulnerability existence	Unknown (U)	0.92
			Reasonable (R)	0.96
			Confirmed/Not defined (C)(N)	1.00
Base Impact (BI) Metrics	Impact on Confidentiality (BI_C), Integrity (BI_I), Availability (BI_A)	Impact on information resource Confidentiality, Integrity and Availability	None (N)	0.00
			Low (L)	0.22
			High (H)	0.56
Environmental (E) Metrics	Confidentiality (E_{CR}), Integrity (E_{IR}), Availability (E_{AR}) Requirements	Importance of confidentiality, Integrity & Availability	Low (L)	0.50
			Medium (M)	1.00
			High (H)	1.50

security. β estimates the impact of a successful attack, with higher values signifying greater potential impact. In this research, μ and β were determined by computing eq.(5.7-5.8) for the attacks in Table 5.2, with the CVSS metrics from Table 5.3 [123]. Acceptable ranges for the resulting values of a_n and b_n) were determined based on the defined threat model and the corresponding CVSS Base Exploitability (BE) and Base Impact (BI) metrics. The means of μ and β values that yielded a_n and b_n within these ranges were selected as the optimal $\mu(0.8)$ and $\beta(0.75)$, as shown in Fig. 5.3 [123].

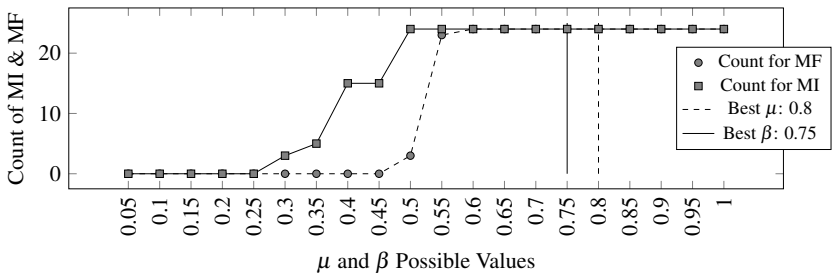

Figure 5.3: Count of MF and MI within Acceptable Ranges given μ and β

Advantage of complex probabilities: Complex probabilities enhance cybersecurity risk assessment by offering nuanced insights into attack likelihood and impact. They illustrate both the probability of an attack occurring and the severity of its consequences, providing a comprehensive view of vulnerabilities. Additionally, complex probabilities allow for a precise representation of how attacks interact. The proposed framework uses the imaginary part to signify constructive or destructive interactions between attacks. This capability allows for a more accurate estimation of the overall impact when multiple attacks simultaneously occur. For instance, in a scenario where two different attacks simultaneously occur and interfere with each other, complex probabilities can capture the combined effect more accurately than real probabilities. In addition to their algebraic

structure, complex probabilities enable risk estimation rooted in probabilistic principles. Unlike 2D vectors, they naturally constrain likelihood within 0 to 1 and impact within -1 to 1, ensuring a probabilistic outcome. Moreover, complex probabilities facilitate objective quantification of the impact of backtracking in the network. This is particularly important in risk modeling, where backtracking refers to an attacker revisiting previously compromised states. The framework enables the quantification of the extent of damage caused by such backtracking, especially when cycles arise due to the attacker repeatedly revisiting already compromised nodes. Following eq. (5.7-5.8), the unconditional probability of a successful attack can be defined as follows.

Definition 5: Unconditional Probability of Successful Exploitation: *The Local Static (LS) or unconditional likelihood of successfully exploiting a vulnerability and executing an attack s_n, denoted as z_n, is depicted by a complex number in the form of $a_n + b_n i$. With a_n the unconditional probability of occurrence of s_n, determined based on exploitability metrics, defined in eq. (5.3) and b_n, the impact of s_n on the network architecture and security measure as defined in eq. (5.6).*

LS probabilities assess individual attacks, but a comprehensive risk assessment requires considering the network environment and possible attack interactions. The following section details how LS probabilities and complex numbers estimate the conditional probability of an attack's success in BAGs, depicting the overall threat level and impact on a network.

5.1.6 Threat Level Modeling with Bayesian Networks

This section presents the use of complex probabilities and unconditional threat levels to model conditional probabilities of attacks, considering attacks' inter-dependencies.

5.1.6.1 Conditional Probability Conditional probability accounts for dependencies where the success of one attack depends on others. The two types of dependencies or join types are OR and AND (Section 5.1.4). For an attack $s_n \in$ S, $Pa[s_n]$ represents the parent set of attacks whose post-conditions lead to s_n's pre-conditions i.e. attack $s_k \in Pa[s_n]$ is considered a parent of s_n in the AG. The set $Pac[s_n]$ includes attacks that lead to the pre-conditions of s_n and any compromised child nodes. This distinction is crucial because compromising a child node affects overall network security, even if the attacker revisits the parent node. BAGs assess attack net risk using conditional probabilities, defined as follows:

Definition 6: Bayesian Attack Graph - *A BAG is defined as a tuple (S, α, ω, Pr), S is the set of all possible attacks in the network. α: set of pre and post-conditions of attacks in S; ω, the list of all attacks join type sets (ε); $\varepsilon \in$ [{OR}, {AND}, {OR, AND}] and Pr, the set of conditional probabilities indicating the attack' success and impact in the BAG.*

Definition 7: Conditional Probability of Successful Exploitation : *For a BAG with the tuple (S, α, ω, Pr), the conditional probability of any attack $s_n \in S$: $Pr(s_n)$ represents the attack likelihood and impact, and is given in eq. (5.9) where the conditional MF and MI are defined in eq. (5.10)-(5.13) for each join type on the set of parent attacks $Pa[s_n]$ or parent and child set of attacks: $Pac[s_n]$, where $s_k \in (Pa[s_n], Pac[s_n])$ and $s_k = 0$ when the attack is unsuccessful or $s_k = 1$ otherwise.*

$$Pr(s_n|Pa[s_n]) = MF(s_n|Pa[s_n]) \pm i * MI(s_n|Pa[s_n]) \tag{5.9}$$

Join type: AND

$$MF(s_n|Pa[s_n]) = \begin{cases} 0, \exists s_k \in Pa[s_n] | s_k = 0 \\ MF_{LS}(s_n) \cdot MF\left(\bigcap_{s_k=1} s_k\right) \end{cases} \tag{5.10}$$

$$MI(s_n|Pac[s_n]) = \begin{cases} 0, \exists s_k \in Pac[s_n] | s_k = 0 \\ MI_{LS}(s_n) \cdot \left[MI_{LS}(s_n) \pm MI(\bigcap_{s_k=1} s_k) \right] \end{cases} \tag{5.11}$$

Join type: OR

$$MF(s_n|Pa[s_n]) = \begin{cases} 0, \forall s_k \in Pa[s_n], s_k = 0 \\ MF_{LS}(s_n) \cdot MF(\bigcup_{s_k=1} s_k) \end{cases} \tag{5.12}$$

$$MI(s_n|Pac[s_n]) = \begin{cases} 0, \forall s_k \in Pac[s_n], s_k = 0 \\ MI_{LS}(s_n) \cdot \left[MI_{LS}(s_n) \pm MI(\bigcup_{s_k=1} s_k) \right] \end{cases} \tag{5.13}$$

Acknowledging the relationships between all attacks, especially destructive ones is important to compute conditional probabilities. The next sub-section explains how attacks can affect each other constructively or destructively.

5.1.6.2 Constructive and Destructive Attacks in BAGs Considering the interactions between attacks in a network under attack is crucial. BAGs are built for each security parameter (C, I, A) by assessing dependencies among attacks targeting a similar parameter. The threat model (Section 5.1.2) assumes any attack can occur, and the analysis of attack dependencies assumes their simultaneous execution on network resources. Table 5.4 shows attacks with destructive or interfering impacts on others. Attacks not mentioned are considered to have a neutral or constructive impact. When attacks have a neutral impact (which can have both constructive and destructive impacts), the overall effect is considered constructive.

For instance, in a network of connected devices under attack from attackers *A* and *B*, where a Sybil attack from attacker *A* on the entire network compromises the network confidentiality. A Node Subversion attack from attacker *B* would result in compromising an illegitimate node. However, having a Sybil attack occurring on only a part of the network and Node Subversion on another part contribute towards compromising the entire network's confidentiality. Consequently, depending on the scenario, Sybil and Node Subversion

attacks can have either destructive or constructive impacts. They are therefore considered as having a neutral impact. For the computation, the constructive or destructive interactions between attacks are translated in eq. (5.11) and eq. (5.13) with the «±» sign. Attacks will be considered to have a constructive impact when their *MI* operator signs are identical and a destructive impact otherwise. Further, the research assumes nonzero prior probabilities of the attacks, not considering the dynamic risk assessment which considers the evidence of an attack in the network and then recomputes the risk level.

Table 5.4: Identified Destructive Attacks for each Security Parameter

Attacks	Explanation	Sec. parameter
(Sinkhole/Spamming) VS Jamming/DoS	Jamming disrupts communication & hinders effectiveness of attacks relying on it.	A
Altering VS Masquerading attacks	Altering sends invalid data to deter the impact of a masquerading attack on network integrity.	I
Node Malfunction/Outage VS Malware attack	A successful malware attack renders a node malfunction/outage attack useless, and vice versa.	A

5.2 Results and Discussion

This section details the design of the C, I, and A BAGs for the threat scenario in Section 5.1.2 and the network architecture in Section 5.1.1. It also considers potential attacks from Table 5.1 and their interdependencies (Table 5.2). The framework is applied to a CAV network use case, illustrating backtracking handling. Python scripts using the `networkx` library were created to develop the BAGs for each security parameter.

5.2.1 Attack Graphs Construction

This section depicts the computations of the LS probabilities for (C, I, A) exploits, given the threat model, the assumptions that any attack can happen (Section 5.1.2), and the values of β and μ as defined in Section 5.1.5. Python scripts following algorithms 1 and 2 were

written to compute the conditional MF and MI for each attack (s_n). Algorithm 1 computes the AND MF conditional probability of an attack s_n. This represents the computation of the conditional likelihood (MF) of s_n given its unconditional (or Local Static (LS)) likelihood (MF_{LS}) and an existing AND join-type on its parent attacks. The inputs are the *LS MF* of the attacks s_n: $MF_{LS}(s_n)$, and the MF of its parent attacks: MF_{pa}. Line 1 sets the conditional MF of the attack to $MF_{LS}(s_n)$, lines 2-3 iterate over the MF of the parent attacks and update $MF(s_n)$ by multiplying it with each MF_{pa}. Finally, line 5 returns the conditional $MF(s_n)$.

Algorithm 1 AND MF Conditional Probability

Input: Local Static MF_{LS} of attack s_n: $MF_{LS}(s_n)$; MF of Parent attacks of s_n: MF_{pa}
Output: conditional MF of attack s_n resulting from AND join-type on its parent attacks MF_{pa}

1. $MF(s_n) = MF_{LS}(s_n)$
2. **for** *i in* MF_{pa} **do**
3. $\quad MF(s_n)$ *= $MF_{pa}[i]$
4. **end**
5. **return** $MF(s_n)$

Algorithm 2 computes the OR MI conditional probability of an attack s_n. This represents the computation of the conditional impact (MI) of s_n given its unconditional (or Local Static (LS)) impact (MI_{LS}) and an existing OR join-type on its parent attacks. Line 1-2 assigns the conditional MI: $MI(s_n)$ and the union of s_{pa} to the *LS MI* of the attack s_n: $MI_{LS}(s_n)$ and the MI of the first parent attack of s_n. Then, lines 3-6 iterate over the MI of the remaining parent attacks, updating the intersection probability of $\bigcap_{s_{pa}}$ and the current parent, the occurrence of one excluding the other and $\bigcup_{s_{pa}}$. Finally, line 7 computes $MI(s_n)$ eq. (5.13), returned on line 8. Algorithm 1's correctness was established through step-by-step reasoning and a loop invariant, ensuring that it accurately computes

Algorithm 2 OR MI Conditional Probability

Input: Local Static MI_{LS} of attack s_n: $MI_{LS}(s_n) = c$; MI of Parent attacks of attack s_n: MI_{pa}
Output: Conditional MI of s_n resulting from an OR join-type of parent attacks MI_{pa}

1. $MI(s_n) = MI_{LS}(s_n)$
2. $\cup_{s_{pa}} = MI_{pa}[0]$
3. **for** *i in* $(1, MI_{pa})$ **do**
4. $\quad \cap_{s_{pa}} = \cup_{s_{pa}} \cdot MI_{pa}[i]$
   $\quad exclusion_1 = \cup_{s_{pa}} - \cap_{s_{pa}}$
   $\quad exclusion_2 = MI_{pa}[i] - \cap_{s_{pa}}$
   $\quad \cup_{s_{pa}} = exclusion_1 + \cap_{s_{pa}} + exclusion_2$
5. **end**
6. $MI(s_n) = MI(s_n) \cdot \left[MI(s_n) + \cup_{s_{pa}}\right]$
7. **return** $MI(s_n)$

an intersect probability that reflects the "AND" relationship among parent attacks [123]. Similarly, Algorithm 2's correctness was validated through detailed step-by-step reasoning, accounting for graph computations and cycles, ensuring it computed a union probability that reflects the "OR" relationship among parent attacks [123]. Moreover, the complexity of both algorithms varies depending on the presence of cycles in the graph. In the best case scenario, a directed acyclic graph can be processed efficiently in topological order with a runtime of O(V+E), with V the number of nodes, and E the number of edges. However, graphs with cycles require iterative updates until stabilization, resulting in O(kc) runtime, for k cycles with c nodes each. However, Section 5.2.4 shows that the framework computations lead to fast convergence after a few cycles, resulting in O(V+E+klog(c)) runtime [123]. This makes the framework scalable and suitable for resource-constrained devices, avoiding heavy computations associated with approximation algorithms [**?**] [21].

For each security parameter, the expected exploitability or threat level reflects a security administrator's perception of compromise likelihood. In the absence of security measures, a 50% chance of successful attack is assumed to account for uncertainty, as there is an equal chance of a successful attack. The expected impact is similarly set at 50%. With

$\mu = 0.8$ and $\beta = 0.75$, the final expected threat and impact levels are 0.40 and 0.38, respectively. Fig. (5.4-5.6) show net threat and impact levels for each attack based on the all the pre-conditions. In reality, attackers may follow specific attack paths, as discussed in Section 5.2.5. Initial requirements for an attack are indicated in gray, constructive attacks in white, and destructive attacks in yellow squares with a dot. For example, Topology Discovery, Spoofing, and Jamming are categorized as initial requirements, constructive, and destructive attacks for Availability (A) compromise. Subsequent sections detail the construction and results for (C), (I), and (A) compromise.

5.2.2 Violation of Confidentiality (C)

To define the nodes of the (C) BAG, attacks in Table 5.1 compromising the (C) are considered. For each attack, the values of the CVSS metrics are determined based on the attack characteristics and, independently of the other attacks (Table 5.3). These values are then used to compute the LS or unconditional probabilities (MF_{LS} & MI_{LS}). For example, for an attack like eavesdropping that compromises the (C), the (ratings, values) for BI_C, BI_I, and BI_A would be (High, 0.56), (None, 0.0) and (None, 0.0) respectively. For the MF_{LS} on the masquerading attack for example, the base metrics (BE_{AV}, BE_{AC}, BE_{PR}, BE_{UI}) and temporal metrics, (T_E, T_{RL}, T_{RC}) are (N, L, L, R) and (F, W, R) respectively, leading to MF_{init}=0.786 (eq. 5.1) and MF_{temp}=0.96 (eq. 5.2), resulting MF_{LS}=0.876 (eq. 5.3). Once the LS probabilities MF_{LS} and MI_{LS} are computed for each attack, the logical connections between the attacks are identified with pre and post-conditions from attack modules (Table 5.2). The unconditional likelihoods MF_{LS} are next used to compute conditional probabilities, considering pre and post-conditions (eq. 5.10-5.13).

Fig. 5.4 displays conditional probabilities for each node or attack, the net threat (31%) and net impact levels (42%) for (C) compromise. The net threat (31%) is lower than

the expected threat (40%) level, while the net impact level (42%) is higher than the expected (38%). Those values are likely due to factors like (a) **Number of attacks:** The computations (Fig. 5.4) assume all pre-conditions as successful, considering all attack paths for net threat and impact levels. In reality, attackers may follow specific paths, leading to lower threat and impact levels. **(b) Type of attacks:** Attacks differ in likelihood and impact. For instance, a Sinkhole attack may have a low likelihood but high impact if successful, while a Node Subversion, though more likely, might have a lower impact on the network. **(c) Attacks dependencies:** The net impact level varies based on the attack relationships. Constructive relationships among attacks can increase the overall impact, while destructive ones can interfere and lower it.

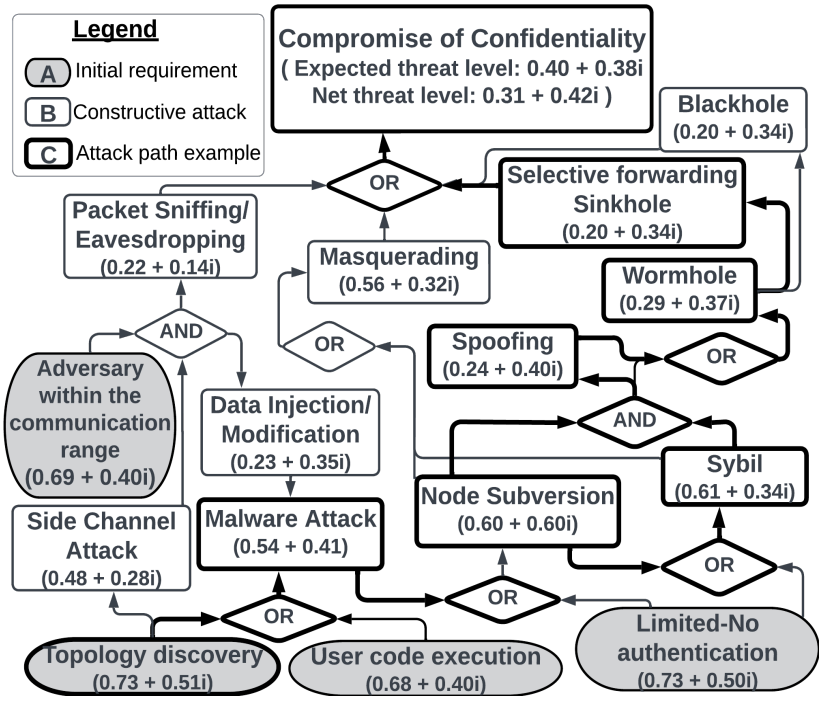

Figure 5.4: Bayesian attack graph for violation of Confidentiality

5.2.3 Violation of Integrity (I) and Availability (A)

Similar computations helped determine the net threat & impact levels for (I) (Fig.5.5) and (A) (Fig.5.6) BAGs, with net threat levels at 35% for (I) and 39% for (A). Although both net threat levels are below the expected 40%, the (A) net threat level (39%) exceeds the (I) level (35%) and is very close to the expected threat level (40%). The difference can be attributed to the number and type of attacks needed to compromise each parameter. 12 total attacks contribute to (A) compromise, with 6 leading directly to the goal state, while 11 total attacks for (I), with 5 directly contributing to (I) compromise. Additionally, attacks affecting (I) have an average likelihood of occurrence of 40% versus 52% for the attacks affecting the (A), showing how the type and likelihood of attacks affect the likelihood of security parameter compromise.

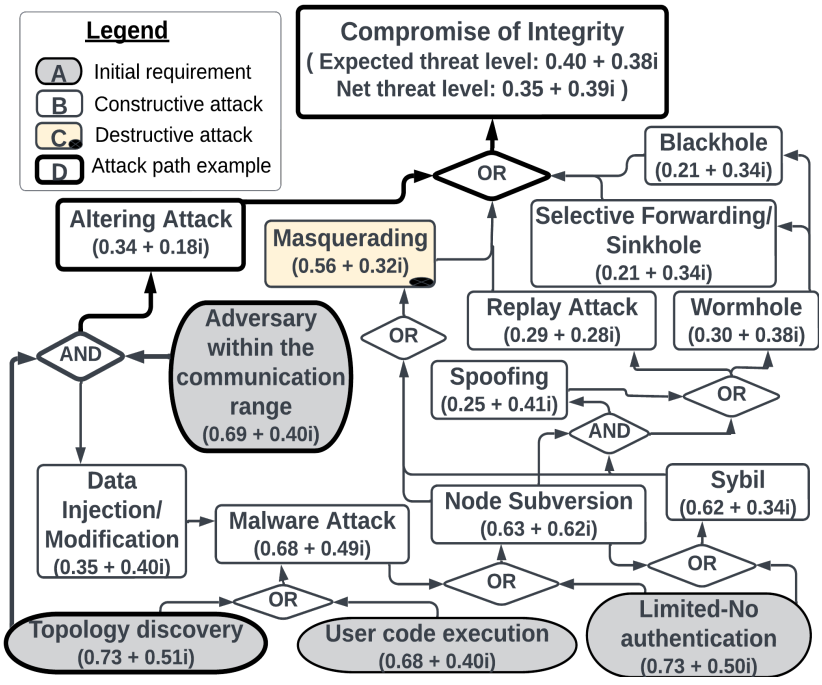

Figure 5.5: Bayesian attack graph for violation of Integrity

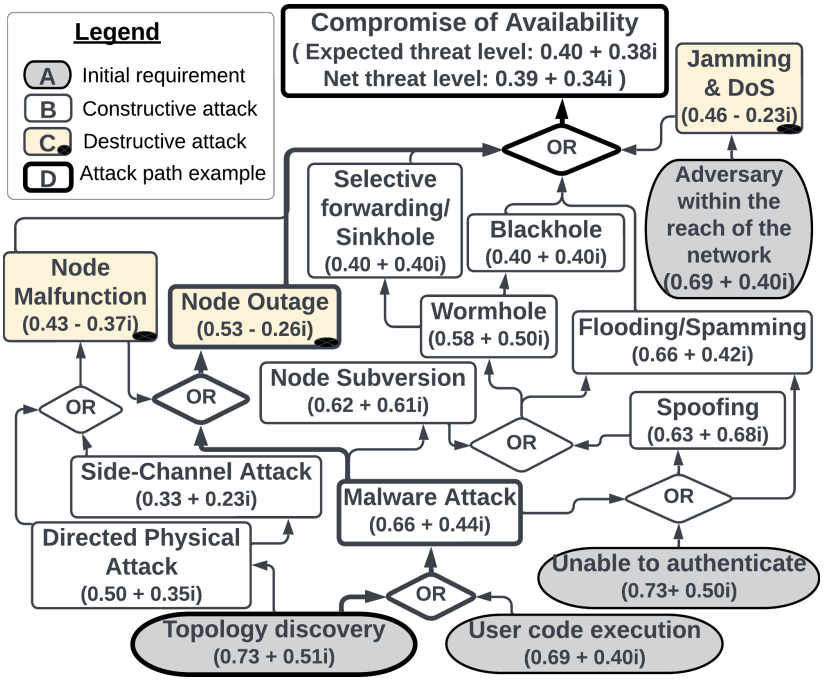

Figure 5.6: Bayesian attack graph for violation of Availability

Regarding MI, unlike the (C) BAG, the (I) and (A) BAGs include destructive attacks (Masquerading attack for (I), Node Malfunction/Outage, and Jamming for (A)). These attacks reduce the impact of the constructive attacks, as shown by the negative sign on the MI (imaginary part) of the destructive attacks in Fig. (5.5-5.6). Although the (A) compromise has a higher net threat level, the resulting net impact is less than the (I) compromise, this is due to more interfering attacks: 1 for the (I) versus 3 for the (A). To better understand the effect of destructive attacks, let's consider the compromise of Availability (A) as the final goal with net threat and impact levels as $Pr(A) = 0.39 + 0.34i$. The pre-condition attacks directly compromising the (A) are Jamming/Dos (DS), Blackhole (BH), Sinkhole (SH), Node Outage (NO), and Node Malfunction (NM) with DS, NM & NO destructive and BH & SH constructive attacks. Assuming all pre-condition attacks as constructive, the net impact levels stay unchanged for constructive attacks, while the

net impact levels of the destructive attacks become: $Pr(DS) = 0.46 + 0.23i$, $Pr(NM) = 0.43 + 0.37i$ and $Pr(NO) = 0.53 + 0.26i$. To compute the resulting impact on (A), the union MI of all successful pre-condition attacks $s_k \in$ (DS, BH, SH, NO, NM) is:

$$= MI(\cup_{s_k=1} s_k) = \sum_{\text{(DS,... NM)} \in \text{[T,F]}} \left[\prod_{i \in [DS,...NM]} (MI_i) \right]$$

$$= 0.871$$

With the expected threat impact of 0.38, the final MI on the Availability will be computed using eq. (5.13) as:

$$= 0.38 * [0.38 + 0.871] = 0.475$$

Considering all pre-condition attacks as constructive, the net threat impact on Availability is 47%, versus 34% when considering the interference between attacks. This 13% difference emphasizes the significance of accounting for types of attack interactions when performing risk assessment. Understanding these dynamics helps allocate resources more effectively and inform security strategies, leading to improved incident response capabilities. The following section demonstrates how the framework addresses backtracking, particularly in cases of repeated compromise in backtracking, leading to cycles.

5.2.4 Backtracking

Backtracking occurs when an attacker, faced with unsuccessful attempts or strong security measures, reverts to a previous state to explore alternative paths or stay undetected. This technique helps attackers find vulnerabilities or weaknesses in a network. Addressing

this dynamic behavior is important in cybersecurity, as it reflects the adaptability of attackers. When an attacker backtracks, the framework must account for these revisits to model the evolving threat landscape accurately. This section develops a use-case scenario showing how the framework handles backtracking.

Considering a BAG with two nodes representing potential attacks (Fig. 5.7), a successful Wormhole (WH) attack on node WH with conditional: $Pr(WH) = 0.40 + 0.41i$ and LS: $P_{LS}(WH) = 0.67 + 0.42i$. Another not yet successful Sinkhole attack (SH) on node SH, with LS probability of $P_{LS}(SH) = 0.67 + 0.64i$. Node WH is the parent of node SH, i.e., post-conditions on node WH satisfy the pre-conditions to execute node SH. Fig. 5.7 illustrates that the edge between the nodes WH and SH (step 0) corresponds to the attacker successfully compromising node SH from node WH, while step 1 corresponds to the attacker backtracking to previously compromised node WH from node SH. On step 0, using eq. (5.12-5.13), the conditional probability on node SH becomes $Pr(SH) = 0.27 + 0.68i$. The updated $MI(WH)$ and $MI(SH)$ can be seen on the first values (on state 0) of Fig. 5.8. When backtracking for the first time (step 1), the MF is assumed unchanged for both nodes, as both states have already been compromised once. This aligns with the MF definition, based on factors like attack attributes, and network architecture to represent exploitability, remaining consistent as per the defined threat model (Section 5.1.2). Therefore, backtracking affects the resulting impact when exploiting the network's security parameters and not the exploitability itself.

After node SH is exploited (state 0) and released (state 1), eq. (13) computes the new MI(W H) when backtracking, revealing an increased MI(W H) of Pr(WH) = 0.40+0.46i due to SH's exploitation. Although backtracking allows attackers to explore vulnerabilities and evade detection, it may also deplete resources and introduce delays, amplifying its negative impact on network security. Initially, when backtracking repeatedly between

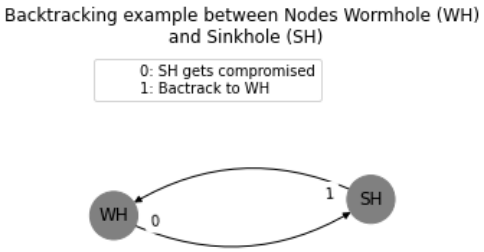

Figure 5.7: Backtracking example between nodes WH and SH

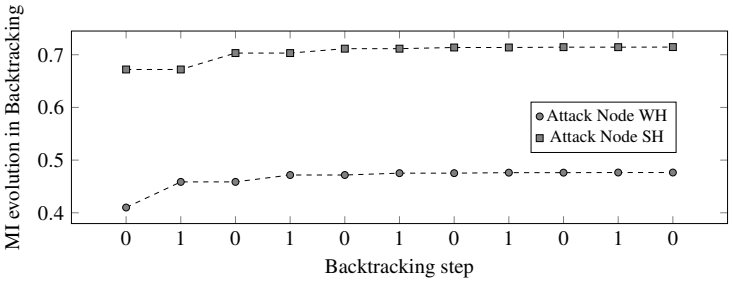

Figure 5.8: Misuse Impact (MI) Evolution within Five Backtracking Cycles

WH and SH, both MI(WH) and MI(SH) increase as resources are potentially depleted and new vulnerabilities are discovered (Fig. 8). However, after several cycles, the impact stabilizes as the attacker moves through compromised states without exploiting any new vulnerabilities. This highlights how networks can adapt and respond to threats, with an initial impact increase during the first backtracking cycles, followed by a level of impact when no new vulnerability is exploited. This emphasizes the need for continuous monitoring and robust security practices to defend against persistent attackers. The next section will detail the computation of net threat levels for the (C), (I), and (A) security parameters in a simulated CAVs network.

5.2.5 Use Case Scenario: CAV Network

This section outlines a compromise scenario in a CAV network with: **(a) Two AVs (V & V2V):** communicating for coordinated driving; **(b) Satellite Connection (V2X):** for wide-area communication, data transfer, and navigation updates; **(c) Roadside Infrastructures (V2I):** like intelligent traffic lights and beacons, improving network connectivity and traffic management.

To assess the net threat level, assumptions about network vulnerabilities and potential attack vectors were made. Firstly, an AV entity and its subsystems (V) can be trusted, and the topology of the CAV is known. Moreover, an attacker is considered to be within the communication range of various network layers (V2V, V2X, V2I). Lastly, the values of μ and β, the expected net threat and impact levels for (C), (I), and (A) are set to 0.8, 0.75, 40%, and 38% respectively, as established in Section 5.2.1 and Section 5.1.5.2. Given the assumptions, attack paths of possible compromise were developed for each security parameter. For the (C) compromise, the attack path is **Topology discovery, malware attack, Node Subversion, Sybil, GPS Spoofing, Wormhole**. For the (I), **Topology discovery then Altering attack**. For the (A), **Topology discovery, Malware attack and node outage**. The attack paths are illustrated with bolded edges in Fig. (5.4, 5.5, 5.6).

The Unconditional or Local Static likelihood (MF_{LS}) and impact (MI_{LS}), along with the conditional probabilities for each attack were computed, resulting in the net threat and impact levels for each security parameter. Table 5.5 shows the computed values for the MF_{LS}, MI_{LS}, MF, and MI for the attacks and security parameters defined in each attack path. Table 5.5 shows that the net threat level for violating Confidentiality is 2%, indicating low exploitability due to the need for consecutively successful attacks for (C) compromise. The net threat impact is 26%, closer to the expected 38%, suggesting that while exploitability is almost negligible, the potential impact on Confidentiality is

significant if the attacker reaches their goal. For instance, a successful sinkhole attack could disrupt communication between AVs and/or the infrastructure, posing privacy and safety risks. The net threat and impact levels are similar and below expected values: 14% & 14% for net threat, and 21% & 26% for net impact for (I) & (A) respectively. This similarity arises because of the number of attacks affecting both parameters (1 and 2 attacks for (I) and (A) respectively compared to 5 for (C) compromise). In fact, despite similar average net threat levels for attacks in each attack path (68% for (I), 69% for (A) and 68% for (C)), the number of attacks in each attack path accounts for the similarity in the resulting net threat for (I) and (A), and the disparity with the (C) compromise.

Table 5.5: CAV Attack Paths Probabilities

Use Case	Attack Compromise	MF_{LS}	MI_{LS}	MF	MI
1	Node Subversion	0.68	0.50	0.36	0.43
1	Sybil attack	0.68	0.31	0.25	0.22
1	GPS Signal Spoofing	0.66	0.54	0.06	0.34
1	Wormhole	0.67	0.40	0.06	0.19
1	Sinkhole	0.67	0.40	0.08	0.34
1 & 3	EV Charging Malware attack	0.72	0.40	0.52	0.36
2	Altering attack	0.68	0.33	0.24	0.18
3	Node Outage	0.66	0.41	0.34	0.32
1	Confidentiality	0.40	0.38	0.02	0.26
2	Integrity	0.40	0.38	0.14	0.21
3	Availability	0.40	0.38	0.14	0.26

Comparative Analysis: Table 5.6 compares the proposed framework to existing RA approaches based on metrics like assessment methods used (BAGs, CVSS, or others), types of IoT attacks addressed, quantification of attack likelihood and impact, inter-attack relationships, scalability, and management of cycles from backtracking.

Table 5.6: Comparative Analysis of Proposed Framework

Research works	Sen et. al [19]	Flores et. al [21]	Oser et. al [124]	Arat et. al [125]	Proposed work
Method	BAGs & CVSS	BAGs & CVSS	CVSS & past exploits	AGs & CVSS	BAGs & CVSS
Exploit quantified	Yes	Yes	Yes	Yes	Yes
Impact quantified	Yes	Yes	No	Yes	Yes
Scalability	Yes	Yes	Yes	Yes	Yes
Cycles handling	No	No	NA	No	Yes
Attack dependencies	Yes	Yes	No	Yes	Yes
Attack destructive interferences	No	No	No	No	Yes
Total Attacks	19	2	NA	13	20
Exploit scale	0-1	0-1	0-10	0-1	0-1
Impact scale	0-1	0-800	0-10	0-1	0-1
Exploit granularity	BE&T	BE	BE	BE	BE&T
Impact granularity	BI&E	BI&E&T	BI	BI	BI&E
Exploit complexity	O(n^2)	O(2^n)*	O(n)	O(V+E)	O(V+E)

BE, BI, T, E and * are respectively Base Exploitability, Base Impact, Environmental, Temporal and approximated values.

5.3 Conclusion

This study introduces a risk assessment framework for IoT systems using Bayesian Attack Graphs and complex probabilities, with a use case on Connected and Autonomous Vehicles. It highlights attack interactions and the effects of backtracking cycles. The framework employs complex probabilities to assess both the likelihood and impact of attacks, including their constructive and destructive interactions. Future work includes developing a generative AI model for improved contextual understanding and aiding security administrators in designing secure systems for large-scale IoT applications, particularly against hard-to-assess zero-day attacks.

CHAPTER SIX

RO3: ASSESSING INTRUSION DETECTION SYSTEMS' DEPLOYMENT FEASIBILITY IN IOT NETWORKS

After assessing IoT network risks, this chapter addresses the challenge of ensuring that deployed countermeasures are computationally feasible for the constrained edge devices they are intended to protect. We developed a metric for assessing the deployment feasibility of ML-based Intrusion Detection Systems on resource-constrained Vehicle's CAN bus, ensuring that proposed IDS defenses are not only effective but also computationally viable in practice [27]. The proposed metric, the Custom Feasibility Score (CFS), combined standard accuracy measures with resource utilization (CPU, memory, throughput), developed a benchmarking framework over 20 state-of-the-art ML-based IDS models, and showed that many high-complexity, high-accuracy models may fail feasibility tests, underscoring the need to consider resource–performance trade-offs during the design of critical infrastructure defenses. The details on the CFS metric are presented in the following sections.

6.1 Proposed Methodology

In this section, we present detailed information about our proposed methodology to evaluate the deployment feasibility of ML anomaly-based IDS models. The proposed method consists of two distinct steps: a comprehensive analysis and implementation of selected current state-of-the-art ML IDS models and the application of our proposed model deployment feasibility metric, the Custom Feasibility Metric (CFS), details of which has been described in Section 6.1.2.2. The values derived from the comprehensive analysis of the selected ML IDS models served as input for our proposed CFS metric.

6.1.1 ML Anomaly-based IDS Models

The models described in this section represent some of the current state-of-the-art ML anomaly-based IDS models that have been proposed by the research community. A total of nine papers [77, 126–133] were surveyed, resulting in the implementation of twenty proposed models. These papers were obtained from the following venues: IEEE, ACM, International Conference of Emerging Technologies, Wireless Engineering and Technology, Computers and Security, and Sensors and Electronics. Keywords like "machine learning for IDS in CAN", "intrusion detection for CAN", and "deep learning IDS models for CAN" were used as criteria to select these papers, which were published between 2018 and 2024. The models are presented based on factors such as **the security attacks** they were trained for, **the dataset** used to train and test them, **the model's architecture, the system information**, and the **evaluation metrics** used to assess their performance. We categorized these models into two types, namely, Category-1 and Category-2. Category-1 models are models whose architecture has been altered either through integration with another algorithm or by changing its default layers. Category-2 models are models used as is by the authors with no modification to their architecture. The following subsections provide the details for each factor mentioned above for Category-1 and Category-2 models.

6.1.1.1 Category-1 Models: These are models whose existing ML architecture was modified from its initial design by the authors. It is also referred to as Cat-1 models. The following models are the Cat-1 ML anomaly-based IDS used for our comprehensive analysis.

a. Long-Short-Term Memory (LSTM)-based Intrusion Detection System (IDS) [126]: The authors proposed a Long-Short-Term Memory (LSTM)-based Intrusion

Detection System (IDS) to detect security attacks in the CAN bus network of a vehicle in IVN. The LSTM model, a deep learning algorithm, was chosen for the IDS due to its ability to perform better than existing methodologies, such as the Hidden Markov Model (HMM) and frequency-based analysis.

Types of Security Attacks and Dataset: The authors chose to focus on these three security attacks: Denial of Service (DoS), fuzzing, and spoofing. A DoS attack disables the services between the Electronic Control Units (ECU); a fuzzing attack sends random malicious messages to the CAN bus to disrupt the ECU's services; and in the spoofing attack, malicious messages are injected into a specific CAN ID, to create a bias in the ECU, therefore, skewing the operation and performance of the ECUs. These attacks were chosen because of the severity of the damage that can be done using these attack models, as they render the CAN bus useless. The **dataset** used to train and validate the model was collected as CAN messages from a Toyota hybrid car using a CAN analysis tool called Vehicle Spy 3. From these messages, an attack dataset for each attack scenario (DoS, fuzzing, and spoofing) was created using an algorithm. In total, two datasets were created for each attack scenario – a normal (benign) dataset and an attack dataset. These datasets, named the NAIST CAN attack dataset, were used to train and validate the proposed LSTM-based IDS model. The NAIST CAN dataset consisted of the following features: CAN ID, DLC, data [D0-D7], and the label column with the label corresponding to the attack type. Using the proposed LSTM-based IDS model, a binary and multiclass classification was carried out. The three attack datasets were combined to represent a single attack dataset. The dataset was pre-processed and reshaped to three dimensions- samples, time steps, and features. Then, it was split into training and testing using the 80:20 ratio.

Model Architecture: Vanilla LSTM and Stacked LSTM models were implemented. The following are the parameters that were used to implement the LSTM models. For the

binary classification, the parameters used are as follows: the tanh function was the input of the activation function, and the sigmoid function was the output of the activation function. The optimizer function was Adam, with a 0.0001 learning rate in 200 epochs and a batch size of 512. The loss function was binary cross-entropy, and the label encoder was the encoder function used. Likewise, for the multiclass classification, the following parameters were used: the sigmoid function was the input of the activation function, and the softmax function was the output of the activation function. The optimizer function was Nadam, with a 0.0001 learning rate in 200 epochs and a batch size of 512. The loss function was categorical cross-entropy, and the label encoder was the encoder function used. Through hyperparameter tuning, the authors were able to identify the parameters that gave the best detection accuracy and rate.

System Information and Evaluation Metrics: The simulation for the proposed model was conducted with an Intel Core i7 CPU 2.20 GHz, 16 GB RAM, Windows 10 (64-bit), and NVIDIA GeForce GTX 1050. PyCharm IDE 2019 2.2, Keras, and TensorFlow were used. The authors evaluated the performance of the models using accuracy, detection rate, F1 scores, and Area Under The Curve (AUC)-Receiver Operating Characteristics (ROC) curve. With the parameters used, the proposed LSTM-based IDS achieved 100% accuracy for the DoS attack and spoofing attack and 99.98% for the Fuzzing attack. To evaluate the efficiency and effectiveness of the proposed model, the authors used another dataset, the survival analysis dataset [134], created by the Hacking and Countermeasure Research Lab (HCRL). The proposed LSTM model provided high detection rates compared to the survival analysis method used.

b. CANTransfer Model [77]: This intrusion detection model combines the Convolutional LSTM (ConvLSTM) Model and one-shot transfer learning to detect network

intrusion attacks in the CAN bus. These models were chosen to ensure the proposed IDS model can dynamically adapt to detect new intrusions using a limited amount of new data.

Types of Security Attacks and Dataset: The following security attacks were the focus of this work: Denial of Service (DoS), fuzzing, and spoofing. The DoS attack floods the CAN bus network with high-priority messages, which denies access to other ECUs. In a fuzzing attack, large amounts of randomly generated messages are injected into the network to delay access, while in replay attacks, non-malicious messages are retransmitted to the network to disrupt network traffic. The authors used the DoS attack dataset to train the proposed model, so the fundamental characteristics are learned by the proposed model. Leveraging the trained model's knowledge, the fuzzing and spoofing attack dataset was used to test the ability of the proposed model to detect new attacks. The **dataset** used in the CANTransfer model was generated from two different cars, Kia Soul and Hyundai Sonata, after driving for more than 24 hours with 10 different drivers. The dataset was transformed and converted to a multivariate time series for the ConvLSTM model. The multivariate time series was then converted into a 2D space. This 2D space was used as an input for the CANTransfer model. 75% of the dataset was used for training, 25% of the dataset was used for validation, and 5% was used for testing.

Model Architecture: The proposed model had a multi-layer ConvLSTM. A ConvLSTM was chosen because it is better at capturing spatial-temporal data compared to the other LSTM-based models. It had six layers. To gain an in-depth understanding of the design of a CAN frame, the authors used an existing algorithm to classify the CAN data field bits into constant fields, multivalued fields, and sensor fields. Using the variance factor, the sensor field data was categorized based on the number of unique field values named low (under 65), medium (between 115 and 475), and high (over 1800). Through this classification, the authors discovered that abnormalities and attacks can be detected using frequency-based

algorithms due to the static rate of the CAN bus frame. Once the CANTransfer model was fully trained, one-shot transfer learning was used to detect new attacks using new intrusion samples, such as fuzzing attacks and replay attacks.

System Information and Evaluation Metrics: The simulation was conducted with an Intel(R) Core (TM) i7-7700HQ CPU @ 2.80 GHz, NVIDIA GeForce GTX 1060 GPU, and Linux 16.04 as the operating system. To develop the algorithm, Python 3.6 with the Keras library for deep learning and TensorFlow were used. To evaluate the efficiency of the proposed CANTranser model, a comparative analysis with the following baseline methods - threshold-based method, classification-based models (one-class SVM model and Isolation Forest (IF)), and an ensemble-based model (a Recurrent Neural Network (RNN) with Heuristics) was done. The following metrics - precision, recall, and F1 score- were used to evaluate the models. Compared to the other baseline methods, the proposed model achieved the following scores. For precision, 94.93% for known attacks and 87.97% for unknown attacks. For recall, 95.57% for known attacks and 88.97% for unknown attacks; and the F1 score, 95.25% for known attacks and 88.47% for unknown attacks. Additionally, the integration of one-shot transfer learning in the CANTransfer to detect unknown attacks resulted in 81% for a precision score, 83% for recall, and 82% for the F1 score. Overall, CANTransfer performed well in detecting new intrusion attacks and maintaining the knowledge of previously known intrusions.

c. Deep Transfer Learning-based LeCun Network Model [127]: For immediate detection of anomalies in CAN bus traffic communication, the authors proposed a deep transfer learning-based LeCun network (P-LeNet) model that detects anomalies in traffic communication. Deep Transfer Learning (DTL) enables the reuse of previously trained model knowledge, allowing the proposed LeCun network model to provide immediate anomaly detection in the CAN bus protocol.

Types of Security Attacks and Dataset: The authors focused on these three types of attack: flooding, fuzzing, and spoofing. In a flooding attack, an ECU node holds more than its allocated resources, leading to communication disruption. A fuzzing attack causes the vehicle to be temporarily unavailable due to message transmission with spoofed CAN IDs from a malicious ECU node, and in a spoofing attack, malicious ECU nodes transmit messages with fake IDs that look identical to a valid ECU node. These attacks were chosen due to the intensity and level of attack severity, as they damage the in-vehicle functions. For the **dataset**, the car hacking dataset [135] was used to train and validate the proposed P-LeNet model. The dataset was preprocessed by checking for outliers using Rosner's test, and missing values were estimated using the predictive mean matching method. Pearson's product-moment coefficient was used to assess feature correlation, and minimum-maximum normalization was used to scale the values of the individual features. The dataset was split for training and testing using the 80:20 ratio. The training data was further split into training and validation using the 80:20 ratio. The validation data was used for hyperparameter optimization.

Model Architecture: The P-LeNet model consists of two convolutional layers, two subsampling layers, one flatten layer, one fully connected layer, and one output layer. The first convolutional layer produces 5 output feature maps with a kernel size of 5, while the second convolutional layer produces 20 feature maps with a kernel size of 20. A pooling subsampling layer followed the convolutional layers consecutively. For the activation function in the convolutional layers, the Rectified Linear Unit (ReLU) was used. The flatten layer converts the pooled feature map to a single column, which serves as an input for the fully connected dense layer. For the output layer, the softmax function was used as the activation function. To train the network, the proposed P-LeNet utilized the categorical cross-entropy loss function.

System Information and Evaluation Metrics: The authors did not provide the system information. To evaluate the performance of the proposed P-LeNet model, the authors used the following metrics: accuracy, precision, recall, and F1 score. The authors also carried out a comparison between Traditional Machine Learning (TML) models (Decision Tree, KNN, SVM), Deep Learning (DL) models (LSTM, NN, CNN), and Deep Transfer Learning (DTL) models (ResNet, IncepNet, FCN). The proposed P-LeNet model outperformed all the other models by having a precision score of 98.14%, a recall score of 98.04%, an F1-score of 97.83%, and an ROC AUC score of 95.42% in the training and testing phases. In summary, the P-LeNet model was able to detect new and known intrusions in IVN communication.

d. VGG16, a CNN-based Model [129]: The authors proposed VGG16, a CNN-based model for the detection of unknown diverse forms of attacks. The VGG16 model was chosen because of its ability to maximize learning generability by extracting hierarchical relationships using features of the input data and minimize classification errors.

Types of Security Attacks and Dataset: The authors focused on these security attacks: Denial of Service (DoS), fuzzy attacks, spoofing attacks on the driving gear, and an RPM gauge. In this case, the spoofing attacks considered were specific to the driving gear ECU and RPM gauge ECU. The car-hacking **dataset** from the Hacking and Countermeasure Research Lab (HCRL) [135] and data from the common HCRL intrusion detection archive [136] were used to train and test the proposed model. Both datasets were combined into a single dataset. The dataset was selected due to the many complex features that will aid in model discrimination training. Redundant normal messages in the dataset were removed, and symbol conversion was performed on the remaining messages. Through a one-hot encoding process, different types of attacks were represented using numeric values (1-5), with 0 representing normal messages. The feature vectors in the dataset were converted to

feature images. The feature images served as input for the proposed model, and the dataset was split into training and testing sets with a ratio of 68.05%:31.95%.

Model Architecture: The VGG16 model comprises 5 convolutional layers, 7 pooling layers, and 1 fully connected convolutional layer. The kernel size, 3x3, was applied to all the convolutional layers. The model had a 150 × 150 × 3 image as the input size, with 20 epochs and a batch size of 32. The strides used were 1,1, tanh was the activation function, categorical cross-entropy was its loss function, and adadelta was the optimizer. The cross-entropy function was used to adjust the learning rate. To avoid data overfitting and maintain high precision in intrusion detection, the VGG16 model was combined with an ensemble-based learning algorithm, the XGBoost classifier. XGBoost was used to validate the trained VGG16 model.

System Information and Evaluation Metrics: The simulation was conducted on an Intel Core i3-4160 dual-core CPU clocked at 3.0 GHz, 8 GB of DDR3 RAM, with the operating system as Ubuntu Desktop 20.04.3 LTS and MongoDB 5.0.3 as the database. Python ML libraries (scikit-learn, numpy, scipy, and pandas) and Tensorflow were used for the model implementation. To evaluate the performance of the proposed VGG16 model, a binary and multiclass classification was carried out. Accuracy, precision, and recall were the metrics used to evaluate the performance of the proposed VGG16 model. For binary classification, the proposed VGG16 model has to detect an attack from the dataset that has malicious and non-malicious messages. It achieved 100% for the accuracy, precision, recall, and F1 score. For the multi-classification, the proposed VGG16 has to detect and classify the different types of attacks from the dataset. It achieved 99.7% for accuracy, 99.9% for precision, 98.5% for recall, and 99.2% for the F1 Score. The proposed model, VGG16, was able to minimize classification errors for the multi-classification of attacks.

e. Convolutional Neural Network (CNN) Model [131]: The authors leveraged the use of a Convolutional Neural Network (CNN) trained on generated Recurrence Plots (RPs) from arbitration IDs to provide a unique way to analyze in-vehicle traffic patterns and identify anomalies. Using RPs helps capture the temporal dependency in sequences of arbitration IDs, a feature the state-of-the-art methods do not currently capture. Moreover, a CNN architecture instead of ANN (Artificial Neural Networks) helps prevent over-fitting and facilitates parameter sharing among layers and faster operations.

Types of Security Attacks and Dataset: The authors focused on these security attacks: Denial of Service (DoS), fuzzy attacks, RPM, and gear spoofing attacks. **Two datasets** were used to train the model, a generated dataset (dataset A) and a dataset released by [137] (dataset B). For each dataset, the files with normal and abnormal traffic were preprocessed individually; a total of 128 arbitration IDs of packets arriving in the CAN bus were collected and encoded to a number between 0 and N. The encoded sequence was then converted to CNN's input image using RPs. The N unique arbitration IDs are from the benign dataset, however, any new arbitration ID not found in the available arbitration IDs list might appear in the CAN bus and would be encoded as -1. For all the datasets, the training, testing, and validation set proportions were 60%, 20%, and 20%.

Model Architecture: The RP's images were then used to train the CNN. Hyperband search algorithm [138] was used to fine-tune the model, by speeding up the random search algorithm through adaptive resource allocation and early stopping, and determine the best hyperparameters; they were 64 and 16 for the size of the output layer for the Convolutional Layers 1 and 2 respectively, 0.9 for the Dropout, 32 for the dense nodes, Nadam for the optimizers, 0.0001 for the learning rate, exponential decay for the Learning schedule, 1000 for the decay steps, 0.1 for the decay rate, and 32 for the batch size for both classification

types. The model was trained to perform binary classification as well as multi-class classification to identify different anomalies.

System Information and Evaluation Metrics: The model makes two types of predictions: offline and online (real-time). No specific information was provided about the resources needed to train the model before real-time deployment. For real-time deployment, the CAN bus external host used was NVIDIA's Jetson TX2. In offline prediction, the model needs an average of 117ms to make a single inference if Jetson TX2 is used, with a higher execution time in real-time deployment due to pre-processing steps for live implementation. As evaluation metrics, the authors used the area under the ROC curve (AUC), and the results showed that the model performed better in the multi-class classification on dataset B compared to dataset A, and overall, the model performed better than the Inception-ResNet-based method [137], with an accuracy of 0.999%. Despite achieving a nearly perfect accuracy, generating and processing RPs to train the model can be computationally expensive and challenging to deploy for real-time analysis, as the current automotive controllers have lower processing power than the NVIDIA Jetson TX2 to execute deep learning models.

f. Unsupervised CANet Model [130]: The authors proposed a novel unsupervised deep learning architecture to improve CAN security and the detection of known and unknown intrusion scenarios. The model can handle the data structure of the high-dimensional CAN bus, where the messages are sent at different times and with different lengths.

Types of Security Attacks and Dataset: A real and synthetic CAN datasets were used to train and test the proposed model. The real data was collected from a test vehicle during an experiment. A total of 13 CAN IDs and 20 signals with physical dependencies were used in the experiment. A total of 13 hours of focusing on normal driving conditions were

recorded, with 12.5 hours allocated to training the model and 0.5 hours to testing the model. The synthetic data, moreover, was created to include 10 different message IDs, with each ID having a different number of signals and varying frequency noise. The synthetic data was created to be similar to the real data, with a total of 20 signals. The real data was not made publicly available, and the synthetic data can be found in [139]. Both real and synthetic data sets are divided into six subsets, with each subset representing normal traffic data or a specific attack. For the **types of security attacks**, the authors focused on five types of attacks, specifically, plateau, continuous change, playback, flooding, and suppress attacks. A plateau attack occurs when a single signal is overwritten to a constant value over some time, resulting in a freeze of the signal. In a continuous change attack, a signal is subtly manipulated, causing it to gradually diverge from its accurate value. This assumes that the attacker wants to set a signal to a particular value while attempting realistic small changes in the signal to fool the IDS. On the other hand, in the playback attack, a signal value is overwritten over some time with a recorded time series of values of that signal to trick the IDS. The flooding attack consists of overloading the CAN bus by sending messages of a particular ID with high frequency to the CAN bus. Lastly, the suppress attack consists of preventing the ECU from sending messages over a period of time, resulting in the absence of some particular ID in the CAN traffic data.

Model Architecture: The proposed unsupervised CANet model is based on an independent LSTM input model for each CAN ID, allowing it to capture the temporal dynamics of the corresponding signals. The output of all input models is then aggregated and fed into a fully connected subnetwork with an autoencoder structure. This enables the network to take the interdependencies of signals of all IDs into account. At each point in time, all potential input signals are reconstructed. The reconstruction error between the true signal values and their reconstruction can be used to calculate the anomaly score.

For each layer, the input and output values varied based on h_{scale}, varying from 5 to 20. raises concerns about whether a simpler model could achieve similar results with less computational cost.

System Information and Evaluation Metrics: The code for training and evaluating the proposed unsupervised CANet model was written in Python 3.5.3 with pyTorch 0.3.0 [140]. All computations were performed on a 3.5 GHz system with 4 cores and 32 GB of installed physical memory (RAM). The authors evaluated their proposed unsupervised CANet model by using the Area Under the Curve (AUC) value of the Receiver Operating Characteristic (ROC) curve, with a focus on the True Positive Rate (TPR) and True Negative Rate (TNR). The model was also compared against two baseline approaches, notably a Predictive Baseline [141] and an Autoencoder baseline [142]. In summary, despite the high performance of the proposed CANet model, there is a lack of discussion on the feasibility and real-time deployment of the model on vehicles.

6.1.1.2 Category-2 Models: These are models whose existing ML architecture was not modified but used as is by the authors. These models are commonly known as baseline models that are used for classification purposes. The following models, also referred to as Cat-2, were used in our comprehensive analysis.

a. Can-Train-and-Test: A Curated Dataset for IDS [133]: The authors developed a comprehensive dataset to overcome the existing dataset access limitations when developing IDS for the CAN bus in IVN.

Types of Security Attacks and Dataset: The proposed dataset was established to allow for both traditional and deep learning IDS training. It was collected live on 4 vehicles from two manufacturers and different car models, with 6 drivers varying in age and gender. This allowed for a more comprehensive dataset to help develop generalizable IDS

models. Additionally, the dataset was labeled to facilitate both supervised and unsupervised learning, capturing information such as the timestamp, the arbitration ID, and the data field. The dataset was collected during 10 different scenarios, an attack-free scenario and 9 attack scenarios; the dataset is available in [143]. The dataset focused on nine **types of security attacks**: Denial of Service (DoS), fuzzy, Interval, standstill, systematic, and spoofing (speed spoofing, RPM, gear, and combined spoofing) attacks. The spoofing attack consists of sending fraudulent messages to manipulate the vehicle's components and make them behave based on the false injected data. Additionally, in the interval attack, false messages are also sent to the CAN bus, and if timed correctly, can spoof the vehicle's gear, RPM, speed, or others. The standstill attack consists of sending forged messages to the CAN to put the vehicle in a stopping or standstill mode. In the systematic attack, the attacker will send messages that are altered in a structured way to the CAN bus to either replay specific messages or disable target features like traction control or airbags.

Model Architecture: Python scripts were developed using libraries like Pandas and scikit-learn to implement ML supervised and unsupervised detection algorithms. For the supervised ML algorithms, KNN, Naive Bayes, Support Vector Machine (SVM) Regression, and Tree algorithms were implemented. Local Outlier Factor (LOF) was the implemented unsupervised ML algorithm. Likewise, clustering models, K-means, and BIRCH were implemented by the authors. For each model, different parameters were used, and those with the highest performance were saved; the models were also trained and tested on the data subsets from different vehicle models.

System Information and Evaluation Metrics: To train the models, a compute cluster of one CPU with 16GB of RAM and a time limit of 6 days was allocated. The implemented models were compared against each other regarding their accuracy, precision, recall, F1 score, training, and testing time. The models performed differently depending on the

training and testing data subsets, but overall, Logistic Regression (Log R), Naive Bayes (NB), and MLP seemed to perform best. In summary, the proposed models were effective at detecting the different attacks present in the dataset.

b. KNN and SVM classification for IDS [132]: The authors used ML classification and clustering techniques- K-Nearest Neighbors (KNN) and Support Vector Machine (SVM) to analyze the offset ratio and time interval between message request and response in the CAN and, therefore, identify potential intrusions.

Types of Security Attacks and Dataset: The authors focused on two attacks: Denial of Service (DoS) and fuzzy attack. In the DoS attack, the attackers inject a large amount of malicious messages to disrupt the CAN bus availability, thereby causing service disruption. A fuzzy attack occurs when a random CAN ID with arbitrary values is injected into the CAN bus for disruption. The **dataset** used to train the models was provided by the Hacking and Countermeasure Research Lab (HCRL) [135]. It was collected from real vehicles by connecting CAN traffic by the OBD-II port and performing message injection attacks. The datasets for the two attacks contain 300 intrusion message injections, with each intrusion achieved from 3 to 5 seconds. The DoS dataset has a total of 3,665,771 numbers of messages, and the fuzzy dataset contains 3,838,860 messages. The dataset was also preprocessed before training for feature extraction.

Model Architecture: The proposed models, kNN and SVM, from Python scikit-learn library were used. The specific parameters for each model were not discussed by the authors. The models were trained with a training vs testing ratio of 70:30, 80:20, and 90:10, with the latter giving the best accuracy, over 97% for both models.

System Information and Evaluation Metrics: The authors did not provide information about the system used for their implementation. The metrics used to evaluate the models were the accuracy, F1-score, precision, and recall values. The results showed that the

accuracy for both models increases as the training set size increases as well. However, the F1 score for the SVM model stays the same for varying training set sizes, while the F1 score of the kNN model still increases.

c. Decision Tree and Multi-Layer Perceptron (MLP) [128]: The authors proposed an ensemble model, consisting of a decision tree and Multi-Layer Perceptron (MLP), to detect intrusion in the CAN bus of a vehicle in an In-Vehicle Network (IVN). These models were chosen due to their ability to detect intrusion faster compared to other ML models.

Type of Security Attacks and Dataset: The authors focused on these two attacks: Denial of Service (DoS) and fuzzy attack. The attack **dataset** used for the proposed IDS model was obtained from the Hacking and Countermeasure Research Lab (HCRL) [135]. The fuzzy and DoS attack datasets were compiled into a single file, pre-processed, and then used to train the ML algorithm classifiers. The 70:30 ratio was used for the dataset split, 70% for training the model and 30% for testing the model.

Model Architecture: Python libraries (Pandas, Matplot, and Sci-kit Learn) and TensorFlow were used to develop the ML algorithms, and the results of these ML algorithms were evaluated using the following metrics: confusion matrix (true positive rate, true negative rate, false positive rate, false negative rate), accuracy, precision. For the MLP algorithm, these parameters were used: 0.01 learning rate for 50 iterations and a size of 120 layers. These parameters were used for the Decision Trees algorithm. The random state was 0, with 5 sets as the maximum depth and 7 sets to be the number of leaves. The authors chose these parameters because they gave the best accuracy score for detecting intrusion in CAN messages.

System Information and Evaluation Metrics: The authors did not provide the system information used to train the model. To evaluate the performance of their proposed models,

the authors used the confusion matrix - true positive, true negative, false positive, and false negative. Based on these metrics, the proposed models achieved a 100% accuracy score.

In Table 6.1, all the discussed models (Cat-1 and Cat-2 models) are summarized by the Machine Learning (ML) algorithms used, specific security attacks considered by the authors, the evaluation metrics used to assess the performance of the ML models and the availability of the dataset used to train and test the ML model. These models were used for our comprehensive analysis to evaluate the deployment feasibility of ML anomaly-based IDS models in a vehicle in an In-Vehicle Network (IVN). The following section elaborates in detail on the feasibility metric, CFS proposed to evaluate models' feasibility in real-world deployment.

6.1.2 Proposed Model Feasibility Metric

The increased usage of ML models in a resource-constrained environment requires effective methods to evaluate their deployment feasibility. We propose a detailed deployment feasibility metric, which we refer to as the Custom Feasibility Score (CFS), that uses standard ML performance metrics (e.g., accuracy and inference time) and attributes relative to its resource consumption (memory and CPU usage), as well as attributes such as available CPU cores, CPU clock speed, required accuracy for the application, and anticipated RAM availability. The introduction of CFS shows that a model's deployment feasibility is significantly shaped not only by its performance but also by the hardware and operational context of its target deployment environment. Therefore, introducing this metric provides different perspectives in analyzing the ML anomaly-based IDS models' feasibility. The following subsections outline the proposed feasibility metric CFS, for assessing the deployment feasibility of ML anomaly-based

Existing Literature	Number of Models	Proposed ML Algorithm	Security Attacks Targetted	Evaluation Metrics	Dataset Availability
LSTM-based IDS [126]	1	Vanilla LSTM, Stacked LSTM	Denial of Service (DoS), Fuzzing, Spoofing	Accuracy, AUC ROC curve, Detection rate, F1 score	No
CANTransfer Model [77]	1	Convolutional LSTM Model and One-Shot Transfer Learning	Denial of Service (DoS), Fuzzing, Spoofing	Precision, Recall, F1 score	No
LeCun Network Model [127]	1	Deep transfer learning-based LeCun network (P-LeNet) model	Flooding, Fuzzing, Spoofing	Accuracy, Precision, Recall, F1 score	Yes
VGG16 [129]	1	CNN-based model, VGG16	Denial of Service (DoS), Fuzzy, Spoofing attack (on driving gear and RPM gauge)	Accuracy, Precision, Recall	Yes
Convolutional Neural Network (CNN) Model [131]	1	Convolutional Neural Network (CNN) trained on generated Recurrence Plots (RPs)	Denial of Service (DoS), Fuzzy, Spoofing attack (on driving gear and RPM gauge)	Accuracy, AUC ROC curve	Yes
Unsupervised CANet Model [130]	1	Unsupervised Deep Learning model	Plateau, Continuous change, Playback, Flooding, Suppress attacks	AUC ROC curve (focused on True Positive Rate (TPR) and True Negative Rate (TNR))	Yes
Can-Train-and-Test [133]	10	KNN, Naive Bayes, K-means, BIRCH, SVM Regression, Tree algorithms, Local Outlier Factor (LOF)	Denial of Service (DoS), Fuzzy, Interval, Standstill, Systematic, and Spoofing (Speed spoofing, RPM, Gear and Combined Spoofing) attacks	Accuracy, Precision, Recall, Training time, Testing time, F1 score,	Yes
KNN and SVM classification for IDS [132]	2	K-Nearest Neighbors (KNN) and Support Vector Machine (SVM)	Denial of Service (DoS), Fuzzing	Precision, Recall, Accuracy, F1 score	Yes
Decision Tree and Multi-Layer Perceptron (MLP) [128]	2	Decision Tree and Multi-Layer Perceptron (MLP)	Denial of Service (DoS), Fuzzy attack	Confusion matrix: True Positive (TP), True Negative (TN), False Positive (FP), False Negative (FN)	Yes

Table 6.1: Existing ML Anomaly-based IDS Models discussed in Section 6.1.1. This table shows the different models used for the comprehensive analysis.

IDS in IVNs. Additionally, the subsections provide insight into the normalization process applied to the attributes used in the CFS metric.

6.1.2.1 Normalization Process In the proposed deployment feasibility metric, we considered various factors that have different units and scales because they are collected from diverse sources. This can lead to skewed results, where some factors may be overemphasized while others are underemphasized. Consequently, finding an optimal trade-off of the different factors becomes challenging, and interpreting the results can be difficult, undermining the metric' significance. Therefore, there is a need for normalization. The application of normalization ensures that each factor is assigned a weight based on its importance rather than its scale, allowing for a fair comparison among the different attributes considered. Existing normalization techniques, such as min-max and Z-score [144], may not be applicable for several reasons. First, the attributes considered in the CFS have different scales: accuracy can range from 0 to 100, or 0 to 1 for the F1 score, while execution time can vary from seconds to several hours. This disparity makes existing normalization techniques unsuitable as they assume features are on a similar scale. Additionally, CFS are non-linear due to the exponentiation of their parameters, and normalizing them linearly may distort the relative importance of the attributes involved.

To address these challenges, we normalized each attribute individually. The normalization process applied for each attribute can therefore be defined as a ratio between the model's performance on the training and testing hardware and the expected model's performance on the vehicle CAN bus once deployed. The expected values of model's performance on the CAN bus were determined using estimated values that represent the existing resource information on the vehicle. These estimates were derived to simulate the practical deployment of the models in vehicles. We utilized these values to normalize

the different attributes in CFS, thereby illustrating the practical impact of these values on overall resource consumption in an actual vehicle within an IVN. This approach leads to a standardized method for normalizing each attribute of the proposed metric. We estimated three values, minimum, average, and maximum, for each attribute. To make these estimates, we researched resource information for vehicles produced by global automakers such as Toyota, General Motors, Audi, and Rivian. This research allowed us to identify resource specifications from hardware vendors like Nvidia [145], Tesla [146], Samsung [147], and NXP [148]. Based on these specifications, we estimated the RAM resource to be *128MB as the minimum, 512MB as the average, and 4GB as the maximum value*. While there are vehicles that feature more than 4 GB of RAM, we set the maximum value at 4 GB, as this seems to be the most common value among the vendors we considered. Additionally, some most recent specifications depicted the use of hardwares with RAM greater than 4GB. However, when assuming similar deployment conditions, once a model is feasible on 4GB RAM hardware, it should be feasible on hardware with higher RAM. Moreover, the minimum, average and maximum values were also estimated for the other parameters contributing to each attribute in the CFS metric. The following subsections provide details about the proposed feasibility metric and the additional estimated parameter values used to normalize their attributes.

6.1.2.2 Custom Feasibility Score (CFS) We further introduced the CFS, a more detailed metric that also considers the characteristics of the deployment environment, to assess ML anomaly-based IDS models' feasibility when deployed on an IVN. This metric considers the different factors related to the testing of a ML model, specifically the testing time and size of dataset, as well as the target deployment environment resources such as memory and processing power. The formula is defined as shown in eq. 6.1.

$$CFS = \omega 1 \times \delta_{norm} - \omega 2 \times \sigma_{norm} - \omega 3 \times \nu_{norm} - \omega 4 \times \Psi_{norm} \tag{6.1}$$

where δ_{norm}, σ_{norm}, ν_{norm}, and Ψ_{norm} respectively represent the normalized accuracy, normalized memory usage, normalized CPU usage, and normalized throughput. Additionally, $\omega 1$, $\omega 2$, $\omega 3$, and $\omega 4$ are weights that give importance to their associated attributes. The $\omega 1$ weight determines the importance of the accuracy of the model in the deployed environment, while $\omega 2$ determine the importance of the normalized memory usage, , $\omega 3$, the importance of the CPU usage, and $\omega 4$, the importance of the throughput of the model relative to the target host available memory, CPU and throughput respectively. To determine the feasibility score in eq. 6.1, the resource usage and throughput are deducted from the accuracy terms; for example, for a model with accuracy (δ_{norm}) equals to 1, with all weights ($\omega 1$, $\omega 2$, $\omega 3$, and $\omega 4$) equal to 1, the model is most feasible when close to 1 and not feasible when negative. Therefore, a model with high accuracy (δ_{norm}), high resource usage ($\sigma_{norm}, \nu_{norm}$), and low throughput ($\Psi_{norm}$) will be penalized more and therefore considered less feasible than a model with high accuracy (δ_{norm}) and only high resource usage ($\sigma_{norm}, \nu_{norm}$). Table 6.2 summarizes the weights assigned to the attributes in eq. 6.1 and highlights their significance. Further details on the selection of these weights are provided in the next section to ensure that no single attribute dominates the others when calculating the feasibility score. The following subsections detail the normalization process applied to normalize each attribute (δ_{norm}, σ_{norm}, ν_{norm}, and Ψ_{norm}) in eq. 6.1.

Normalized Accuracy: ML or Deep Learning (DL) models have different accuracies. The accuracy of a model can vary based on the model's architecture or the training dataset. Nonetheless, not all accuracies can be considered good enough, especially for critical applications like IDS in the CAN bus. For IDS, it is therefore crucial to evaluate the model's accuracy (δ_{model}) against a predefined standard or expected accuracy. The

CFS Attribute	Weight	Weight Significance
δ_{norm}	$\omega 1$	**Accuracy's weight:** to prioritize in safety-critical environments
σ_{norm}	$\omega 2$	**Memory usage weight:** to prioritize in resource constraint environments
ν_{norm}	$\omega 3$	**CPU usage weight:** to prioritize in resource constraint environments
Ψ_{norm}	$\omega 4$	**Throughput weight:** to prioritize for real-time deployment scenarios

Table 6.2: Significance of the Weights of the CFS Attributes. This table gives a detailed description of the weights for each attribute, considering a real-world deployment scenario.

expected accuracy ($\delta_{expected}$) can be defined as the minimum acceptable accuracy of a functional IDS on a vehicle and can be on a scale from 0 to 1 or from 0 to 100, depending on the accuracy scale of the trained model. The normalized accuracy, given the expected accuracy of an IDS on a vehicle, is expressed as defined in (eq. 6.2).

$$\delta_{norm} = \frac{\delta_{model}}{\delta_{expected}} \tag{6.2}$$

The normalized accuracy (eq. 6.2) is an appropriate ratio-based normalization of the accuracy, and will allow a fair comparison of different models given a standard value. Regardless of the model, a normalized accuracy >=1 would mean the model satisfies or exceeds the expectations or does not otherwise. One may question the effectiveness of other normalization methods like the min-max normalization; however, using an expected standard accuracy makes the normalization directly interpretable and relative to the target deployment environment, making this applicable to domains other than the automotive domain. The details about the attributes used to normalize the accuracy can be found in Table 6.3.

Normalized Memory and CPU Usage: When training and testing a model, the resource usage mostly depends on the model complexity and the training or testing size. Memory and CPU usage are important factors to consider, especially for

Attribute	Description	Unit
δ_{norm}	Normalized form of the model's accuracy given the expected accuracy on the vehicle	-
δ_{model}	Model's accuracy when tested: F1 score, model accuracy, or any metric relevant to the analysis. The F1 score was considered for the CFS computations.	-
$\delta_{expected}$	The expected accuracy on the vehicle: dependent on the expected security level; with 100 or 1 representing a perfect IDS accuracy	-

Table 6.3: Details on the Normalized Accuracy Terms for the CFS Metric. This table provides detailed information about the parameters considered in normalizing the accuracy of an ML model.

resource-constrained environments. Complex models like DL models trained in traditional computing environments might not be deployable in resource-constrained environments. Additionally, analyzing resource usage alone can be misleading unless placed in the context of what is available in a resource-constrained environment. In the context of vehicles in an IVN, evaluating the models' resource usage relative to the available resources will help analyze the feasibility of those models on the vehicles. The normalized memory and CPU attributes can be found in eq. 6.3 and 6.4 respectively.

For the **normalized memory** (σ_{norm}) in eq. 6.3, σ_{model} represents the total RAM used by the model during testing, and $\Gamma_{testdatasize}$ represents the test dataset size in bytes. Therefore, $\frac{\sigma_{model}}{\Gamma_{testdatasize}}$ accounts for how much RAM the model uses per unit of test data size, ensuring that large or small test datasets do not bias the memory usage evaluation and normalization. Moreover, σ_{cav} represents the total available RAM on the autonomous vehicle, $\Gamma_{expdatasize}$, the size of the traffic data packet the IDS processes on the vehicle at a given timestamp, and ε, the maximum percentage of available RAM for IDS allowing other vehicle's processes to still function properly. Therefore, $\frac{\sigma_{cav}*\varepsilon}{\Gamma_{expdatasize}}$ accounts for the available RAM per traffic data packet for IDS on the vehicle. Consequently, σ_{norm} represents the normalized memory usage of the model per unit of test dataset, given the expected memory usage for a traffic data packet. A summary of the description and unit of each term for the normalized memory (eq. 6.3) is provided in Table 6.4.

$$\sigma_{norm} = \frac{\sigma_{model} \times \Gamma_{expdatasize}}{\Gamma_{testdatasize} \times \sigma_{cav} \times \varepsilon} \tag{6.3}$$

Attributes	Description	Unit
σ_{norm}	Normalized form of the model's memory usage on testing, relative to the available memory for IDS purposes on a vehicle	-
σ_{model}	RAM used by the model during testing time	bytes
σ_{cav}	Total available RAM memory on the vehicle	bytes
ε	Percentage of available RAM for IDS purpose relative to the overall RAM on the vehicle	percent
$\Gamma_{testdatasize}$	Size of the dataset used to test the model	bytes
$\Gamma_{expdatasize}$	Traffic packet size processed by the CAN in real-time	bytes

Table 6.4: Details on the Normalized RAM Memory Terms for the CFS Metric. This table provides detailed information about the parameters considered in normalizing the memory usage of an ML model.

Normalizing the CPU usage (ν_{norm}) (eq. 6.4) helps to compare the computational resources needed for the model to perform under testing with the available processing resources of the target host on the vehicle. The computational resources are not limited only to the CPU usage in percent, as the number of cores and the CPU clock speed also play a crucial role in the number of cycles or computations a processor can execute per second. The $\nu_{testused}$ represents the raw CPU usage during the model testing execution, and using the $\nu_{testused}$ alone to compare the model's performance across different environments does not result in an objective result, as explained above. Therefore, the $\nu_{testused}$ is scaled by the clock speed ($\nu_{testspeed}$) and the number of available cores ($\nu_{testcores}$) on the system used to train the model. Additionally, models can be trained on different datasets with various testing sizes, and when deployed, they process CAN data traffic packets one at a time. Therefore, $\Gamma_{samplesdatasize}$ provides the number of samples present in the testing set and helps normalize the overall computational workload given the number of samples, resulting in the model's computational workload for one sample. Similar parameters are considered

to determine the vehicle's available computational resources, with $\nu_{cavcores}$, $\nu_{cavspeed}$ and $\nu_{cavused}$ representing the number of CPU cores on the vehicle, their clock speed and the maximum percentage of CPU the IDS can use while still allowing other vehicle's processes to function properly. Considering all these factors provides a more valuable way to obtain a normalized CPU resource usage, independently of the dataset size or the resources used to train and test the model. A summary description of all the terms used for eq. 6.4 and their units is available in Table 6.5.

$$\nu_{norm} = \frac{\nu_{testused} \times \nu_{testcores} \times \nu_{testspeed}}{\Gamma_{samplesdatasize} \times \nu_{cavcores} \times \nu_{cavspeed} \times \nu_{cavused}} \tag{6.4}$$

Attribute	Description	Unit
ν_{norm}	Normalized form of the model's CPU usage when testing given the available CPU resources for IDS purpose on a vehicle	-
$\nu_{testused}$	CPU usage by the model during testing time	percent
$\nu_{testcores}$	Total number of CPU cores on the resource device used to test the model	-
$\nu_{testspeed}$	CPU clock speed of the resource device used to test the model	GHz
$\Gamma_{expdatasize}$	Expected size of the data traffic packet processed by the CAN in real-time	bytes
$\Gamma_{samplesdatasize}$	Number of samples elements processed by the model during testing	-
$\nu_{cavcores}$	Number of expected logical CPU cores on the vehicle resource hosting the IDS	-
$\nu_{cavspeed}$	CPU clock speed on the vehicle resource hosting the IDS	GHz
$\nu_{cavused}$	Expected CPU usage for an IDS on a vehicle resource	percent

Table 6.5: Details on the Normalized CPU Terms for the CFS Metric. This table provides detailed information about the different parameters considered in normalizing the CPU usage of a ML model.

Normalized Throughput (Ψ_{norm}) : The throughput is a valuable metric to consider when analyzing the effectiveness of an IDS on a vehicle. It refers to the amount of network traffic an IDS can process and analyze within a given time. An efficient IDS should be able to accurately identify intrusions in incoming traffic data as it is processed in real time by the CAN without causing any delay for other processes in the vehicle. To determine a model's throughput when testing on a dataset, the test dataset size is divided by the total

testing time taken by the model to process it. $\Gamma_{testdatasize}$ represents the total size in bytes of the test dataset, and $\rho_{modeltesttime}$ provides the time duration required by the model to process the test dataset in seconds. The model's throughput is then compared against the expected minimum throughput of an IDS on the vehicle $\Psi_{cavthroughput}$ (in bytes per seconds) while still allowing other vehicle systems to remain functional. The result is then inverted, ensuring high throughput is not penalized, but low throughput is, as defined in eq. 6.5. A result over 1 would imply the model is too slow and might interfere with other vehicles' functions once deployed in real-time. A summary of the attributes used in eq. 6.5 is provided in Table 6.6.

$$\Psi_{norm} = \frac{\rho_{modeltesttime} \times \Psi_{cavthroughput}}{\Gamma_{testdatasize}} \tag{6.5}$$

Attribute	Description	Unit
Ψ_{norm}	Normalized form of the inverse of the model's throughput when testing, given the expected IDS throughput on a vehicle	-
$\rho_{modeltesttime}$	Model's time execution on test dataset	seconds
$\Gamma_{testdatasize}$	Dataset size used to test the model	bytes
$\Psi_{cavthroughput}$	Expected throughput for an IDS on a vehicle	bytes per seconds

Table 6.6: Details on the Normalized Throughput Terms for the CFS Metric. This table provides detailed information about the different parameters considered in normalizing the throughput of an ML model.

To normalize the CFS attributes mentioned above, minimum, average, and maximum values of each parameter regarding the CAN bus resources were estimated, as established in Section 6.1.2.1. To normalize CFS attributes, estimated values were determined from the existing literature. In [149], different CAN types were described from the initial CAN designs (the standard or classical CAN with lower speed) to the CAN FD (Flexible Data-rate) with higher speed and the CAN XL with increased payload, speed, and reliability

for future high-data applications. The values of the additional parameters used to normalize the CFS attributes in Table 6.7 have been determined based on the existing literature on the standard CAN and the CAN FD. Moreover, the International Organization for Standardization (ISO) recommends prioritizing the adoption of the CAN FD compared to the standard CAN [150]; regulations about the usage of the CAN XL were not found, however. Additionally, [151] [149] elaborated on the CAN FD throughput being 3 to 8 times higher than the throughput on the standard CAN. Therefore, the throughput value estimated in Table 6.7 was derived from the expected throughput on the CAN FD.

CFS attribute	**Resource Information**	Estimated Values			**Unit**
		Minimum	**Average**	**Maximum**	
$\Gamma_{expdatasize}$	Size of a CAN data frame message [152] [151]	8	32	64	Bytes
σ_{cav}	Available RAM on the vehicle [148]	128	512	4e+9	Bytes
ε	Maximum allowed RAM usage for IDS	0.05	0.08	0.10	Percent
$\nu_{cavcores}$	Number of CPU logical cores on the CAN bus [148] [145]	1	4	8	-
$\nu_{cavspeed}$	CPU clock speed on the CAN bus [148] [146]	1.2	1.6	2.6	GHz
$\nu_{cavused}$	Maximum allowed CPU usage for IDS	5	8	10	Percent
$\Psi_{cavthroughput}$	Minimum throughput for IDS on a vehicle [151]	0.12e+6	0.43e+6	1.0e+6	Bytes/s
$\omega 1, \omega 4$	Attributes' weights for the accuracy and throughput	1	-	2	-
$\omega 2, \omega 3$	Attributes' weights for the memory and CPU usage	0.5	-	1	-

Table 6.7: Estimated values used to normalize the CFS attributes. These values were considered as the actual resource information present in a vehicle in an IVN to normalize the CFS attributes in eq. 6.1.

For the other attributes like the RAM, the number of cores of the CPU and the clock speed, specific resource specifications for vendors established in Section 6.1.2.1 were considered.As no concrete source was found, an estimation was provided for the maximum allowed RAM (ε) and CPU ($\nu_{cavused}$) usage for IDS purposes, still allowing the other processes in the vehicle to function adequately. For $\omega 1, \omega 2, \omega 3$, and $\omega 4$, the values were determined based on the importance of each metric in the CFS formula. For example, in an IVN, the accuracy of an IDS is of utmost importance to guarantee the security and

safety of the people involved, as well as the throughput. Therefore, $\omega 1$ and $\omega 4$ are set to vary between the range 1 and 2, while the weight for the resource usage ($\omega 2$ and $\omega 3$) will vary between 0.5 to 1. This ranges of values were chosen because going below 0.5 or above 2.0 would lead to a skewed optimization, where the effect of some attributes could dominate others, like the accuracy (δ_{norm}), dominating over the resource usage ($\sigma_{norm}, \nu_{norm}$). Moreover, the weights of the accuracy and throughput are higher than those of the resource usage, as the precision and speed of an IDS when processing incoming traffic can be considered crucial to the IVN security. A summary of the CFS attributes, their estimated values, and sources is provided in Table 6.7.

6.2 Simulation and Results

The simulations were done in two distinct steps. First, we replicated the twenty proposed ML models in Table 6.1, listed in Section 6.1.1, to simulate current state-of-the-art ML anomaly-based IDS models that can be deployed in a vehicle in an In-Vehicle Network (IVN). *It should be noted that the goal of our simulations is not to replicate the accuracy scores obtained by the authors, but to implement the proposed models, as these models were evaluated for deployment feasibility*. The models were evaluated using the accuracy score, precision, recall, F1 score, testing time, CPU usage, and RAM usage. Consequently, we used the proposed metric, to calculate its deployment feasibility score using CFS in eq. 6.1. Additionally, we present the results we obtained from our simulation, the implementation of the models, and the deployment feasibility scores of these models in this section.

6.2.1 Simulation Setup

This section presents details on the simulation set-up used to implement the proposed ML anomaly-based models, factors that influenced these models' performance, and metric considered to calculate the deployment feasibility score of these models.

System Information: These simulations were carried out on two different platforms - a platform with 16GB RAM, a 13th Gen Intel(R) Core(TM) i9-13900H 2.60 GHz computer with Windows 11 as the operating system and 20 logical CPU cores, and a platform with 8GB RAM with 4 logical CPU cores, 2.2 GHz CPU clock speed on Google Collab. All simulations were conducted using Python Notebooks from Jupyter and Google Colab. The details of the code base of this implementation can be found in [153].

Dataset and Security attacks: In replicating the ML anomaly-based IDS models in Section 6.1.1, not all authors made their training datasets publicly available, as shown in Table 6.1. Therefore, for models with inaccessible datasets, we used publicly available datasets in [78] [137]. These datasets were chosen as they were closely related to the dataset generated by the authors. Additionally, these datasets were generated from vehicles for the specific attack scenarios considered by the authors. Furthermore, data preprocessing activities were carried out: removing duplicate and missing values, converting the CAN bus hexadecimal values to the specified input type (as defined by the authors), and encoding the target features. Once the dataset was preprocessed, it was split using the specified ratio defined by the authors as listed in Section 6.1.1 and then used to train and test the model. Regarding the **security attacks**, the simulated models were trained and tested to detect the following security attacks: denial-of-service (DoS), fuzzy, and spoofing attacks. These attacks were considered because the dataset used for model implementation by the authors was generated for these attacks, as shown in Table 6.1.

Metrics for evaluating the performance and deployment feasibility of the ML anomaly-based IDS models: To evaluate the performance of the simulated ML anomaly-based IDS models, standard ML model performance metrics (accuracy, precision, recall, and F1 score) were used. To evaluate the computational resources used by these models, metrics like the CPU usage and the RAM usage on the test dataset were considered. In our simulation, we only considered the test dataset because we assumed the models that will be deployed in the vehicle are only carrying out offline computation. Hence, no training will be conducted in real-time after being deployed in the vehicle. Additionally, the test dataset depicts the data that will be analyzed by the models in real-time to make a prediction. The CPU usage and memory usage are, respectively, the proportion of the CPU's processing power and RAM the model uses when processing the test dataset and were determined using functions from the psutil Python library. The σ value was measured in bytes, and ν and δ were measured in percentage. We applied estimated values, listed in Section 6.1.2, to normalize the values of τ and σ.

Considered scenarios: In our simulation, we considered three different scenarios to represent practical deployment approaches for each of the proposed metric. Given that three different values were estimated to represent the *minimum, average, and maximum* resources in a vehicle, as mentioned in Section 6.1.2.1, we combined the different values of the RAM usage and execution time, such that high RAM and CPU usage should equate to a lower execution time, as well as, low RAM and CPU usage should equate to a higher execution time. However, in the CFS metric, we did not consider the scenario of high RAM and CPU usage resulting in a lower execution time and vice versa, because increasing the RAM and the CPU resources for an IDS model is less probable to result in the IDS model's throughput reducing compared to when run on a system with lower RAM and CPU resources, and vice versa. Therefore, we considered the minimum, average, and maximum estimated values of

the attributes in the CFS. The results obtained from the implemented ML anomaly-based IDS models and their calculated feasibility scores, are presented in the following section.

6.2.2 Results and Discussions

We present the results obtained for the implementation of the ML anomaly-based IDS models listed in Table 6.1, as defined in Section 6.1.1, for Cat-1 and Cat-2 models, in this section. These models were evaluated using the metrics (accuracy, resource utilization (RAM and CPU), and testing time) defined in Section 6.2.1. Additionally, we present the deployment feasibility scores of these models using the CFS metric defined in Section 6.1.2.

Standard ML model performance evaluation: For the accuracy, precision, recall and F1 score, the results obtained from our simulation differ from the results stated in the existing literature by the authors due to the following factors: the dataset used, dataset size, and differences in simulation environment, as shown in Table 6.8 and Table 6.9. For some of the models (LSTM-based IDS [126] and CANTransfer model [77]), the datasets used by the authors were not made publicly available. Therefore, we had to use other datasets [78, 137], and this affected the performance of the models. Additionally, due to the difference in the simulation environment in terms of GPU/CPU, number of cores, and the processor speed, the number of samples we considered was reduced to fit our simulation environment. Datasets with more than 1 million samples were reduced by 25%. The data size reduction was done by reducing the individual samples by 15% for each attack category to ensure the dataset was balanced. Therefore, this affected the results, as a reduced number of samples was used to train and test the ML model.

Resource performance evaluation of ML model: The evaluation of the CPU and RAM usage helped measure the resources used by the ML model when making a prediction

ML Model	Accuracy (%)	Precision	Recall	F1 Score	Testing Time (seconds)	CPU Use (%)	RAM Use (MB)	Test Set Size (MB)	CPU Cores	CPU Clock Speed (GHz)
LSTM-based IDS [126]	50.20	0.50	1.0	0.67	0.0026	0.3	10217.44	4.12	20	2.6
CANTransfer Model [77]	81.93	0.38	0.04	0.75	268642.9	2.0	13960.79	1.80	20	2.6
LeCun Network Model (P-LeNet) [127]	83.52	0.69	0.83	0.76	507503.5	1.8	11485.12	12.49	20	2.6
VGG16 [129]	66	0.66	0.66	0.66	3.91	20.8	7386.82	65.47	20	2.6
Convolutional Neural Network (Rec-CNN) [131]	100.0	1.0	1.0	1.0	31.3	98.1	5.42	112.72	4	2.2
Unsupervised CANet Model [130]	56.0	0.37	0.64	0.47	6.98	91.1	7.42	5.45	4	2.2

Table 6.8: Performance of the simulated ML anomaly-based IDS for category-1 models, as listed in Section 6.1.1. This table provides detailed information about the performance metrics (accuracy, precision, recall, F1 score, testing time, CPU usage, and RAM usage) used to evaluate the performance of category-1 ML anomaly-based IDS models.

(intrusion detection) with the test dataset, as shown in Table 6.8 and Table 6.9. These values depend on the following factors: input data size and type, complexity of the model's architecture, and model inference time. For example, the VGG16 model [129] converted the preprocessed input data into image data and carried out a multi-classification attack detection. This requires more processing power when making predictions. Additionally, more computational resources are consumed by deep learning models like [129] [127] when making predictions. This is due to the complexity of the model's architecture. Furthermore, the longer it takes to make a prediction, the higher the resource consumption will be. These values, CPU and RAM usage, served as input for the proposed metric, CFS in eq. 6.1.

ML Model	Accuracy (%)	Precision	Recall	F1 Score	Testing Time (seconds)	CPU Use (%)	RAM Use (MB)	Test Set Size (MB)	CPU Cores	CPU Clock Speed
Random Forest (RF) [133]	100.0	1.0	1.0	1.0	0.52	53.8	19.78	17.21	4	2.2
Birch [133]	34.0	0.39	0.72	0.50	0.71	60.0	1.23	17.21	4	2.2
LOF [133]	45.0	0.46	0.98	0.63	0.021	99.1	581612.77	21.51	4	2.2
Bernoulli Restricted Boltzmann Machine (B. RBM) [133]	89.0	1.0	0.76	0.86	0.35	45.5	5.41	21.51	4	2.2
Naive Bayes (NB) [133]	100.0	1.0	1.0	1.0	0.16	31.6	42.81	21.51	4	2.2
Linear Regression (Lin R) [133]	95.0	1.0	0.9	0.95	0.13	34.1	13.36	21.51	4	2.2
Gradient Boosting (GB) [133]	100.0	1.0	1.0	1.0	0.39	47.4	32.07	21.51	4	2.2
KMeans [133]	85.0	1.0	0.68	0.81	0.09	95.6	1.19	21.51	4	2.2
MiniBatchKMeans (MKMeans) [133]	88.0	1.0	0.75	0.86	0.089	21.9	17.82	18.07	4	2.2
Logistic Regression (Log R) [133]	99.0	1.0	0.99	0.99	0.093	32.6	0.135	21.51	4	2.2
KNN [132]	100.0	1.0	1.0	1.0	596.88	47.4	11.62	21.51	4	2.2
SVM [132]	99.0	1.0	0.99	0.99	12.99	95.6	1.19	21.51	4	2.2
Decision Tree (DT) [128]	99.74	1.0	0.98	0.99	0.056	5.0	14975.16	54.19	20	2.6
Multi-Layer Perceptron (MLP) [128]	83.93	0.84	1.0	0.91	0.076	62.0	13172.78	54.19	20	2.6

Table 6.9: Performance of the simulated ML anomaly-based IDS for category-2 models, as listed in Section 6.1.1. This table provides detailed information about the performance metrics (accuracy, precision, recall, F1 score, testing time, CPU usage, and RAM usage) used to evaluate the performance of category-2 ML anomaly-based IDS models.

Execution time evaluation: The testing time in seconds represents the time duration the model takes to perform intrusion detection on the testing dataset. This value is dependent on the dataset structure and size. For example, models with a higher training

and testing set ratio, like 70:30, will have higher execution time compared to models with a lower training and testing set ratio, like 90:10. The testing time was used in the proposed metric, CFS, to compute the throughput, as defined in eq. 6.5.

6.2.2.1 Custom Feasibility Score (CFS) For the CFS metric, eqs. 6.2-6.5 were used along with the estimated values and scenarios as defined in Section 6.1.2.2 and Section 6.2.1, to normalize the performance results of the ML anomaly-based IDS Cat-1 and Cat-2 models (Table 6.8 and 6.9). Figs. 6.1-6.6 show the resulting CFS values for IDS model in Cat-1 and Cat-2. For each category, the figures show the CFS with different combinations of $\omega 1$, $\omega 2$, $\omega 3$, and $\omega 4$ for the minimum, average and maximum estimated values (Table 6.7). For example, Fig. 6.2 shows the CFS results for the Cat-1 models for all combinations of $\omega 1$ and $\omega 4$, with $\omega 2$=1 and $\omega 3$=0.5, given the average estimated values. in Table 6.7. Fig. 6.6 shows the CFS results for the Cat-2 models for all combinations of $\omega 1$ and $\omega 4$, with $\omega 2$=1 and $\omega 3$=1, given the maximum estimated values. The results for all combinations of $\omega 1$, $\omega 2$, $\omega 3$, and $\omega 4$ can be found in [153]. Moreover, the negative values for the model's CFS in all the figures have been scaled between -2 and 0 for Cat-1 and Cat-2 models for better visualization.

CFS results analysis for Cat-1 models: The results obtained in Figs. 6.1-6.3 showed that overall, the models with the CFS above 0 are feasible, while the models below 0 are not. Fig. 6.1 shows the models' feasibility scores with the minimum of all estimated values (Table 6.7), Fig. 6.2, the feasibility scores for the average values, and Fig. 6.3, the feasibility scores with the maximum values, with the weights varying between 1 and 2 for $\omega 1$ and $\omega 4$, and between 0.5 and 1 for $\omega 1$ and $\omega 2$.

One notable observation across Figs. 6.1-6.3 outlines that the models' feasibility scores are consistent across scenarios for most of them, except for the CANet, which is only

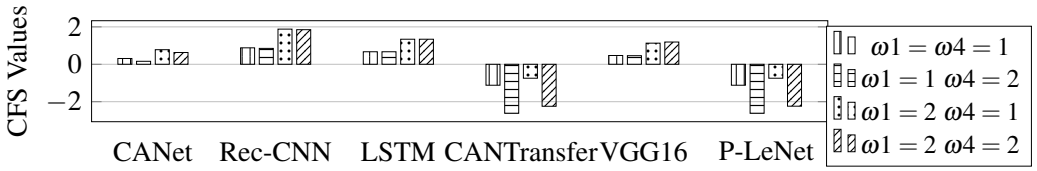

Figure 6.1: CFS values for Cat-1 models for $\omega 1$ and $\omega 4$ between 1 and 2, and fixed values of $\omega 3$=$\omega 2$=0.5, for the minimum values of the available RAM ($\sigma_{cav} = 128MB$), the usage percentage ($\varepsilon = 0.05$), the CPU clock speed ($\nu_{cavspeed} = 1.2GHz$), the number of logical CPU cores ($\nu_{cavcores} = 1$), the allowed CPU usage for IDS purposes ($\nu_{cavused} = 5\%$), the expected IDS minimum throughput ($\Psi_{cavthroughput} = 0.13MBytes/s$) and the expected size of a CAN data frame message ($\Gamma_{expdatasize} = 8Bytes$)

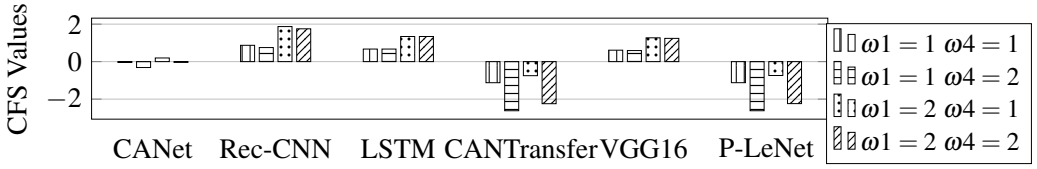

Figure 6.2: CFS values for Cat-1 models for $\omega 1$ and $\omega 4$ varying between 1 and 2, and fixed values of $\omega 3 = 0.5$ and $\omega 2 = 1$, for the average values of the available RAM ($\sigma_{cav} = 512MB$), the usage percentage ($\varepsilon = 0.08$), the CPU clock speed ($\nu_{cavspeed} = 1.6GHz$), the number of logical CPU cores ($\nu_{cavcores} = 4$), the allowed CPU usage for IDS purposes ($\nu_{cavused} = 8\%$), the expected IDS minimum throughput ($\Psi_{cavthroughput} = 0.43MBytes/s$) and the expected size of a CAN data frame message ($\Gamma_{expdatasize} = 32Bytes$)

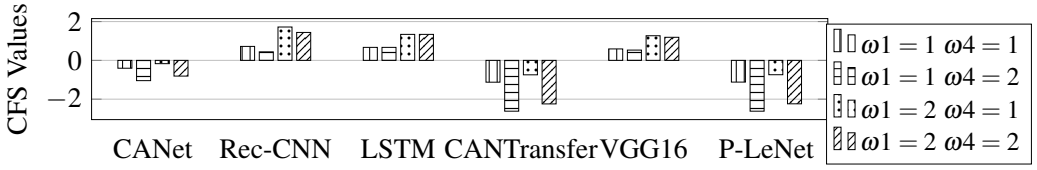

Figure 6.3: CFS values for Cat-1 models for $\omega 1$ and $\omega 4$ varying between 1 and 2, and fixed values of $\omega 3 = \omega 2 = 1$, for the maximum values of the available RAM ($\sigma_{cav} = 4GB$), the maximum usage percentage ($\varepsilon = 0.1$), the CPU clock speed ($\nu_{cavspeed} = 2.6GHz$), the number of logical CPU cores ($\nu_{cavcores} = 8$), the maximum allowed CPU usage for IDS purposes ($\nu_{cavused} = 15\%$), the expected IDS minimum throughput ($\Psi_{cavthroughput} = 1MBytes/s$) and the expected size of a CAN data frame message ($\Gamma_{expdatasize} = 64Bytes$).

feasible for the first scenario with minimum estimated values. The figures show that out of 6 models, only 3 can be considered as feasible across different environments with varying resource availability: Rec-CNN, LSTM, and VGG16. Despite being feasible, those models are not necessarily practical and suitable for real-time deployment. For example, VGG16's feasibility score highlights its low accuracy ($CFS = 0.56$ when $\omega 1 = 1$ and $\omega 4 = 1$ or $CFS = 1.19$ when $\omega 1 = 2$ and $\omega 4 = 2$, with VGG's F1-score= 0.66). This indicates that while the model appears feasible, its accuracy, particularly for IDS purposes, makes it unsuitable for real-time deployment. The same analysis can be done for the LSTM model, with a low F1-score, making the LSTM model not practical for real-time deployment. The Rec-CNN however, seems feasible and more suitable for real-time deployment, with $CFS = 0.88$ when $\omega 1 = 1$ and $\omega 4 = 1$ or $CFS = 1.85$ when $\omega 1 = 2$ and $\omega 4 = 2$. With this observation, one can therefore assume that a model is feasible when its CFS is above 0, and more practical when the CFS is as close as possible to the expected perfect accuracy (1) when $\omega 1 = 1$ and double the accuracy (2) when $\omega 1 = 2$.

Additionally, despite having the CANet model feasible in a low-resource environment (Fig. 6.1 with CFS greater than 0), the Figs 6.5 and 6.6 showed that the model slowly becomes infeasible, with CFS under 0, as the resources and throughput requirements and weights increase, with the values of $\omega 1$ and $\omega 4$ changing. This is most likely due to the model's complexity and high usage of resources; CANet uses 91.1% of CPU resources to process 5.45 MBytes of data (Table 6.8). Moreover, the CANet model's F1-score of 47% renders the model impractical, especially for detecting intrusions, even in resource-constrained environments. It is important to note that for most models in Cat-1, the accuracy achieved during our simulation was lower than that reported by the authors, for several reasons, as discussed in Section 6.2.2. However, this does not invalidate the fact that most Cat-1 models are not feasible, mostly because of their complex architecture, as

even a higher accuracy cannot guarantee the feasibility, especially if the model has a high resource usage or low throughput.

For the only model that seems feasible, Rec-CNN, the authors [131] performed a transformation of the arbitration IDs into Recurrence Plots described in Section 6.1.1. That transformation is computationally expensive and can cause the model to become infeasible in high data traffic frequency networks with a higher expected throughput. This behavior is captured in Fig. 6.3, where the CFS decreases more as the throughput is given higher importance than the accuracy ($\omega 4 >= \omega 1$) in high-resource environments, compared to resource-constrained environments (Fig. 6.1). This shows that in the last scenario (Fig. 6.3), Rec-CNN's throughput is relatively close to the expected throughput. Therefore, the model might struggle processing incoming traffic as its frequency increases in a high-resource environment.

CFS results analysis for Cat-2 models: The results obtained in Figs. 6.4-6.6 show that overall, more models are feasible across various scenarios, compared to models in Cat-1. Additionally, most models seem practical; in fact, across all figures, (Figs. 6.4-6.6), most models like Naive Bayes have their CFS always close to 1 when $\omega 1 = 1$ and always close to 2 when $\omega 1 = 2$, depicting the almost perfect accuracy of the model, with a higher throughput than expected, and low resource usage. Meanwhile, when considering a model like Birch, with the CFS always close to 0.5 when $\omega 1 = 1$ and close to 1 when $\omega 1 = 2$, we can conclude that Birch model is not practical, due to its low F1-score or accuracy. Moreover, the Birch model is infeasible across all scenarios, due to a constant pattern across them (Figs. 6.4-6.6). The figures show that the model's feasibility scores are consistent across the different scenarios (for a low or high resource environment), even when the importance of the throughput is twice as important as the accuracy ($\omega 4 = 2$ and $\omega 1 = 1$). However, when analyzing the SVM model as shown in Figs. 6.4 to 6.6, we observe that

the model's feasibility varies significantly across different scenarios, particularly based on the weights $\omega 1$ and $\omega 4$. Specifically, in a resource-constrained environment, the difference in the CFS when accuracy is prioritized ($\omega 1 > \omega 4$) compared to when it is less prioritized ($\omega 1 <= \omega 4$) is minimal (see Fig. 6.4). In contrast, in an environment with more resources, this difference becomes more pronounced, as depicted in Fig. 6.5. The discrepancy seems to continuously increase until the model becomes infeasible in specific scenarios, such as when $\omega 1 = 1$ and $\omega 4 = 2$, illustrated in Fig. 6.6. This observed pattern in the SVM model indicates that the model's throughput closely aligns with the expected throughput. However, when deployed in a resource-rich environment, where the model is expected to process a heavier data load at a faster rate, it fails to do so. This explains why prioritizing throughput with a higher weight leads to the SVM model becoming infeasible in high-resource environments.

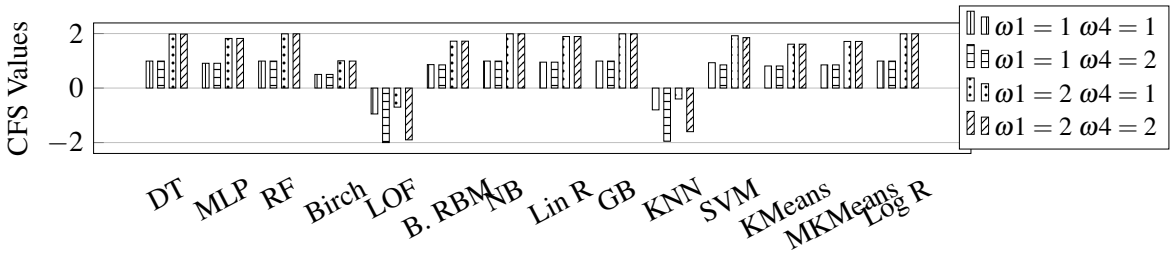

Figure 6.4: This figure shows the CFS values for Cat-2 models for $\omega 1$ and $\omega 4$ varying between 1 and 2 and fixed values of $\omega 3$=$\omega 2$=0.5, for the minimum values of the available RAM ($\sigma_{cav} = 128MB$), the usage percentage ($\varepsilon = 0.05$), the CPU clock speed ($\nu_{cavspeed} = 1.2GHz$), the number of logical CPU cores ($\nu_{cavcores} = 1$), the allowed CPU usage for IDS purposes ($\nu_{cavused} = 5\%$), the expected IDS minimum throughput ($\Psi_{cavthroughput} = 0.13MBytes/s$) and the expected size of a CAN data frame message ($\Gamma_{expdatasize} = 8Bytes$)

Fig. 6.4 shows the CFS results obtained when $\omega 1$ and $\omega 4$ vary between 1 and 2, with $\omega 2$ and $\omega 3$ equally set to 0.5 for the minimum estimated values (Section 6.2.1). The

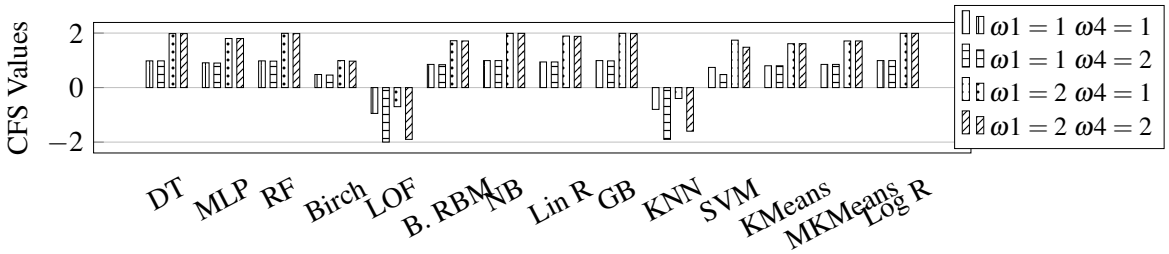

Figure 6.5: CFS values for Cat-2 models for $\omega1$ and $\omega4$ varying between 1 and 2, and fixed values of $\omega3 = 0.5$ and $\omega2 = 1$, for the average values of the available RAM ($\sigma_{cav} = 512MB$), the usage percentage ($\varepsilon = 0.08$), the CPU clock speed ($\nu_{cavspeed} = 1.6GHz$), the number of logical CPU cores ($\nu_{cavcores} = 4$), the allowed CPU usage for IDS purposes ($\nu_{cavused} = 8\%$), the expected IDS minimum throughput ($\Psi_{cavthroughput} = 0.43MBytes/s$) and the expected size of a CAN data frame message ($\Gamma_{expdatasize} = 32Bytes$)

same analysis has been done for different combinations of $\omega2$ and $\omega3$ varying between 0.5 and 1, and the results show patterns similar to Fig. 6.4 for each model, respectively, with little to no variations [153]. Given that ω_2 and ω_3 represent the weights for resource consumption (memory and CPU usage, respectively), the observation that there is little to no variation when resource usage is halved ($\omega_2 = \omega_3 = 0.5$) compared to when it is not ($\omega_2 = \omega_3 = 1$) [153] suggests that most of the proposed models are resource-efficient, utilizing significantly less than the expected resources. This finding further reinforces the choice of weight values, highlighting their importance.

Given the feasibility results of Cat-1 and Cat-2 models for the CFS metric, one may question the metric' usefulness. The next section elaborates on the rationale behibd the choices for the CFS, and its usability.

6.3 CFS rationale

In this section, we present a contrastive evaluation of the proposed deployment feasibility metric, CFS, in terms of the metric formulation and results obtained for Cat-1 and Cat-2 models. Furthermore, we present the lessons learned from evaluating the

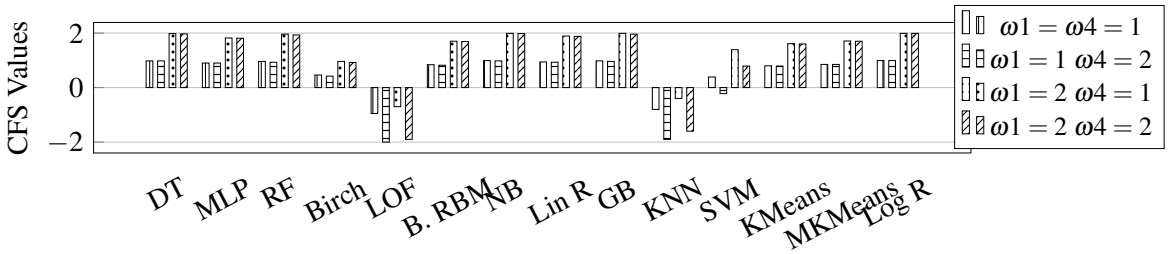

Figure 6.6: CFS Values for Cat-2 models for $\omega 1$ and $\omega 4$ varying between 1 and 2, and fixed values of $\omega 3 = \omega 2 = 1$, for the maximum values of the available RAM ($\sigma_{cav} = 4GB$), the maximum usage percentage ($\varepsilon = 0.1$), the CPU clock speed ($\nu_{cavspeed} = 2.6GHz$), the number of logical CPU cores ($\nu_{cavcores} = 8$), the maximum allowed CPU usage for IDS purposes ($\nu_{cavused} = 15\%$), the expected IDS minimum throughput ($\Psi_{cavthroughput} = 1MBytes/s$) and the expected size of a CAN data frame message ($\Gamma_{expdatasize} = 64Bytes$)

deployment feasibility of the ML anomaly-based IDS models using the proposed feasibility metric as shown in Section 6.1.2.

To design the CFS metric in eq. 6.1, we adopted a more detailed approach that includes these attributes in their normalized form: the accuracy of the ML model (δ_{norm}), RAM usage (σ_{norm}), CPU usage (ν_{norm}), and the throughput (Ψ_{norm}). We introduced the Ψ_{norm} factor that is determined by the testing time and memory usage. The values of the attributes were normalized to reflect the expected resource utilization (RAM and CPU) and the model's performance (accuracy and throughput) for an ML anomaly-based IDS model in an IVN. This comprehensive normalization for the specified attributes takes into account various factors, such as the expected traffic data packet size that the CAN bus can process at any given time, to ensure the model can manage the data quickly as required by the CAN bus. Additionally, we considered the maximum percentage of total RAM and CPU that the IDS should utilize while still allowing other vehicle functions to operate efficiently. The size of the test dataset for the model was also factored in, as well as elements contributing to the processor's computational power, such as the number of CPU

cores or processor speed, as shown in eq. [6.2, 6.3, 6.4, 6.5]. Each attribute of the CFS was assigned an importance weight ($\omega 1, \omega 2, \omega 3, \omega 4$), and we analyzed the variations in cFS for various weights, providing the feasibility of each model. The factors considered by the CFS are crucial in identifying deployable solutions, which other metrics that consider only the model's accuracy and basic resource usage might miss. Therefore, the CFS offers a stricter assessment for complex models, highlighting deployment challenges.

The next section provides a summary of the observations made from the feasibility metric, as well as the result, and provides with a guideline for the metric's usability.

6.3.1 Lessons Learned

In the deployment feasibility assessment of the Cat-1 and Cat-2 models, the following number of models were evaluated as feasible. The CFS resulted in a total of 15 feasible models, 3 of 6 Cat-1 and 12 of 14 Cat-2 models.

6.3.1.1 Model's accuracy The proposed metric CFS, is designed to assess models based on their resource utilization and latency relative to their accuracy. As a result, models with lower accuracy receive proportionately lower feasibility scores, and vice versa. This highlights the critical need for optimizing the accuracy of IDS. For real-time anomaly intrusion detection, prompt and precise identification of malicious activity is essential. Thus, a model's efficacy in differentiating between legitimate and anomalous CAN bus traffic is critical in determining the model's practical feasibility in a real-world scenario. Therefore, a model that cannot achieve an adequate accuracy level, even if it demonstrates moderate resource consumption, may be deemed impractical. This is exemplified by the Birch model within Cat-2 models in Table 6.9, Fig. 6.6. Despite its low resource consumption and execution time, the model's feasibility scores are low for the CFS because

of the model's accuracy (34%) and F1-score (0.5), rendering it unsuitable for real-world deployment.

6.3.1.2 Model architecture and computational overhead The architectural complexity of ML anomaly-based IDS models plays a crucial role in determining their computational requirements, which in turn influences their feasibility for deployment in real-world environments. Simpler ML models like decision trees, SVM, and other models in Cat-2 are often defined by more lightweight architectures. This leads to a less computational resource overhead relative to the ML models in Cat-1 with more architectural complexity, like LSTM and VGG16. Although the Cat-1 models may achieve a higher accuracy under specific conditions, the high computational overhead can present significant challenges for deployment in a resource-constrained environment. For example, the consistent deployment feasibility result for the Cat-2 models suggests that for these models with less complex architectures, high-level attributes like the accuracy, RAM/CPU usage, and testing time provide a sufficient evaluation of their suitability for deployment. Their simpler architectures generally translate to lower computational overhead and memory requirements, making them less susceptible to the nuanced constraints that CFS's detailed analysis might emphasize. Moreover, from our feasibility analysis, we observed that despite the simple architecture of Cat-2 models, they were able to outperform some Cat-1 models with more architectural complexity, given the same dataset. This shows that it is important to consider the uniqueness of a model, compared to other similar models, in terms of its architectural complexity and performance before deployment to an environment. This trade-off between model complexity and computational efficiency is a vital consideration in the design and selection of ML anomaly-based IDS for IVNs. This is because the efficiency

of a model's architecture has a direct impact on its processing time, memory consumption, and energy utilization, factors that might have constrained resources in a vehicle in an IVN.

6.3.1.3 Data preprocessing complexity The computational overhead associated with data preprocessing is a crucial factor in determining the feasibility of a model. Some ML models require elaborate preprocessing of raw data, like the VGG16 models that convert raw CAN bus data into images and Rec-CNN models that convert the arbitration IDs into images. Under a high-frequency data exchange, these transformations can significantly increase the computational overhead and potentially introduce latency for the other processes being carried out in the vehicle. Even though such pre-processing techniques can potentially improve the model's performance and the model itself might be lightweight, the added computational requirement may render the model's suitability for practical deployment in a resource-constrained environment not feasible. Therefore, it is vital to carefully assess the efficiency of the data processing pipeline, covering both preprocessing and processing phases.

6.3.1.4 Target deployment environment available resources The availability of hardware resources in the target deployment environment is a critical factor that directly affects the feasibility of IDS models. The characteristics of the CAN bus and the capabilities of the host ECU can vary widely across different vehicle manufacturers. Although this paper discusses scenarios typical of a more generalized automotive system, it is important to recognize that certain vehicles, particularly high-end models or those equipped with advanced driver-assistance systems, may feature ECUs with enhanced processing capabilities and memory. Some may even leverage external computing resources, enabling the potential deployment of more sophisticated ML anomaly-based IDS models that might otherwise be considered impractical. A comprehensive understanding of the target

vehicle's specific resource constraints and capabilities is therefore essential for an accurate assessment of model feasibility.

Furthermore, the dynamic environment of vehicles in an IVN is characterized by many concurrent processes that pose additional challenges regarding resource availability for IDS models. The computational load on the CAN bus and the resource consumption of other processes can vary significantly, particularly during peak system demand or during an active attack. In such scenarios, the resources accessible to the IDS model may be severely constrained, leading to critical concerns about its robustness and reliability. Models that initially appear feasible and practical under standard operating conditions may become infeasible or impractical during periods of high resource usage. Thus, the capacity of the IDS models to sustain performance and resource efficiency in fluctuating system loads is a vital component of their overall feasibility.

Finally, it is important to note that though a model is evaluated as feasible for deployment, given the results of the deployment feasibility metric, it might not be suitable for the practical deployment of real-time intrusion detection. This is because the execution time it takes to make a prediction is longer than the specific time required for a real-time detection to be made, hence leading to a delayed detection. Likewise, it is also crucial that the model's accuracy score is near 1 because of the sensitivity of the task to be performed and the mission-criticality of the task. Therefore, these attributes - execution time and model's accuracy - should still be considered and evaluated after deployment, to ensure the model performs effectively.

6.4 Conclusion

This paper addresses the challenge of assessing the practicality of ML anomaly-based models for IDS in resource-constrained environments, particularly in in-vehicle networks

(IVN). To evaluate model feasibility, we introduced the CFS metric, which combine key ML evaluation criteria with system-level constraints specific to the target environment platform. Our analysis involved both baseline ML classification algorithms (Cat-2), like KNN or Decision Trees, and models with a more complex architecture, mostly based on neural networks (Cat-1), revealing notable differences in feasibility assessments depending on the model.For models in Cat-1, the CFS provided a stricter assessment, with more factors accounting for the higher resource demands of complex models and the target deployment environment. For less complex ML models in Cat-2, almost all models were considered feasible, suggesting that the models performed with low computational overhead and execution time. However, the disparities observed with more complex models in Cat-1 emphasized the need for a detailed feasibility assessment like CFS to balance the model's detection accuracy with practical deployment.

Overall, this study highlights the necessity of a comprehensive evaluation when assessing the feasibility of IDS in automotive contexts. It underscores that predictive performance alone is insufficient; practical deployment considerations in resource-constrained and dynamic automotive environments are equally crucial. Key factors, such as accuracy, model architecture, the complexity of data preprocessing, available system resources, and the influence of concurrent vehicle processes, must be meticulously analyzed. This careful consideration will ensure that the IDS solutions selected and implemented are both effective in detecting threats and feasible for real-world applications. Our future research would focus on developing a structured guideline for deploying ML models in resource-constrained scenarios. This would involve a comprehensive framework for model selection, optimization, and system-level considerations, streamlining the identification and implementation of viable ML solutions. Additionally, incorporating training duration into the feasibility evaluation is essential for

IDS models that require ongoing training or adaptation to evolving threats, enabling a thorough feasibility assessment for more real-world practicality.

CHAPTER SEVEN

RO4: INDUCING CIPHERS' FORMAL GRAMMARS FROM IMPERATIVE CODE, A TIERED ISOMORPHIC ALIGNMENT APPROACH

Securing the Internet of Things (IoT) demands multiple layers, from socio-technical trust and risk assessment to resource optimization, but all depend on the correctness of the underlying software. As automated systems increasingly underpin critical infrastructure, formal verification has become essential, providing mathematical proofs of correctness that go beyond bug-finding tests to guarantee adherence to specifications. However, a key challenge limits its adoption in IoT: a formalization gap. Most systems are implemented in imperative languages (e.g., Python, C), while verification tools rely on declarative specifications (e.g., Isabelle/HOL) [24], making manual translation costly and highly dependent on expert knowledge.

This chapter addresses Research Objective 4 using a Tiered Isomorphic Alignment (TIA) approach, which automatically assists current LLMs in translating implementation code into formal logic. By leveraging TIA, the method underscores the need for mathematically precise specifications in cryptography,where even minor implementation errors can be catastrophic [22],and shows current LLMs' limitations in cross-paradigm reasoning beyond surface syntax. TIA helps to bridge the semantic gap between imperative implementations and formal specifications of lightweight cryptographic ciphers [28] by organizing code–proof pairs into a four-tier hierarchy: Tier-1 focuses on lexical foundations, Tier-2 on the functional units of the cipher (such as the primitive), Tier-3 on the structural composition of the cipher, and Tier-4 on the top-level orchestration, inducing a complete recursive specification of the cipher loop. The approach is detailed in the following sections.

7.1 Approach

Our approach bridges the semantic gap between imperative and formal specifications through three main stages: building a cross-paradigm cryptographic database, designing a tiered dataset that maps Python to Isabelle/HOL, and enhancing LLMs' cross-paradigm translation abilities through a logic-centric training methodology.

7.1.1 Cross-Paradigm Dataset Design

LLMs struggle with cross-paradigm auto-formalization because they replicate surface patterns without grasping the operational semantics needed to connect imperative and functional logic. To address this, we create a formally verified dataset of nine standardized block ciphers and their variants. Our dataset is designed for isomorphic consistency, ensuring that each Python implementation structurally and semantically mirrors the formal Isabelle/HOL version, providing a rigorous ground truth for logic induction.

7.1.1.1 Ciphers Selection Rationale: Block ciphers are suitable formalization candidates exemplifying cross-paradigm translation due to their deterministic semantics, dense algebraic structure, and hierarchical composition. Their precise behavior and clear separation of elements facilitate the evaluation of semantic preservation and compositional reasoning. We selected nine ciphers and their security variants from three primary families to assess LLMs' Python to formal Isabelle/HOL cross-paradigm abilities. This includes SIMON [154], SIMECK [155], and XTEA [156] from the Feistel family. SPECK [154], SPARX [157], and LEA [158] from the ARX family. PRESENT [159], GIFT [160], and RECTANGLE [161] from the SPN family. This diverse selection aims to evaluate variations in operation types, composition patterns, and the models' ability to learn modular representations, particularly in hybrid ciphers like SPARX, an ARX-based primitive in

an SPN structure. Table 7.1 provides details on the cipher families and their specific composition patterns. Our evaluation focuses on three key translation aspects: converting mutable state to immutable values, mapping imperative control flow to functional recursion, and aligning standard integer arithmetic with formal constraints.

Table 7.1: Cipher Families and Translation Complexities

Family	Composition Pattern	Key Primitives
Feistel	*Sequential rounds with left/right swapping*	Bitwise AND, rotations
ARX	*Linear mixing parallel operations*	Add-Rotate-XOR
SPN	*Substitution & Permutation layers*	S-Box substitutions, permutation

7.1.1.2 The TIA Hierarchical Taxonomy The TIA framework decomposes cryptographic implementation into four incremental levels of abstraction. This stratification ensures that the model masters foundational constants before attempting to induce complex recursive state-flows. **Tier-1 or Lexical Foundation**, focuses on atomic values such as word sizes, round counts, and primitive deltas (e.g., SIMON's rotation amounts). This tier anchors the model's value consistency. **Tier-2 or Functional Units**, aligns mathematical transformations and primitives like S-Boxes, bitwise rotations, and modular addition. These are the building blocks of the cipher's semantics. **Tier-3, or Structural Composition** is the recursive bridge, where iterative state mutations (e.g., a single Feistel round) are mapped to functional definitions. It requires the model to understand local state-transition logic. **Tier-4, or Top-Level Orchestration**, represents the highest level of abstraction, where the model must induce the full recursive schedule, transforming the entire cipher loop and key schedule into a formal functional induction.

7.1.1.3 Tiered Isomorphic Aligned Dataset The existing implementations of the selected ciphers show diverse styles, making it difficult to identify the structural similarities between Python and Isabelle [162–166]. To facilitate learning, we enforce a tiered isomorphic hierarchy across both languages. We reimplemented all ciphers using a unified Python class structure that isolates semantic components (Constants, Round Functions, Key Schedules) into discrete methods. We then implemented each cipher variant's formal specification into a dedicated Isabelle/HOL theory file, maintaining a 1-to-1 structural correspondence with its Python counterpart. To establish the mathematical integrity and functional correctness, our Python implementations were tested against official test vectors provided by the original cipher authors, covering both intermediate round values and final ciphertexts. Additionally, our Isabelle/HOL implementations were verified in two-stages, notably the formal implementations' correctness using Isabelle 2025 software and running test vectors for both paradigms. The latter helped verify the execution and output of intermediate states, like the key schedule generation for both implementations. This helped ensure the mathematical equivalence of the different representations across paradigms.

To bridge these paradigms, we developed specific cipher extractors. The TIA extraction process (Algorithm 3) decomposes cryptographic implementations into functionally-aligned pairs. It starts (Line 2) by isolating the Lexical Foundation (Tier-1), extracting bit-widths, rotation constants, and S-box tables to establish the cipher's algebraic basis. The algorithm then maps Python methods like round functions and loops to their recursive definitions in Isabelle/HOL (Tier-2 and Tier-3). Finally, Top-level Orchestration (Tier-4) captures the complete cipher interface and key schedule logic, preserving structural isomorphism between the imperative source and functional target (Lines 4:7). Additionally, we ensure the model learns the relationship between numeric constants and their bit-width representations through variant-specific Python parameters during extraction. The extractor

Algorithm 3 Dataset Tiered Isomorphic Alignment Design

Input: Set of standard ciphers in Python $\mathscr{C}$ and Isabelle/HOL theories $\mathscr{T}$
Output: Unified Dataset $\mathscr{D}$

1. $\mathscr{D} \leftarrow \emptyset$
2. **foreach** *cipher* $c \in \mathscr{C}$ *and theory* $t \in \mathscr{T}$ **do**
3. $\quad \mathscr{S}_{\text{meta}} \leftarrow \text{ExtractParametricMetadata}(c)$ `/* Retrieve block size, rounds, etc. */`
4. $\quad$ **foreach** *tier* $\tau \in \{\text{Lexical}, \text{Functional}, \text{Structural}, \text{Orchestration}\}$ **do**
5. $\quad\quad P_{\text{comps}} \leftarrow \text{DecomposePython}(c, \tau)$ $I_{\text{specs}} \leftarrow \text{MapIsabelle}(t, \tau)$
6. $\quad\quad$ **foreach** *aligned pair* $(p_i, i_j) \in (P_{\text{comps}} \times I_{\text{specs}})$ **do**
7. $\quad\quad\quad instr \leftarrow \text{GeneratePrompt}(\tau, \mathscr{S}_{\text{meta}})$
8. $\quad\quad\quad component \leftarrow \{\text{Instruction} : instr, \text{Input} : p_i, \text{Output} : i_j\}$
9. $\quad\quad\quad \mathscr{D} \leftarrow \mathscr{D} \cup \{component\}$
10. $\quad\quad$ **end**
11. $\quad$ **end**
12. **end**
13. **return** $\mathscr{D}$

also extracts multi-dimensional semantic metadata for each translation pair. This metadata provides context for navigating complex paradigm shifts and enables ablation studies to identify key factors influencing translation accuracy (Line 3). The metadata consists of three main dimensions: the architectural descriptors (e.g., cipher family, cipher operational orders), the variant-specific constants (e.g., block size, key size and word width), and the transformation patterns, outlining the logic needed for effective mapping, including strategies like loop to recursion and modular arithmetic. By including the metadata, we can assess the model's contextual awareness and determine if it is memorizing code or using architectural hints for reasoning.

The final output is a concatenation of the Instruction (indication to the model to perform the cross-paradigm translation), the Input (Python implementation), the Output (Isabelle/HOL implementation), and the metadata (Lines 8:13).

7.1.2 Model Selection and Architectures

We evaluate our methodology using five state-of-the-art open-weights LLMs, specialized for code. This includes DeepSeek-Coder-Instruct (1.3B, 6.7B) based on Llama architecture [167], a dense architecture with multi-head attention, optimized for repository-level dependency logic, providing a critical test of whether structural awareness from large-scale imperative codebases transfers to the functional recursion of Isabelle/HOL. We also used StarCoder2 (3B, 7B) developed by BigCode [168], using grouped query attention, and trained on high density of formal languages (*The Stack v2*), for their specialized features such as sliding window attention and an expanded context window. Lastly, we used Qwen2.5-Coder-Instruct based on Qwen2 architecture [169], optimized for advanced mathematical reasoning and strict instruction adherence.

While the selected models are dense decoder-only Transformer architectures [170], with every parameter activated for every token, we also analyse a Mixture-of-Experts (MoE) model, DeepSeek-Coder-V2-Lite. It uses a sparse activation method, activating only a subset of its 16B parameters (2.4B) per inference step. This allows us to explore whether sparse expert routing enhances the inductive bias for the multi-tiered logic needed in formal cryptographic translation.

7.1.3 Metadata Strategy Ablation Study

A core contribution of our work is determining the optimal information density required for auto-formalization. We evaluate the impact of the extracted metadata on translation fidelity through four configurations: 1) baseline instruction only (No-Metadata), 2) partial metadata like cipher parameters (Partial-metadata), 3) structured metadata, where a dynamically adapted list of parameters is added to the instructions (Structured-metadata), and 4) the entire metadata, flattened representation of all metadata (All-metadata).

All strategies included the constraint: "Provide ONLY the Isabelle/HOL code; No explanations". This allowed us to analyze how different metadata presentations influence formal verification translation quality, focusing on semantic matching and value consistency.

7.1.4 Evaluation Framework: Proposed Metrics

Traditional NLP metrics like BLEU and CodeBLEU are inadequate for evaluating generated formal cipher grammars, as they emphasize token-level similarity over logical correctness. In cryptography, even small discrepancies (e.g., $\oplus$ vs $\wedge$) can lead to errors. Therefore, we propose a hierarchical rubric to assess the structural, functional, and parametric correctness of the generated cipher implementations consisting of the Syntax Validity (SV), the Semantic Match (SM), and the Value Consistency (VC) metrics, ranging from 0-1.

7.1.4.1 The Syntax Validity (SV) Metric

The SV metric assesses the structural integrity of the generated translations and their compliance with Isabelle/HOL formalisms. By using a structural pass with language isolation, SV distinguishes theorem-prover-compatible code from syntactically similar but semantically incompatible outputs like Standard ML, or procedural pseudocode like Pythonic definitions. SV is defined in eq. 7.1,

$$SV = F_{basic} \times F_{align} \tag{7.1}$$

where basic Validity (F_{basic}) involves identifying the primary formal entry point $C \in \{\text{definition, fun, function, primrec, lemma}\}$ and checking its logical integrity through a stack-based balanced-parenthesis verification. $F_{basic} = 1.0$ indicates a valid C with balanced parentheses, 0.5 denotes a valid C but lacking mandatory structural keywords

(e.g., where, =), while a score of 0.0 signifies either no valid C was identified or there are unbalanced delimiters.

Formal Alignment (F_{align}) represents the formal alignment of the generated grammar against the expected or reference grammar. It is defined in eq. 7.2,

$$F_{align} = \max\left(0, (0.4 \cdot S_{cmd} + 0.6 \cdot S_{mark}) - \sum \mathscr{P}_{anti}\right) \tag{7.2}$$

where S_{cmd} measures the similarity of the formal command (exact match = 1.0; related family match = 0.5), and S_{mar}, the set-overlap ratio of Isabelle delimiters $\mathscr{M} = \{::, \text{where}, \Rightarrow, \times, \text{word}\}$. To strictly focus on the target language, a penalty ($\mathscr{P}_{anti} = 0.3$) is deducted for any foreign anti-patterns (e.g., SML-style val, Pythonic def, or C-style). This ensures that high semantic vector scores align with formal grammar.

7.1.4.2 The Semantic Match (SM) Metric The *SM* metric assesses the functional alignment of operational logic, focusing on cryptographic primitives. Unlike syntactic metrics, *SM* treats implementations as multi-sets of bitwise and arithmetic operations, emphasizing the correct mathematical vocabulary for formal parity. As defined in eq. 7.3,

$$SM = \left(0.3 \cdot R_{seq}\right) + \left(0.7 \cdot R_{occ}\right)\right) \cdot \left(1 - \sum \mathrm{P}_{crit}\right) \tag{7.3}$$

SM is a weighted combination of operator frequency (R_{occ}) and sequence similarity (R_{seq}), adjusted by a critical mismatch penalty. The Occurrence Ratio (R_{occ}), weighted at 70%, measures operator density and presence, serving as the main indicator that the model has induced the correct transformation primitives (e.g., XOR, rotations) required by the cipher tier.

The Sequence Similarity (R_{seq}), weighted at 30%, employs Gestalt pattern matching to assess the longest common subsequence of operations. Although the logical order is crucial for cryptographic security, a lower weight is assigned to the lexical sequence to reflect

the structural differences in the declarative paradigm. In Isabelle/HOL, similar functional logic can be expressed through various isomorphic structures, such as nested let bindings versus recursive folds, which do not correspond directly to Python's iterative sequence. This lower weight helps avoid over-penalizing valid but structurally diverse formalizations. Additionally, a linear incremental penalty (P_{crit}) of 0.3, and capped at 1 is applied for each family-level operator mismatch (e.g., confusing bitwise logic with list-processing primitives), penalizing the use of syntactically similar but semantically incorrect operators.

7.1.4.3 The Value Consistency (VC) Metric The VC metric, as defined in eq. 7.4, evaluates the cipher's parameters integrity (bit-widths, S-Box entries, round counts) of the generated code. In cryptography, a single error in rotation or an S-box can invalidate the entire specification.

$$VC = \left(\omega \cdot R_{occ} + (1-\omega) \cdot R_{seq}\right) - \sum \mathscr{P}_{size,rot} \tag{7.4}$$

VC performs a static analysis of numeric constants and penalizes cryptographic mismatches, such as key size or shift amount. Similar to SM, VC is a weighted combination of R_{occ}, the number of occurrences of the constants, and R_{seq} that uses Gestalt to find the similarity between the generated and referenced constants.

To avoid the Commutativity Paradox in operations like XOR and addition, we implement a Context-Aware weight shift. For commutative operations, we set $\omega = 0.8$ to prioritize correct value presence, while for non-commutative contexts, $\omega = 0.4$ ensures positional integrity. We also apply penalties ($\mathscr{P}_{size,rot}$, up to 1) for errors in critical cryptographic parameters, such as bit-widths or rotation amounts, to protect cipher security. Additionally, we transform code to mask variable names and normalize decimal, hexadecimal, and binary values into a unified numeric format, enabling the model to handle non-standard naming while capturing context-dependent significance in constants.

This allows flexible ordering in commutative operations while maintaining verification for critical structural parameters.

SV, *SM*, and *VC* are calculated after regex-based cleaning, focusing on semantic content, with all metrics defaulting to 0.0 if no output is generated. This method separates variable naming from semantic evaluation, allowing us to assess a model's ability to produce correct logical structures independent of naming conventions. *SV* ensures well-formed specifications in the Isabelle/HOL grammar, *SM* evaluates the alignment of cryptographic logic, and *VC* indicates functional correctness and serves as a precise measure of a model's ability to maintain data integrity for formal specifications. However, these metrics are domain-agnostic and not specific to cryptography. They focus on evaluating grammar compliance, operator sequencing, and constant integrity of formal outputs, providing a general framework for evaluating formal neural-symbolic translations.

7.2 Results and Discussion

Dataset Analysis: The extracted dataset focuses on transitioning from imperative bit-manipulation to declarative formal specification for each cipher and its security variants. It consists of $N = 715$ extracted components from SIMON, SIMECK, SPECK, SPARX, PRESENT, and GIFT for training, $E = 18$ for evaluation on LEA, and Out-of-Distribution ($OOD = 40$) components extracted from XTEA, RECTANGLE, for assessing zero shot generalization.

Structural and Family Distribution: The training set is divided into four hierarchical Tiers based on their logical role in the cipher (Section 7.1.1.2 (Table 7.2). This distribution reflects the inherent architectural density of cryptographic designs, where parametric invariants (Tier-1) and recursive state transitions (Tier-3) form the logical bulk of the system.

Table 7.2: Dataset Distribution across TIA Tiers and Cipher Families

Semantic Tier	ARX	Feistel	SPN	Total
Tier-1 (Lexical)	97	107	62	**266**
Tier-2 (Functional)	50	39	53	**142**
Tier-3 (Structural)	110	98	37	**245**
Tier-4 (Orchestration)	26	26	10	**62**
Total	**283**	**270**	**162**	**715**

Quantifying the Semantic Gap: To characterize the difficulty of cross-paradigm auto-formalization, we analyze the Structural Divergence between the two languages. We define this divergence as the non-linear relationship between the average Logical Lines of Code (LLOC) required to implement the same cryptographic logic in Python versus Isabelle/HOL. Table 7.3 shows that isomorphic compression (Tier-3) reduces verbose Python for-loops into higher-order functional fold operations in Isabelle, resulting in a code volume ratio of approximately 0.42, where about 12 lines of imperative code condense to around 5 lines of formal logic. In contrast, Formal Expansion (Tier-4) often increases orchestration logic size in Isabelle from 4.50 to 6.50 LLOC due to mandatory bit-width casting and strict type constraints related to the word theory. This structural mismatch, along with a near-zero Jaccard similarity vocabulary overlap of ($J = 0.0608$) indicates that the task is not a simple translation but rather a deep logic induction problem.

Table 7.3: Structural Complexity and Logical Density: Comparing average Logical Lines of Code (LLOC) across Semantic Tiers.

Tier	$\text{Python}_{\text{LLOC}}$	$\text{Isabelle}_{\text{LLOC}}$	Vocab Overlap (J)
Tier-1	1.00	1.10	0.041
Tier-2	6.12	4.80	0.065
Tier-3	12.45	5.20	0.072
Tier-4	4.50	6.50	0.064
Mean	**6.18**	**6.19**	**0.0608**

OOD Structural Generalization: To assess the framework's robustness, we examine the combinatorial novelty of our out-of-distribution (OOD) ciphers (LEA, XTEA, RECTANGLE). Although these ciphers use core algebraic primitives found in the training set (e.g., bitwise rotations and modular addition), they introduce novel architectural compositions that the model hasn't encountered. This approach tests zero-shot structural generalization, especially the model's capacity to integrate known transformation patterns into a new cipher logic. By keeping pattern parity while varying structural execution, we focus on the model's ability to generate semantic grammar rather than merely memorizing specific implementation templates.

Tokenization Analysis for Hyperparameter Rationale: To determine the models' hyperparameters, we analyzed a tokenization density across all models. Given the input sequence (Instruction + Metadata + Input + Output), we identified DeepSeek-Coder as the upper-bound baseline for vocabulary verbosity, with a 95th percentile (P_95) total sequence length of 905 tokens and a maximum length of 1,300 tokens. Given the Output sequence only, we also identified DeepSeek-Coder as the upper-bound baseline for vocabulary verbosity, with a maximum output of 476 tokens, and 296 as the 95th percentile. We therefore set the models' global context window to $L_{max} = 1024$ and the generation limit to $N_{out} = 512$, ensuring over 95% of instances are processed without information loss from context truncation, while providing the necessary headroom for the structural expansion observed in Tier-4 orchestration tasks.

7.2.1 Simulation Setup

All simulations were done on a High-Performance Computing cluster (NVIDIA V100 (16GB) GPUs), with specific configurations for reproducibility and rigor. The repository is available on [171].

Quantization and Memory Optimization: We used 4-bit NormalFloat (NF4) quantization with *bitsandbytes* to fit 7B parameter models within the 16GB VRAM limit of V100 GPUs. By implementing Gradient Checkpointing, we reduced memory usage during the backward pass, enabling a stable effective batch size of 16 via gradient accumulation.

PEFT via LoRA and Label Masking: We employed Parameter-Efficient Fine-Tuning (PEFT) with the LoRA adapter methodology. To optimize the gradient budget, we implemented a *instructional label masking* protocol, calculating cross-entropy loss only on target Isabelle/HOL tokens. Prompt metadata and source Python tokens were labeled (-100) to neutralize their gradient contribution, enabling the model to concentrate on mapping to formal logic.

Scale-Specific Hyperparameters: For the dense models, Due to the disparity in knowledge capacity across model scales, we employed non-uniform LoRA ranks. For the 1.3B models, we used a high-capacity LoRa rank $r = 64$ and LoRa ($\alpha = 128$) targeting Query, Vector, Key, and Output (Q, V, K, O) modules to provide sufficient trainable surface area for new syntax acquisition. For the 7B variants, we used lora rank $r = 32$, and lora alpha ($\alpha = 64$) targeting Q, V modules to maintain stability and balancing cross-domain knowledge retention with task-specific specialization [172]. For Deepseek-V2-Lite, given the mixture of experts, we used lora rank $r = 16$ and lora alpha ($\alpha = 32$) targeting (Q, V, K, O) modules.

Stability and Convergence: All trainings were conducted for 4 epochs using the AdamW optimizer with a linear learning rate warmup ratio of 0.1. To ensure the models generalized beyond cipher-label memorization, we implemented Early Stopping with a patience of 3, monitored against the validation loss on the validation dataset.

7.2.1.1 Training Protocols To evaluate the framework's ability to induce cryptographic logic, we perform two main simulations: the baseline and OOD generalization evaluations.

Zero-Shot Baseline Evaluation: Prior to fine-tuning, all base models were evaluated on a test set to establish a performance floor. To ensure a deterministic and conservative baseline, we utilized greedy decoding ($\tau = 0.0$). Based on an empirical analysis (Section 7.2) of the dataset's token density across all model-specific vocabularies, the context window was set to $L_{max} = 1024$ and the generation limit to $N_{out} = 512$, ensuring zero information loss due to truncation for all the instances.

Out-of-Distribution (OOD) Generalization: To mitigate the variance inherent in small datasets and better simulate real-world formalization tasks, we perform an Out-of-Distribution (OOD) training protocol. We used the unified training corpus of all 715 extracted components, ensuring the model sees the full diversity of SIMON, SIMECK, SPECK, SPARX, GIFT, and PRESENT, and for validation and early stopping, we used the LEA cipher (ARX family). This helped maximize the gradient signal while providing a generalization through unseen cipher architectures. Lastly, the models' final performances are benchmarked zero-shot on two novel architectures: RECTANGLE (SPN family) and XTEA (Feistel family), the defined OOD set. This tests cross-paradigm logic induction across all three primary cipher families.

7.2.2 Performance Evaluation Metrics Interpretation

Given the strict type-system of Isabelle/HOL and the binary nature of cryptographic correctness, to evaluate the models and the TIA approach, we disaggregate our evaluation into three distinct metrics, as established in Section 7.1.4 (Table 7.4). Critically, the scoring scale is non-linear; while the difference between a 0.4 and 0.5 may reflect minor syntactic noise, the transition from 0.85 to 1.0 represents a rigorous leap from an approximate

translation to a logically sound, executable specification. Therefore, we considered Syntax Validity (SV) as the primary metric, where a score below 0.6 indicating a failure to conform to mandatory Isabelle declaration patterns (e.g., `definition...where`), rendering the output non-compilable within the Isabelle/HOL kernel. Semantic Match (SM) and Value Consistency (VC) measure logical and parametric fidelity, respectively. For all the metrics, we define an overall score ≥ 0.85 as *excellent*, representing a high-quality translation that requires negligible human correction for integration into an Interactive Theorem Prover. Conversely, scores below 0.30 signify a *failed* induction where the cryptographic constants or operator sequences are fundamentally decoupled from the reference implementation.

Table 7.4: Range Values Interpretation for SV, SM and VC Metrics

Range	**Class**	**Cryptographic Utility**
0.85–1.00	Excellent	High-fidelity; formal verification ready
0.70–0.84	Good	Mostly correct; minor syntactic fixes
0.50–0.69	Moderate	Correct logic; requires structural repair
0.30–0.49	Poor	Major parametric/operator mismatches
0.00–0.29	Failed	Fundamentally incorrect; non-usable

7.2.3 Models' Performance and Ablation Analysis

The aggregated performances of all the dense models across all TIA tiers and metadata configurations(Fig. 7.1) reveal a clear trajectory from baseline functional failure to tiered logical alignment.

The Zero-Shot (Baseline), shows a significant failure in both syntactic and functional areas, with a Value Consistency (VC) of 0.05 in functional units (Tier-2) and 0.002 in orchestration (Tier-4) (Fig. 7.1). This indicates that even high-capacity models struggle to connect imperative cryptographic structures to declarative Isabelle/HOL logic without specific domain knowledge. The introduction of TIA training even without

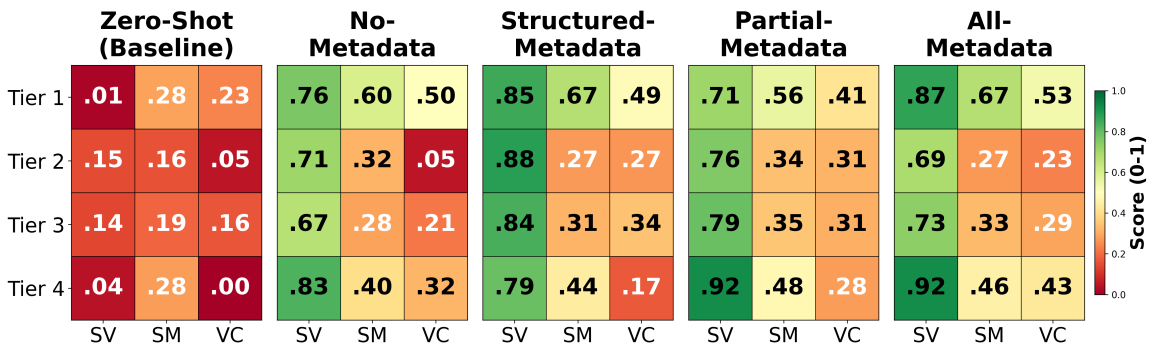

Figure 7.1: OOD Performance: Zero-Shot vs Metadata Ablations for Dense LLMs

metadata (No-Metadata) results in a significant literacy increase, with Syntax Validity (SV) stabilizing above 0.66 across all tiers. However, logic remains weak in Tier-2, with negligible VC scores (0.049). Additionally, the shift from No-Metadata to Partial-Metadata shows significant framework efficacy, with a 6.4 times improvement in Tier-2 Value Consistency (from 0.049 to 0.314). This indicates that targeted algorithms and variant parameters facilitate functional grounding.

Moreover, the Partial-Metadata optimizes Tier-2 Functional and Tier-3 Structural logic, while the All-Metadata achieves the highest Tier-4 Orchestration score (0.432). This highlights a trade-off between selective grounding, improving local functional accuracy, and exhaustive context, valuable for the ciphers' global orchestration. Across all metadata-augmented runs, SV scores consistently surpass VC scores. This finding separates translation fluency from logical alignment, confirming our hypothesis that the main challenge of auto-formalization is preserving functional invariants across different paradigms, rather than target language syntax.

7.2.3.1 Model Sensitivity and Functional Gains We define functional sensitivity (ΔVC) as the gain in Value Consistency (VC) from the baseline (i.e., No-Metadata) to the

Partial-Metadata configuration. The Partial-Metadata setup acts as a primary comparison, focusing on targeted parameters and isolating the framework's capacity for high-density logical grounding, free from the noise of the exhaustive All-Metadata configuration.

Table 7.5 shows the computed ΔVC for all the models across tiers. It shows a significant gain in Tier-2 (+0.757) for Qwen2.5 (Qn), denoting that Qwen2.5 uses TIA as a semantic map for high-precision translation, moving from zero to strong functional awareness. Additionally, the drop observed for Deepseek-Coder-6.7B (DSC) highlights the potential conflict between internal inductive biases and the external TIA constraints, as DSC is heavily trained on imperative patterns. Moreover, Tier-2 shows the most consistent positive response to TIA, except for DSC-1.3B and StarCoder2-7B (SC2), highlighting that the model size can potentially impact the model's learning capabilities. Among the dense architectures, Qwen2.5-Coder-7B in the Partial-Metadata configuration proved to be the most resilient, excelling in semantic grounding. Additionally, the selected dense architectures showed an aggregated ratio of 9.7, 2.9, and 1.9 times increase in Syntax Validity, Value Consistency, and Semantic Match, respectively, compared to the zero-shot baseline.

Table 7.5: OOD Functional Sensitivity: VC Delta ($\Delta VC = VC_{Partial} - VC_{None}$). Positive values (bold) show successful semantic grounding, & negative values show instruction interference.

Model	Tier-1	Tier-2	Tier-3	Tier-4	Avg.
Qn-7B	-0.016	**+0.757**	**+0.238**	**+0.375**	**+0.339**
DSC-1.3B	+0.019	+0.085	-0.025	+0.024	+0.026
DSC-6.7B	-0.313	**+0.341**	+0.123	-0.552	-0.100
SC2-3B	-0.117	-0.007	+0.043	-0.042	-0.031
SC2-7B	-0.014	**+0.146**	**+0.081**	-0.010	+0.051

7.2.3.2 Comparative analysis against MoE DeepSeek-V2 (DS) Lite (MoE) shows a better performance on zero-shot baseline evaluation, compared to the dense architectures. It shows an average of 0.61 for SV, 0.52 for SM, and 0.21 for VC, with an average increase of 1.6 times from baseline to Partial-Metadata configuration. When compared to the best performing dense model, Qwen, the results in Table 7.6 show that DS-V2 Lite outperforms Qwen in Tier-1 (+11.8%) and Tier-3 (+7.1%) due to its Mixture-of-Experts (MoE) architecture, which effectively retrieves specific cryptographic constants. However, Qwen excels in Tier-4 (54.2% vs 8.3%), indicating a complexity ceiling in MoE designs, where dense models maintain better global state coherence. In Tier-2, both models perform similarly (75%), highlighting it as a convergence zone where architectural differences are minimized.

Table 7.6: VC Comparison of Qwen2.5-7B vs. DeepSeek-V2 Lite for Partial-Metadata Configuration.

Tier	Qwen	DS-Lite	Δ
Tier-1	0.456	0.574	**+0.118**
Tier-2	0.757	0.753	**-0.004**
Tier-3	0.352	0.423	**+0.071**
Tier-4	0.542	0.083	**-0.459**

7.3 Conclusion

To address the challenges of translating imperative code into Isabelle/HOL specifications, we present the Tiered Isomorphic Alignment (TIA) framework. TIA leverages hierarchical decomposition and targeted metadata to bridge the semantic gap between those paradigms and improve LLMs' cross-paradigm translations. Our evaluation shows that TIA provides a transformative scaling effect for dense architectures, yielding

a 9.7, 2.9, and 1.9 times increase in newly proposed metrics of Syntax Validity, Value Consistency, and Semantic Match, respectively, compared to the zero-shot baseline. While the DeepSeek-V2 Lite (MoE) architecture shows superior baseline performance, it only achieves a 1.6 times improvement for each metric due to instruction interference. Despite our advancements, a complexity ceiling in structural composition hinders fully automated end-to-end verification. Future research will focus on adaptive metadata routing and reinforcement learning to improve logical alignment in high-assurance software engineering.

CHAPTER EIGHT

RO5: CRYPTOGRAPH: A NEURO-SYMBOLIC AUTOMATED FRAMEWORK FOR LIGHTWEIGHT BLOCK CIPHERS SECURITY EVALUATION

Building on top of [28], we propose CryptoGraph [29] as a glass-box analysis framework that converts formal cipher specifications into graph-based representations (ASTs), preserving their structural information during analysis. Building on these representations and high-level metadata vectors, we employ Graph Neural Networks (GNNs) to learn the relationship between a cipher's internal structure (e.g., diffusion layers) and its cryptographic strength, yielding an automated and interpretable metric for early-stage security assessment. This work is motivated by the rapid growth of non-standard lightweight ciphers for IoT networks, where manual cryptanalysis is too expensive and black-box ML models for ciphers' security assessment lack interpretability.

8.1 Methodology

Our proposed framework, CryptoGraph, bridges the gap between formal verification and deep learning to automate the structural security assessment of lightweight block ciphers, leveraging Graph Neural Networks (GNNs) to reason about the cipher's internal algorithmic topology. To ensure algorithmic diversity and robustness, CryptoGraph focuses on 11 block ciphers (55 total variants: 47 for training and 8 for testing) across three main cipher families: Feistel, ARX, and SPN. For Feistel family, we used variants of SIMON [154], SIMECK [155], and XTEA [156]. For the ARX family, we used variants of SPECK [154], HIGHT [173], and LEA [158] . For SPN family, we used variants of PRESENT [159], GIFT [160], SPARX [157], SKINNY [174] and RECTANGLE [161]. This selection includes primitives from the NIST Lightweight Cryptography finalists [175] (SKINNY as

the core of Romulus [176], GIFT as the core of GIFT-COFB [177]), ensuring the framework is evaluated against modern, industry-standard designs.

A critical challenge when applying machine learning to code is the semantic gap, or ensuring the model learns from the logic of the algorithm rather than the syntax of the implementation. To address this, we used Isabelle/HOL [24], an interactive theorem prover, to create formal specifications from the ciphers' imperative implementations (e.g., Python, C++). We selected Isabelle/HOL over Coq [178] or ACL2 [179] for its robust support for machine word semantics and automated termination proving. Unlike Coq's constructive logic, which often requires manual proofs for termination, Isabelle's function package automates pattern-matching compilation and termination checking for the recursive structures common in ciphers (e.g., Feistel networks). Furthermore, we leverage the Word library from the Main theory, which provides pre-verified automation for fixed-width bit-vector arithmetic (e.g., modulo 2^n addition, bitwise rotations), ensuring that our graph representations strictly adhere to hardware-level semantics without requiring manual lemma derivation for every operation.

We assume that the formal properties extracted from the Isabelle/HOL definitions, especially the data-flow topology, operation types, and connectivity, are sufficient proxies for the structural complexity that underlies cryptographic security. We rely on the established cryptanalytic literature (e.g., differential/linear attack complexity) to establish the ground truth security levels (Broken, Low, Medium, High) for our training data, assuming that the consensus of the cryptanalytic community regarding these well-studied ciphers is accurate. Additionally, to balance the training data for each class, we generated synthetic variants. For our synthetically generated variants, we assume a monotonic relationship between round depth and security. We posit that reducing the round count significantly below the established security margin (e.g., to 50% of the standard

rounds) deterministically degrades the security level to 'Low' or 'Broken,' as supported by differential and linear cryptanalysis bounds [180]. This allows the framework to dynamically assign labels to synthetic variants without performing manual cryptanalysis for every instance. With this understanding, we present our framework architecture and implementation details in Fig. 8.1.

CryptoGraph automates structural security assessments for lightweight block ciphers using a four-stage neuro-symbolic pipeline. Stage I consists of translating reference implementations into Isabelle/HOL theory files to verify functional correctness and ensure logically sound generated datasets. Stage II extracts both a graph-based Abstract Syntax Tree (AST) and a statistical Protocol Design Vector (PDV) to capture data-flow topology and high-level structural features. Stage III uses a constraint-preserving engine to generate diverse, logically valid cipher variants for a balanced training distribution. Stage IV employs a hybrid model combining a scale-invariant Graph Neural Network and a dense classifier to map structural patterns to security labels. By integrating formal verification (Stages I-II) with geometric deep learning (Stages III-IV), the framework eliminates the garbage-in, garbage-out risk typical of software analysis models, ensuring that predictions are grounded in verified cryptographic logic.

8.1.1 Stage I: Formal Specification and Verification

CryptoGraph is founded on the rigorous formalization of the selected lightweight block ciphers in Isabelle/HOL. This helps replace ambiguous software implementations with ground truth, which anchors the rest of the learning pipeline.

Semantic Translation Protocol: We selected reference imperative implementations of the ciphers as functional blueprints due to their readability and widespread use, and performed a manual refactoring to convert the imperative code into functional

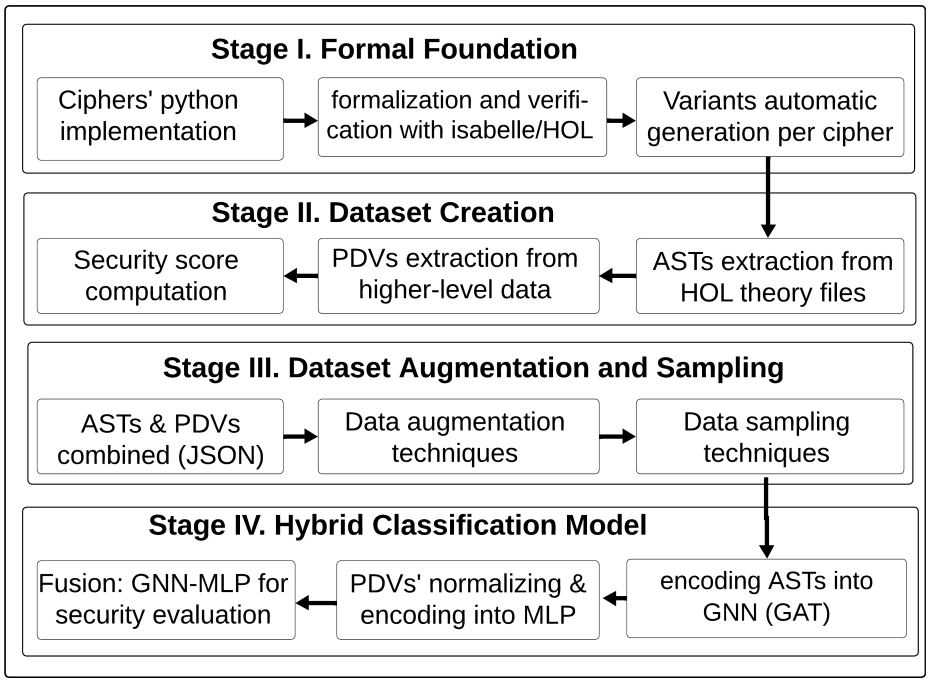

Figure 8.1: Automated Cipher Security Evaluation Architecture

Isabelle/HOL specifications (Table 8.1). This process follows a three-step protocol. First, we perform *type strengthening* by replacing dynamic imperative types (e.g., integers) with Isabelle's statically typed bit-vectors from the `Word` library (e.g., a 64-bit block becomes `64 word`), eliminating overflow and type ambiguities. Second, we map mutable state updates (e.g., $x' = x \oplus y$) into pure functional `let`-bindings. Third, we transform stateful control structures, specifically iterative for loops, into stateless recursive functions or fold operations. This rigorous translation ensures that the resulting formal specification preserves the exact cryptographic semantics of the reference implementation while adhering to the side-effect-free requirements of HOL.

Dual-Validation of Correctness: To mitigate translation errors, we implement a strict dual validation process that enforces both standard compliance and mathematical consistency. First, we run the implemented Isabelle/HOL cipher functions against the

official test vectors in the design specifications, to confirm bit-exact input/output matches with the standard, validating the correctness of the implemented functions. Second, we use Isabelle's automated reasoning engine to formally prove invertibility, showing that for all keys and messages, Decrypt(k, Encrypt(k, m)) = m. This guarantees that each encryption function is a true permutation and that no structural information is lost in the subsequent graph representations. Verified theory files serve as the immutable kernel from which all subsequent graph datasets are derived.

Table 8.1: Semantic Translation: Python to Isabelle/HOL

Feature	Imperative Reference	Formal Specification
Type System	Dynamic, unbounded integers	Static, fixed-width bit-vectors (e.g., `32 word`)
State	Mutable variable updates (`x = x + k`)	Immutable functional chains (`let x2 = x1 + k`)
Validation	Unit tests on finite inputs	Symbolic proofs over infinite domains
Properties	Implicitly assumed	Explicitly proven (Bijectivity, Invertibility)
Role	Functional Blueprint	Verified Ground Truth

8.1.2 Stage II: Dataset Creation and Structural Feature Engineering

Given the selected block ciphers, we treat their open-source reference implementations not as ground truth, but as functional blueprints SIMON, SPECK [162], PRESENT [163], LEA, XTEA, HIGHT [164], GIFT [165], SPARX [181], SKINNY [182] and RECTANGLE [166]. We manually translate these blueprints into HOL definitions within Isabelle, translating imperative code (loops, variable updates) into pure mathematical functions (recursion, bitwise logic). To ensure these translations faithfully represent the intended cryptographic standards, we apply a dual-validation strategy. First, Test Vector Validation confirms compliance with official standards by verifying input-output equivalence on reference vectors. Second, Formal Verification establishes mathematical consistency by proving the invertibility theorem in Isabelle/HOL ($\forall m, k...$), guaranteeing that the

extracted graph representation captures the complete, reversible logic of the cipher without information loss. A challenge to our approach is to transform the formally verified specifications into machine-learnable representations. We bridge this semantic gap by extracting two complementary representations from the Isabelle/HOL theory files of each cipher variant, a fine-grained Abstract Syntax Tree (AST) graph that encodes algorithmic topology, and a high-level Protocol Design Vector (PDV) that captures the cipher's structural statistics. Together, they ensure the model learns both the cipher's data flow and architectural patterns.

Trust Boundary: It is critical to distinguish between the *verification base* and the *heuristic layer*. Our Trusted Computing Base (TCB) consists solely of the manual correctness of the Isabelle/HOL cipher definitions. We trust that if the Isabelle theory faithfully represents the cipher specification, the extracted AST is a correct dependency graph. The Hybrid classification component (HybridGAT) lies **outside** the TCB; it is a probabilistic heuristic tool designed to identify structural patterns and filter obviously weak designs (structural sanity checks). Instead of a mathematical proof of security, it provides a high-confidence estimation of algorithmic complexity and topological soundness.

8.1.2.1 AST Graph Extraction We developed a family-agnostic ExpressionParser that recursively traverses Isabelle/HOL cipher implementations, to build a directed graph $G = (V, E)$. In G, nodes (V) represent atomic operations (XOR, AND, ROTL), variables, and function calls. They are annotated as a 114-dimensional feature vector capturing the node type (e.g., op, var), cryptographic role (e.g., confusion, diffusion), and mathematical properties (e.g., nonlinearity score). The edges (E) encode data dependencies with 13 distinct edge types (e.g., argument, binding, control_flow), enabling the GNN to distinguish operand inputs from logical sequencing. Additionally, this parser is cryptography-aware,

tracking key injection points and distinguishing round functions from key schedules, tagging them with dedicated metadata in the graph.

Node Feature Vector: Given the structural tree, nodes and features are created to form a more detailed graph. The AST's nodes are identified by their node ID and created along with a set of features that vary depending on the node's significance and type. The node feature vector has a total of 114 dimensions, concatenated from three distinct groups of features: the base, function-specific numerical, and categorical semantics, summarized in Table 8.2. First, the base numerical features apply to all nodes, three raw floating-point values to quantify their direct cryptographic properties: the crypto_strength, the diffusion_power, and nonlinearity. The crypto_strength quantifies the operation's strength from a cryptanalysis perspective. For example, we determined an ordinal ranking for each operation type, and for linear operations like XOR, a value of 1.0 was assigned. For bitwise operations are linear but provides diffusion, hence a slightly higher value of 1.5 was assigned. For non-linear operations like AND, a value of 3.0 was assigned for the crypto_strength. Using the same logic, the diffusion_power characterizes operations that spread bit influence or diffusion, and is considered cryptographically stronger than simple mixing. Whereas nonlinearity helps identify the nodes that provide confusion. Secondly, the function-specific numerical features are extracted from nodes of type `function` that are obtained by flattening the cryptographic_patterns block into a 10-dimensional vector. This includes values like feistel_network or operation counts from cryptographic_operation_distribution (e.g., linear_mixing, nonlinear_mixing). For other node types (e.g., op, var), these features are set to zero to ensure consistent vector length. Finally, the categorical features are generated by scanning the entire theory files to find all unique string identifiers for four categories: the type, the cryptographic role, the data flow

role, and the context of the node. Each category vector is then one-hot encoded, and the resulting vectors are concatenated into a 101-dimensional vector.

Table 8.2: 114-Dimensional Node Feature Vector - Metadata

Feature	Description (Examples)
Base Numerical (3-dims): Raw values quantifying direct node properties	
crypto_strength	Operation cryptographic power (AND=3.0, XOR=1.0)
diffusion_power	Operation's bits diffusion (ROTL=2.0, AND=0.0)
nonlinearity	Operation's non-linearity (AND = 3.0, XOR = 0.0)
Function-Specific Numerical (10-dims): From cryptographic_patterns	
is_feistel	Boolean flag: `feistel_network` (1.0 or 0.0)
arx_chains	Integer count: `len(arx_operation_chains)`
is_spn	Boolean flag: `spn_layers` (1.0 or 0.0)
keysched_complex	Numerical value:`key_schedule_complexity`
feistel_rounds	Integer count: `feistel_rounds_detected`
round_structs	Integer count: `len(round_structures)`
op_dist_linear, op_dist_nonlinear, op_dist_diffusion, op_dist_op	Integer count from `cryptographic_operation_distribution`
Categorical (101-dims) Semantic roles' one-hot encoded vector	
type	Node's syntactic type (op, var, literal, function)
crypto_role	Node's cryptographic purpose (linear_mixing, nonlinear_mixing, diffusion, sbox_substitution)
data_flow_role	Node's role in the transformation path (processing, confusion, diffusion_layer)
context	A node's syntactic block (XOR_arg, AND_arg, rotation_amount, binding_rs_x, F_function)

Edge Feature Vector: Each edge connects 2 nodes and contains 3 features: source, target, and an edge type. From the theory files and the data flow, the parser identifies 13 unique edge types (e.g., child, arg, amount, contains, binding). Each type is mapped to a unique integer index (e.g., child=4, arg=1). This generates a 1-dimensional edge attribute tensor, enabling the GNN to learn the types of relationships between nodes. Further, we use the parser to extract rotation amounts and their order for analyzing diffusion properties. E.g., from "word_rotl 1 (word_rotl 8 x)", the parser extracts the rotation amounts (1 and 8) and their application sequence. For complex cryptographic constructs, the parser maintains context through recursive calls, capturing key schedule recursion and managing case expressions and pattern matches for decryption. It parses fold operations in iterative encryption, revealing the round application structure. These processes enhance the dataset's

structural accuracy and semantic richness. Fig. 8.2 shows an illustration of the AST structure for Simon's F-function (*F_function x = xor (and (word_rotl 1 x) (word_rotl 8 x)) (word_rotl 2 x)* .

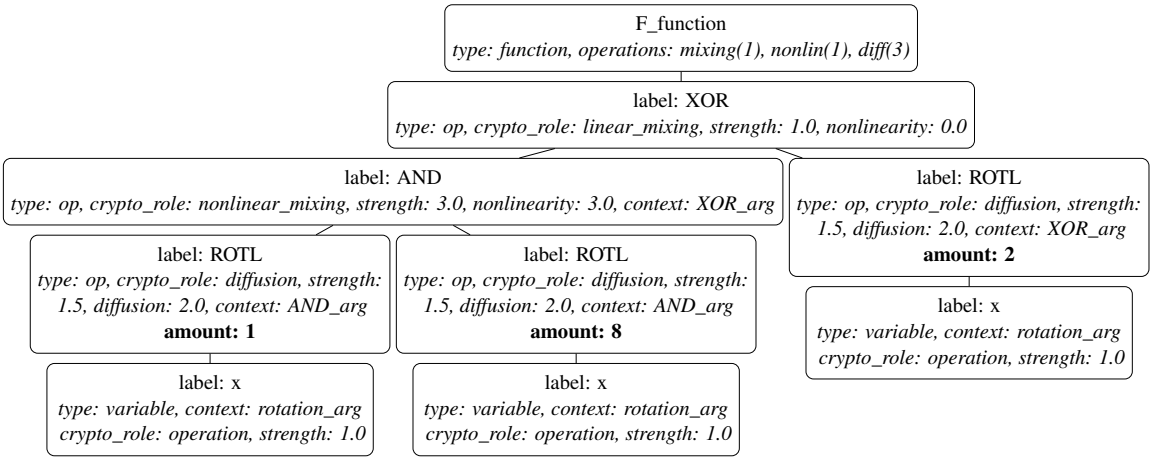

Figure 8.2: AST for Simon's F-function with some nodes' semantic features annotated (Table 8.2)

AST and PDV Extraction Process: As described in Algorithm 4, the process takes as input the cipher variant theory file and the cipher family profile, and outputs a JSON file containing the nodes, edges, function set, security score, and security level.

It starts by identifying the cipher family using the cipher family profile input, then instantiates the family-specific extractor class (among FeistelExtractor, ARXExtractor, and SPNExtractor), and the cipher variant, by extracting the block and key sizes (Lines 1-3). Given the cipher family, the specific cipher class is then used to identify and extract all cipher-specific definitions and function blocks within the theory file (Lines 4-15). For each extracted function, a recursive parser instance from the ExpressionParser class traverses the right-hand side of the definition, transforming the nested Isabelle expressions into a tree structure. This tree is then flattened into a set of nodes and edges to form the final graph

Algorithm 4 Extraction Pipeline for AST, PDV, and Security Score

Input: Cipher family profile P, cipher variant theory file T
Output: AST $G = (V, E)$, PDV PDV, Security score/level SS/SL

1. **Step 1: Initialization**
   Identify cipher family F from P & Instantiate family extractor: $E \leftarrow$ (FeistelExtractor, ARXExtractor, SPNExtractor)
2. Extract $(block\_size, key\_size)$ from filename pattern
3. **Step 2: Cipher Family-Specific Structural Analysis**
4. **switch** F **do**
5. | **case** *Feistel* **do**
6. | | Detect F-function, rotation diversity, swaps
7. | **end**
8. | **case** *ARX* **do**
9. | | Extract shift constants, ARX operation patterns
10. | **end**
11. | **case** *SPN* **do**
12. | | Analyze S-box and permutation layer composition
13. | **end**
14. **end**
15. **Step 3: Expression Parsing**
16. Extract functions $(f_i, body_i)$ from T using E
17. **foreach** *function f in definitions* **do**
18. | $G_f \leftarrow$ ExpressionParser.parse$(body_i)$
19. | **Enrich Nodes** V_f**:** Add $\{type, crypto\_role,$
20. | $data\_flow\_role, strength, diffusion, nonlinearity\}$
21. | **Create Edges** E_f**:** Add $\{contains, child, arg, amount, ...\}$
22. | $G \leftarrow G \cup G_f$
23. **end**
24. **Step 4: PDV Construction & Normalization**
25. $PDV_{family} \leftarrow$ General+cipher-specific features from E
26. $PDV \leftarrow$ UnifiedPDVProcessor(PDV_{family}, G)
27. **Step 5: Security Score Assessment**
28. $attacks \leftarrow$ Lookup known cryptanalysis for variant
29. $SS \leftarrow$ Uses family-specific security parameters
30. $SL \leftarrow$ Classify security level (low-medium-high)
31. **Return:** AST graph $G = (V, E)$, unified PDV PDV, security score SS, security level SL

(Lines 16-24), with the nodes enriched with the nodes' features described in Table 8.2. The PDV is then built uniquely for each cipher family, and then unified into a single schema to facilitate MLP training (Lines 25-26). Lastly, the security score SS and corresponding security level SL, are computed and assigned for each cipher variant, given its theory file

or Isabelle/HOL specifications T, and cipher profile P. Details on the PDV extraction and security score computation are provided in the next subsections.

8.1.2.2 The Protocol Design Vector (PDV) Extraction To characterize high-level cipher properties across block cipher families, we introduce a 33-dimensional PDV that encodes fundamental structural properties rather than implementation-specific artifacts. We group the 33 PDV features into four categories (Table 8.3). Firstly, the *topological metrics* (e.g., branch_count, critical_path_proxy) let the model distinguish deep, narrow Feistel networks (e.g., Simon, depth ≈ 90) from shallow, wide ARX designs without explicit family labeling. Secondly, the *connectivity metrics* (e.g., avg_degree_centrality, interaction_density) help quantify how strongly the cipher mixes state variables. SPN ciphers (e.g., GIFT) typically exhibit higher centrality due to S-box interconnectedness compared to the linear chains of ARX ciphers. The *algebraic complexity*, on the other hand, is captured via an algebraic_degree_proxy that measures the density of nonlinear (e.g., AND, modular ADD) versus linear (XOR) operations, helping differentiate arithmetic hardness (ARX) from boolean hardness. Lastly, instead of raw counts, we normalize *operation frequencies* by block size and round count (e.g., `sbox_density`), ensuring the model is scale-invariant and can compare 64-bit ciphers to 128-bit ciphers fairly.

Table 8.3: 33-Dimensional Protocol Design Vector (PDV), a high-level fingerprint of the cipher variant for the MLP.

Category	Structural Metrics
Topology	`branch_count`, `critical_path_proxy`, `permutation_type`, `sbox_type`
Connectivity	`avg_degree_centrality`, `interaction_density`, `cyclomatic_complexity_proxy`
Algebraic	`algebraic_degree_proxy`, `nonlinearity_density`, `diffusion_density`
Operation Density	`key_state_ratio`, `state_update_density`, `sbox_density`
Core Params	`block_size`, `key_size`, `rounds`, `key_injection_type`

8.1.2.3 Security Assessment and Ground Truth Labeling Assigning valid security labels to block ciphers is a challenge because security is not a binary property but a continuous spectrum of resistance. To train our model on ground truth rather than heuristics, we constructed a cryptanalytic knowledge base (`ATTACK_DB`) and a deterministic scoring process. For the cryptanalytic knowledge base, we curated a database of the best known attacks for every supported cipher family, as well as their complexity. This data was extracted exclusively from peer-reviewed proceedings (e.g., FSE, Eurocrypt). For instance, for SIMON and SIMECK, we utilize the differential bounds established in [154] [155] and subsequent linear hull improvements [183] [184]. This ensures that our computed security scores reflect the state-of-the-art in cryptanalysis. The scoring process maps the best known attacks to a normalized security score $SS \in [0, 10]$. Its logic prioritizes empirical resistance over theoretical parameters using a weighted aggregation. Algorithm 5 establishes the steps used to compute the security final score, defined in eq. 8.1.

$$SS = 10 \cdot P_{margin} \cdot (0.7 \cdot S_{att} + 0.2 \cdot S_{des} + 0.1 \cdot S_{ev}) \tag{8.1}$$

where S_{att} is the attack resistance of the cipher, obtained by normalizing the complexity of the best-known attack C_{best} against the 128-bit NIST baseline [185], $S_{att} = \min\left(1.0, \frac{\log_2(C_{best})}{128}\right)$. While lightweight standards sometimes accept 80-bit security, we treat 128 bits as the long-term gold standard, so a cipher with 2^{128} resistance scores 1.0 and one with 2^{64} operations scores 0.5. The security margin penalty P_{margin} is derived from the Security Margin principle [180] $M = (N_{total} - N_{broken})/N_{total}$; if attacks reach the full round count ($M \leq 0$), the cipher is deemed structurally compromised and we apply a hard penalty of 0.2, forcing its overall score below 2.0 even with a large key, distinguishing structurally flawed algorithms from merely small ones. The design strength

S_{des}, which quantifies the theoretical upper bound of security from the cipher's parameters (key size k and block size b), by normalizing them against the family-specific maxima (e.g., $k_{max} = 256$, $b_{max} = 128$): $S_{des} = 0.6 \cdot \frac{k}{k_{max}} + 0.4 \cdot \frac{b}{b_{max}}$. This captures resistance to generic time–memory trade-off attacks and rewards robust parameter choices even when no specific structural weaknesses are known. Finally, we assign attack severity $Sev \in (0,1)$ based on the practicality of the threat model. Baseline attacks ($S_{ev} = 1.0$) correspond to brute-force search or attacks requiring unrealistic data, such as a full codebook. Standard attacks ($S_{ev} = 0.8$–0.9) include differential and linear cryptanalysis, which serve as the primary benchmarks for block cipher security. Theoretical attacks ($S_{ev} < 0.5$) capture results with controversial or highly constrained models, such as biclique techniques or weak-key distinguishers, which may have limited impact on real-world confidentiality.

Weighting Rationale $(0.7, 0.2, 0.1)$**:** The final score aggregates S_{att}, S_{des}, and S_{ev}), using $W = (0.7, 0.2, 0.1)$, reflecting Kerckhoffs's Principle that security should come from the key and intrinsic algorithmic strength rather than obscurity [186]. The dominant weight $W_{att} = 0.7$ emphasizes empirical resistance, where a cipher with a large key but vulnerable to differential cryptanalysis must still be judged insecure. The secondary weight $W_{des} = 0.2$ recognizes that while larger parameters (k,b) enable higher security, they do not guarantee it, ensuring that small but secure ciphers like Simon-32/64 are not overshadowed by large but broken designs. Finally, $W_{sev} = 0.1$ serves as a fine-tuning term, distinguishing catastrophic breaks from more theoretical weaknesses without overriding the primary structural assessment.

Threshold Justification and Labeling: We calibrate the categorical labels by mapping the continuous security score SS to four classes based on Effective Security Bits (ESB). Variants with $SS < 2.0$ are labeled Broken, corresponding to negative security margins ($M \leq 0$) and validated against reduced-round attacks where full breaks occur at roughly

65–70% of the rounds. Low security ($2.0 \leq SS < 5.0$) corresponds to ESB of about 25–64 bits and covers valid but legacy-level ciphers such as Simon-32/64. Medium security ($5.0 \leq SS < 7.0$) corresponds to ESB of roughly 64–90 bits, meeting legacy standards (ISO/IEC 29192) but not modern High requirements. Finally, High security ($SS \geq 7.0$) corresponds to ESB above 90 bits, aligning with NIST High Security recommendations for constrained environments [185].

Calibration against Cryptographic Standards: To validate the semantic correctness of our scoring model, we benchmarked the generated labels against the key strength recommendations in NIST SP 800-57 Part 1 [187]. We validated the security scores against legacy and standard compliance. For example, our framework classifies Simon-32/64 (Score 4.13) as **Low**, aligning with NIST's classification of 64-bit keys as "disallowed for applying cryptographic protection" in modern systems, restricting them to legacy or non-critical use cases. Conversely, Simon-128/128 receives a score of 7.92 (**High**), consistent with the NIST requirement of ≥ 112 bits of security for federal information systems. Additionally, we verified the structural failure detection for the synthetic broken cipher variants. For example, synthetic Simon-32/64-Broken (reduced to 12 rounds) drops to a score of 1.52 (**Broken**). This confirms that our *Security Margin Penalty* successfully overrides the theoretical key size, correctly identifying structurally compromised variants even when they share parameters with valid designs.

8.1.2.4 Dataset Expansion via Cryptanalytic Injection A fundamental challenge in applying machine learning to cryptography is the extreme class imbalance; standard libraries provide mostly secure parameter sets, leaving the model with no ground truth for structural failures. To overcome this, we leveraged the parameterized nature of our Isabelle/HOL specifications to systematically generate synthetic variants that target the

Algorithm 5 Security Scoring Logic

Input: Block size b, Key size k, Rounds r, Broken rounds r_{br}, Best attack Complexity C, Attack Type T

Output: Final Score $SS \in [0, 10]$

1. Security Margin (M):
  if r_{br} *is defined* **then** $M \leftarrow (r - r_{br})/r$
else $M \leftarrow 0.5$
2. Design Strength (S_{des}):
  $S_{des} \leftarrow 0.6 \cdot (\frac{k}{256}) + 0.4 \cdot (\frac{b}{128})$ `// Normalized`
3. Attack Resistance (S_{att}):
  if *Attack Exists* **then**
    $Base \leftarrow \min(1.0, \frac{\log_2(C)}{128})$ `// 128-bit Baseline`
    if $M \leq 0$ **then**
      $Multiplier \leftarrow 0.2$ `// BROKEN: 80% Penalty`
    else
      $Multiplier \leftarrow 0.8 + (0.2 \cdot M)$ `// Margin Bonus`
    end
    $S_{att} \leftarrow Base \cdot Multiplier$
else
    $S_{att} \leftarrow \min(1.0, \frac{k}{128})$ `// Theoretical Max`
end
4. Aggregation:
$Sev \leftarrow \text{GetSeverity}(T)$ `// e.g., Differential=0.8`
$S_{raw} \leftarrow 0.7 \cdot S_{att} + 0.2 \cdot S_{des} + 0.1 \cdot Sev$
return $\min(10.0, S_{raw} \times 10.0)$

specific security classes defined in Section 8.1.2.3. By adjusting the round depth R and key width K while maintaining functional correctness, we created a balanced dataset that mirrors real-world cryptanalytic thresholds. We created synthetic "Broken" variants ($M \leq 0$) in order to teach the GNN to identify catastrophic structural failure; this consisted of generating variants where the round count (R) was reduced below the known cryptanalytic break point ($R \leq R_{broken}$). For example, while *Simon-32/64* is attacked up to 24 rounds via integral cryptanalysis [188], we therefore instantiated a broken Simon-32/64 broken variant with $R = 12$ rounds. For the complexity of the synthetic broken variants, since attack complexity scales exponentially with round depth, we assigned these broken variants a trivial complexity floor ($C \approx 2^{10}$), representing the minimal cost to distinguish the cipher from random permutation, well below the complexity of the full key schedule. Additionally, to better balance the classes and allow the model to distinguish between "broken" and "low" security levels, we generated variants with reduced margins. These configurations resist trivial breaks but lack the safety margin required for modern standards. For instance, we generated *GIFT-64-128-Low* with $R = 22$ rounds; while this resists the full 20-round linear attack, the margin is negligible. This category also includes standard legacy ciphers with constrained state sizes (e.g., *Simon-32/64* classified as Low because its 64-bit key restricts the maximum attack complexity to 2^{64}, insufficient for modern post-quantum standards), ensuring the model learns that weakness can stem from either insufficient depth (like rounds) or insufficient width (like block size).

8.1.3 Stage III: Dataset Augmentation and Stratified Sampling

Training a Graph Neural Network (GNN) on a finite set of our original 47 cipher variants poses two significant risks: overfitting to trivial patterns (e.g., memorizing variable names) and bias from class imbalance. To address this, we developed

a robust augmentation and sampling approach grounded in three core principles: *data anonymization*, *iso-functional graph augmentations*, and *variant-aware stratified sampling*.

Data Anonymization: Before augmentation, every AST is anonymized to prevent information leakage. This involves removing cipher-specific metadata and replacing function identifiers (e.g., simon_round) and variables (e.g., z_sequence) with generic tokens (e.g., FUNC_0, VAR_0). This approach ensures that the GNN predicts security solely based on the topological shape of the computation graph, avoiding the association of specific string literals with security labels.

Iso-functional Data Augmentation: To expand the dataset size without compromising the integrity of the ground truth labels, we developed a set of cryptographically-aware perturbations that ensure the mathematical function of the cipher remains unchanged (iso-functional). Our approach involves two levels of perturbation. First, under Topological Perturbation, with high impact, we introduce structural variety while preserving critical paths. This includes crypto-safe node dropout, where we randomly drop 10% of auxiliary nodes, such as type casting and intermediate bindings, while locking critical nodes like S-boxes, XORs, and Modular Additions to enhance the graph's robustness against sparsity. Additionally, we employ Edge Rewiring to create synthetic dependencies between non-interacting branches, simulating the complex data flows of optimized implementations and increasing the average degree without affecting the output ciphertext. Second, through algebraic rewriting, with medium impact, we leverage algebraic properties to generate visually distinct yet mathematically identical graphs. Moreover, we applied commutative reordering, where operations like $A \oplus B$ were randomly swapped with $B \oplus A$, and function inlining, which involves altering the graph depth by randomly inlining or abstracting helper functions. For each original variant V_{orig}, we generate $N = 10$ augmented copies,

resulting in a 10x expansion of the raw dataset while ensuring 100% label consistency. This augmentation was also used for data size sensitivity analysis, helping to determine which size is optimal for the model's training, before it starts overfitting.

Variant-Aware Stratified Data Sampling: While augmentation provides a substantial number of graphs, training on the entire distribution indiscriminately obscures the model's data efficiency. To empirically determine the optimal dataset size for structural generalization and to enforce class balance, we implemented a variant-aware sampling approach. This approach constructs a progressive series of datasets, denoted as $\mathscr{D}_k$, where k represents the target cardinality, or number of samples per variant. The sampling logic enforces three critical constraints to ensure experimental rigor. First, it helps perform data sensitivity analysis when training the GNN models. In fact, by varying k systematically (e.g., $k \in [2, 10]$), we isolate the impact of dataset size on model performance. This helps identify the specific threshold where the GNN transitions from overfitting to robust structural reasoning, regardless of the model architecture. Second, Class Stratification is implemented to address the asymmetry in the raw variant distribution, which was introduced during the synthetic generation phase. The sampler dynamically calibrates class quotas based on the minority class cardinality, ensuring that the dataset $\mathscr{D}_k$ maintains a balanced distribution, where N_{broken}, N_{low}, N_{med}, and N_{high} are approximately equal. This approach prevents decision boundary bias. Finally, Ground-Truth Prioritization is employed to maximize representativeness, with the selection algorithm enforcing a priority queue; the raw anonymized theory file is chosen first, and the remaining $k-1$ slots are filled through stochastic selection from an augmented pool. This process guarantees that every dataset $\mathscr{D}_k$ is anchored in formally verified logic.

The "Medium" class in our dataset (Table 8.4) is underrepresented compared to the extremes, reflecting the nature of lightweight ciphers. These designs typically exhibit

Table 8.4: Class Distribution Across Stratified Datasets. The sampling protocol maintains a stable equilibrium between Broken, Low, and Medium classes, while tracking the naturally more abundant High class.

Dataset($\mathscr{D}_k$)	Total	Broken	Low	Medium	High
$\mathscr{D}_1$ (raw)	47	11	11	10	15
$\mathscr{D}_2$	74	19	19	15	21
$\mathscr{D}_4$	153	39	39	30	45
$\mathscr{D}_6$	220	57	57	43	63
$\mathscr{D}_8$	298	77	76	59	86
$\mathscr{D}_{10}$	375	96	96	76	107

a binary security posture: they either maintain strong security ("High") or are deemed unsafe to use ("low"). Consequently, fewer configurations fall within the intermediate "Medium" state, and without any synthetic variant generation, this leads to a leads to a sparser population for the "Medium" class.

8.1.4 Stage IV: Neuro-Symbolic Hybrid Model

We introduce a Hybrid dual-stream architecture that combines topological reasoning with global parametric analysis to predict security labels from verified code. This approach captures the dual aspects of cryptographic security, relying on both local interaction patterns like S-box primitives and global design choices like key or block sizes.

Data Preprocessing and Feature Encoding Before training, the generated datasets/JSON files undergo a two-phase transformation to normalize inputs for the GNN models, *fit* and *transform*. In the *fit* phase, the encoder analyzed the entire training dataset to construct comprehensive dictionaries of unique categorical strings, establishing the feature space with 114 node feature dimensions and 13 edge types from the dataset. In the *transform* phase, each JSON file was processed to generate five tensors needed for the hybrid model. First, the node feature tensor, shaped as $[N, 114]$, where N is the number of nodes and 114 is the dimension of the node feature vector (Table 8.2). We concatenate one-hot encoded categorical roles with normalized numerical properties. Additionally, to

ensure numerical stability across graphs of varying sizes, we apply GraphNorm, which normalizes feature distributions per graph instance, mitigating the vanishing gradient problem common in deep GNNs. Second, the edge index tensor, defining the graph's topology, and shaped as $[2, M]$, where M is the number of edges, representing the source and target of each connection. Third, the edge attribute tensor, shaped as $[M, 1]$, contained the integer index for each of the 13 unique edge types, and provided the semantic attributes for each edge defined in the edge index tensor. Fourth, the unified PDV, shaped as $[1, 33]$, containing the high-level data of the cipher (see Table 8.3). Additionally, the PDV is normalized via Min-Max scaling to the $[0, 1]$ range, ensuring equal weighting with the graph embeddings. Fifth, the target label y, a single integer (e.g., 0 for "broken", 1 for "low", 2 for "medium", and 3 for "high") required for the model training. The preprocessed data is then used to train the hybrid model, composed of a GNN and MLP components, as described in the sections below.

8.1.4.1 Graph Neural Network (GNN) Component The GNN component is used for analyzing the cipher's AST, which is represented as a semantically-rich graph. Among all GNN architectures, we chose the Graph Attention Network (GAT), which employs a multi-head attention mechanism. Unlike a regular Graph Convolutional Network (GCN), which averages neighboring nodes uniformly, GAT utilizes a multi-head attention mechanism to assess the importance of different nodes. This is beneficial for our dataset because GAT effectively manages high-dimensional, sparse features by prioritizing critical elements while discarding noise, and it enhances performance on small, repetitive datasets by focusing on key cryptographic components that define the cipher's security. The GNN stream consists of one hidden GAT convolutional layer of 64 neurons, with 2 attention heads, followed by a ReLU activation and dropout. The final graph embedding is generated

via global mean pooling, producing a fixed-size vector of 88 dimensions that encapsulates the AST's learned structural properties.

8.1.4.2 Multi-Layer Perceptron (MLP) Component The MLP component is designed to process the cipher's high-level metadata encapsulated in the 33-features Protocol Design Vector (PDV) (Table 8.3). It acts as a context encoder, capturing global properties that are computationally expensive to infer from the graph topology alone. It consists of a single hidden layer of 64 neurons with ReLU activation, projecting the metadata into a complementary embedding space. This embedding complements the GNN's structural analysis. Additionally, due to the small dataset size and subsequently small batches, batch normalization layers were removed for the GNN and MLP components, as they could potentially introduce instability. Instead of batch normalization, we incorporated dropout and weight decay to reduce overfitting and improve generalization.

8.1.4.3 Integration and Classification Head To evaluate the cipher variants' security levels, the core of our approach relies on integrating and merging the GNN and MLP component streams. The final graph embedding vector from the GAT and the feature embedding from the MLP are concatenated and fed into a final MLP classifier head, which consists of 2 hidden layers of dimensions 128 and 64 with ReLU activation function, and a final softmax layer to compute the probability distribution of each variant across the four security classes. The detailed architecture of the hybrid model is shown in Fig. 8.3.

8.1.4.4 Optimization and Hyperparameter Search Our training approach involves a systematic grid search to optimize model architecture, hyperparameters, and dataset size, divided into two main areas. First, in the multi-scale dataset analysis, we perform a sensitivity analysis to enhance data augmentation, generating multiple balanced datasets of

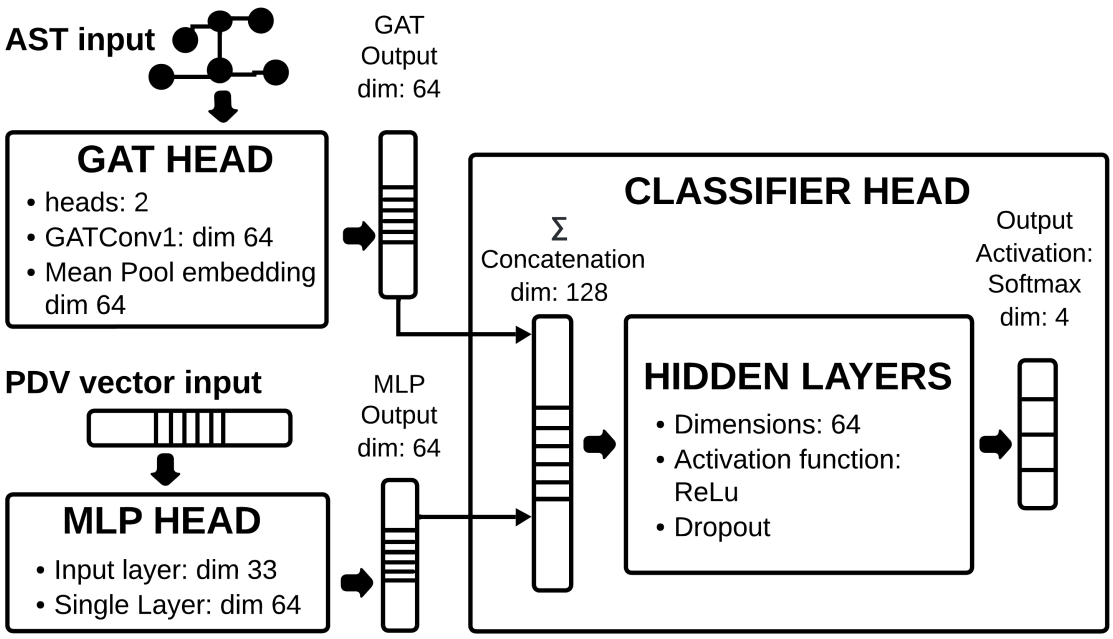

Figure 8.3: Proposed hybrid model, with GAT and MLP training on AST and PDV, merging their embeddings in a classifier head for security assessment.

varying sizes. This helped us identify the optimal level of augmentation, striking a balance between having enough data for generalization while avoiding overfitting. Second, we conduct a Hyperparameter Grid Search for each dataset size, exploring various model sizes and architectures, and tuning critical hyperparameters, including architectural depth/width, attention heads, regularization, and optimization settings (Table 8.5). We used the Stratified 5-Fold Cross-Validation across all experiments. This configuration allocates 80% of the data for training, ensuring the GNN receives sufficient structural diversity to converge even at lower dataset sizes, while maintaining a validation split robust enough to detect overfitting. The optimization process aims to minimize the Cross-Entropy Loss between the predicted softmax probabilities $\hat{y}$ and the ground-truth labels y. To enhance generalization and prevent memorization of the training data, we employed two control mechanisms. First, we used early stopping with checkpointing, monitoring the validation F1-Score, with

patience=20. Training was halted if there was no improvement for 20 epochs (patience). Additionally, to counter over-training in small data scenarios, the system restored model weights from the epoch with the best validation performance. The best performing model with the highest F1-score was then selected and evaluated on zero-shot ciphers, thereby ensuring a robust assessment of its generalization capabilities. We present our results and findings in the next section.

Table 8.5: Hyperparameters Tuned During GAT Grid Search

Parameter	**Range/Value**	**Parameter**	**Range/Value**
GAT Dims	$[120,72]$ - $[128,96]$	GAT Heads	$\{2,3,4\}$
MLP Dims	$[64,32]$ - $[64,48]$	Classifier depth	$\{2,3\}$-layers
Dropout rate	0.3 - 0.6	Weight decay	1e-4 - 5e-4
Learning rate	0.0002 – 0.001	-	-

8.2 Results and Discussion

Our simulation, covering formal verification to hybrid model training and evaluation, was executed on an HPC Cluster, of 16GB RAM. Formal verification used Isabelle/HOL 2025, while data creation, processing, model training, and evaluation were implemented in Python 3.13.5 with libraries like PyTorch Geometric, Scikit-learn, and Pandas. The hybrid model was trained on sampled datasets of different sizes, as detailed in Section 8.1.3 and Table 8.4, and fine-tuned through a grid search, as outlined in Section 8.1.4.4.

Dataset Split: To evaluate the model's generalization capabilities, we curated the split of the entire dataset (55 total variants: 47 for training, 8 for testing) based on a canonical teacher, complex student protocol, designed to maximize the structural distance between the training and the test sets. The training set was composed of the AST and PDV representatives for each cipher variant for SIMON, SIMECK, and XTEA (Feistel); SPECK

and LEA (ARX); PRESENT, GIFT and RECTANGLE (SPN); and SPARX (Hybrid). The test set was made of ciphers for zero-shot analysis, specifically SKINNY (SPN) and HIGHT (Generalized Feistel-ARX). The test set was also made of unseen variants from ciphers included in the training set, especially extreme-scale variants for GIFT, SPECK, SIMON, and SPARX on key and block size. The split between the training and the test sets follows a canonical teacher, complex student protocol, to maximize the structural distance between both sets, and improve the models' generalization capabilities.

8.2.1 Dataset Analysis

A core contribution of our work is creating a high-dimensional, semantically-rich dataset for cryptographic analysis. We conducted a sensitivity analysis to find the optimal training dataset by using various well-balanced sizes across the four security classes. After the full training and evaluation pipeline across all the datasets, further discussed in Section 8.2.1.3, we identified `Dataset-4` ($\mathscr{D}_4$) as the optimal trade-off between model performance (F1-score) and security level prediction of unseen ciphers. All subsequent analyses are performed on $\mathscr{D}_4$, which contains a balanced mix of 153 cipher families, to evaluate the underlying structural features contributing to the model's performance.

8.2.1.1 Cipher Family Fingerprints Using high-level graph metrics, Table 8.6 shows that Feistel, ARX, and SPN families have distinct, learnable fingerprints, making them statistically separable. SPN cipher PRESENT is the largest and most complex, with $\sim$280 nodes, while ARX cipher Speck is more compact at $\sim$175 nodes. Additionally, each cipher family is characterized by unique cryptographic components: Feistel features the feistel_f_function and modular_operation; ARX, the diffusion and modular_addition; and SPN, the sbox_substitution and a high rate of linear mixing.

Using high-level graph metrics (Table 8.6), we observe that the cipher families possess distinct, statistically learnable fingerprints. For the Feistel family (SIMON, SIMECK, XTEA), the average nodes are 173.2 with 272.3 edges, resulting in computationally intensive graphs. The deep computation chains are predominantly influenced by `linear_mixing` and `modular_operation`, necessary for the non-linear function F. In contrast, ARX ciphers (SPECK, *Lea*, *Hight*) have more compact graphs, averaging around 93.2 nodes, yet they exhibit a higher average degree of 3.32 due to the dense connections from Modular Addition and rotations. For the SPN family (PRESENT, GIFT, SKINNY), Substitution-Permutation Networks are characterized by approximately 122.3 nodes and unique semantic roles, featuring explicit sbox_substitution nodes. Their lower average degree of 3.02 reflects a layered structure with defined connections between S-box and Permutation layers, in contrast to the recursive feedback loops in ARX. In comparison, the Hybrid paradigm (Sparx) serves as a structural bridge with around 161.8 nodes and a higher average degree of 3.34, blending characteristics of Feistel architectures and ARX-like connectivity.

Table 8.6: $\mathscr{D}_4$ Cipher family Fingerprints Analysis

Family	Avg. Nodes	Avg. Edges	Avg. Deg.	Dominant Roles (Ct)
Feistel	173.2	272.3	3.13	`linear_mixing` (369) `modular_op` (271)
ARX	93.2	155.4	3.32	`linear_mixing` (193) `modular_add` (87)
SPN	122.3	195.4	3.02	`linear_mixing` (97) `sbox_sub` (45)
Hybrid	161.8	271.7	**3.34**	`modular_op` (66) `linear_mixing` (27)

8.2.1.2 Component Distribution Analysis The structural diversity in $\mathscr{D}_4$ highlights distinct family signatures that potentially enhance the model's classification performance. Notably,

SPN graphs are characterized by a high proportion of variable nodes (49.3%), which contrasts with the 32% found in Feistel structures, reflecting the significant use of substitution layers. In Hybrid ciphers like Sparx, there is a reliance on literals (17.4%), indicative of constant-heavy affine transformations. Additionally, the distribution of cryptographic operations reveals a pronounced non-linearity gap: ARX architectures completely lack component nodes, depending instead on generic non-linear operations (4.5%), while SPN architectures are marked by the presence of explicit component nodes (0.92%) for S-box lookups, with minimal loose non-linear arithmetic. This clear distinction enables the model to effectively differentiate between the complex operations in ARX and SPN lookup tables with high accuracy.

8.2.1.3 HybridGAT Performance Evaluation Using the grid search hyperparameters, we trained different Hybrid GAT configurations across progressive dataset sizes ($\mathscr{D}_1$ to $\mathscr{D}_{10}$) to identify the minimum data required for structural reasoning. As summarized in Table 8.7, performance followed a distinct trajectory: The performance trajectory outlined in Table 8.7 reveals key findings regarding model efficiency and reasoning capabilities. Initially, with minimal data ($\mathscr{D}_{1-3}$), the model struggles, reflected in a low F1-validation score and poor generalization to other ciphers like SKINNY or SPECK. However, at $\mathscr{D}_4$ it converges (F1-score=0.94), indicating effective learning of topological rules. Increasing the data to $\mathscr{D}_{10}$ enhances in-distribution accuracy to 0.99 but does not improve zero-shot performance, reaffirming the sufficiency of $\mathscr{D}_4$. The model excels in reasoning for SPN architectures, accurately distinguishing between security tiers in the SKINNY family, notably SKINNY 64/64, classified as "Medium", SKINNY 64/128, classified as "High", and synthetic broken SKINNY 64/128 variant, correctly classified as "Broken". HybridGAT however, faces

challenges with the HIGHT family, consistently misclassifying HIGHT 64/128 variant as "High" security, highlighting its limitations in detecting specific parametric weaknesses.

Table 8.7: HybridGAT Convergence and Generalization Results

Metric / Cipher (Count)	$\mathscr{D}_{1-3}$	$\mathscr{D}_4$	$\mathscr{D}_{10}$
F1-Score	0.67 ± 0.14	0.94 ± 0.05	0.99 ± 0.01
SKINNY (3)	1/3	**3/3**	3/3
HIGHT-128 (1)	0/1	0/1	0/1
GIFT-128 (1)	1/1	**1/1**	1/1
SIMON-128 (1)	0/1	**1/1**	1/1
SPECK-256 (1)	0/1	**1/1**	1/1
SPARX-128 (1)	0/1	**1/1**	1/1

8.2.2 Evaluating Other Candidate ML Models

We evaluated the effectiveness of our proposed HybridGAT (HGAT) through a comparative analysis with two baseline ML models and two hybrid GNN models that utilize GCN and TransformerConv layers. This evaluation aimed to determine if the hybrid GNN-MLP architecture outperforms simpler models and whether the attention mechanism in HybridGAT enhances generalization over traditional GNNs.

For the two hybrid GNN models, HybridGCN (HGCN) and HybridTrans (HTrans), we developed models using GCNConv and TransformerConv convolutional layers, respectively. GCN offers a simpler architecture than GAT, well-suited for small datasets. It does not use an attention mechanism and considers all neighboring nodes to be equally important. It is therefore adequate to evaluate whether the nodes' importance is essential when assessing the ciphers' security. Additionally, while GAT and TransformerConv both use a self-attention mechanism, unlike GCN or GAT, which can only use the AST edge indexes, TransformerConv incorporates the ASTs' edge features into its attention

calculations. This helps the model learn the types of operations and relationships among the nodes. For the baseline ML models, KNN, SVM, RandomForest (RF), and XGBoost (XGB) models were used. We flattened each graph into a single, high-dimensional feature vector, which was created by concatenating the node features and the Unified PDV vector components. The first component was a concatenation of the mean, standard deviation, and maximum value for each of the 114 node features, resulting in 342 dimensions. The second component was the complete 33-feature PDV vector, contributing an additional 33 dimensions. This process resulted in a single 375-dimensional vector for each cipher. The objective was to test whether a statistical summary of a graph is enough to adequately represent its true topological structure and train baseline ML models to assess the cipher's security levels. Models were trained on $\mathscr{D}_4$, and tested on the test set, including zero-shot tests; the results are shown in Table 8.8. KNN and SVM perform poorly on the training

Table 8.8: Comparative Analysis of Zero-Shot Generalization.

Model	F1	SKINNY			GIFT	HIGHT	Scale
Label	-	*High*	*Med*	*Brk*	*High*	*Med*	*High*
KNN	0.37	✗	✗	✗	✓	✓	✓
SVM	0.50	✗	✗	✓	✓	✗	✓
RF	0.72	✓	✓	✓	✓	✗	✓
XGB	0.84	✓	✗	✓	✓	✗	✓
HGCN	0.93	✓	✗	✓	✓	✗	✓
HTrans	0.91	✗	✓	✓	✓	✗	✓
HGAT	0.94	✓	✓	✓	✓	✗	✓

set (F1 $\leq$ 0.5) and on the test set for zero-shot generalization. While ensemble baselines like Random Forest and XGBoost achieve higher validation accuracy than other baseline ML models, they also fail to generalize to unseen architectures, erroneously classifying valid HIGHT 64/128 variant as "Broken", and Skinny 64/64 as "Low". Among the graph models, the GAT demonstrates superior topological granularity: whereas the GCN overestimates the security of the parametrically weak SKINNY-64-64 (predicting "High")

and the Transformer underestimates it, the GAT is the only architecture to correctly identify the "Medium" security level of Skinny 64/64. This confirms that the GAT's attention mechanism effectively weighs specific weak paths (like reduced state size) against global topological strength, a critical requirement for automated vulnerability detection. Additionally, all models correctly identify the security levels of Sparx 128/128, Speck 128/256, Simon 128/128 (Scale), showing that the models correctly learned from

The HIGHT cipher paradox: While Hybrid GNN models (HybridGAT, HybridGCN, HybridTrans) models showed consistent generalization across SPN and standard ARX architectures, the persistent misclassification of the *HIGHT* family reveals a critical boundary of graph-based cryptanalysis. Although standard HIGHT 64/128 is cryptanalytically classified as Medium security due to known attacks penetrating up to 28 of 32 rounds [189], our hybrid GNN models consistently predicted it as High with $> 98\%$ confidence. This discrepancy stems from HIGHT's 32-round generalized Feistel Network, which generates a dense, deeply interconnected topology indistinguishable from secure primitives like SIMON, masking the specific algebraic weaknesses in its rotation constants that compromise its margin. Unlike SKINNY 64/64, where the security flaw (reduced state size) is explicitly encoded in the graph attributes, HIGHT's vulnerability is parametric rather than topological. This finding underscores the role of GNNs for such tasks, serving as estimators of algorithmic complexity and structural soundness, but must be complemented by symbolic verification to detect mathematically subtle, non-structural flaws.

8.2.3 Glass-Box Interpretability Analysis

We analyzed the model on unanonymized cipher variants to improve interpretability, and selected SIMON 64/128, GIFT 128/128, SKINNY 64/64 and HIGHT 64/128. As

summarized in Table 8.9, the model adapts its focus to the structural definitions of each cipher family: for Feistel fingerprinting (Simon), it places maximum weight on the `simon_F_function` and its core operations (`ROTL`, `XOR`), correctly identifying the round function $f(x)$ as the main source of cryptographic strength; for the unseen ARX cipher HIGHT, attention shifts to `hight_enc_round`, especially `LIST_INDEX` (key access) and `ADD` (modular addition), mirroring ARX's reliance on modular arithmetic and key whitening. For the SPN cipher GIFT, the model prioritizes `gift_sbox_layer` (substitution) and `gift_perm_layer` (permutation), successfully mapping graph nodes to the classical components of a Substitution–Permutation Network.

Quantitative Distribution: This semantic alignment is supported by the aggregated attention distribution. For ARX ciphers, the model allocates $\sim 39\%$ of its attention mass to variable nodes (tracking state updates). For SPN ciphers, focus is more evenly distributed across operation layers ($\sim 30\%$). This dynamic reallocation validates that the model has learned the semantic definitions of cryptographic security.

Table 8.9: HybridGAT Unanonymized Top Semantic Nodes by Attention Score for SIMON, HIGHT and GIFT

Simon (Feistel)	**HIGHT (ARX)**	**GIFT (SPN)**
`simon_F_function`	`hight_enc_round`	`gift_encrypt_round`
`XOR` (Mix)	`LIST_INDEX` (Key)	`gift_sbox_layer`
`AND` (Non-Linear)	`ADD` (Mod Op)	`gift_perm_layer`
`ROTL` (Diffusion)	`XOR` (Mix)	`gift_add_round_key`

8.3 Conclusion

The proliferation of custom, non-standard lightweight ciphers in IoT systems creates significant security risks, as manual cryptanalysis cannot scale with automated design

tools. This work mitigates that gap by introducing the first framework that links formal verification with geometric deep learning. We present a methodology that converts Isabelle/HOL specifications into semantically rich Abstract Syntax Trees (ASTs) and Protocol Description Vectors (PDVs), forming a rigorous dataset for automated security assessment. On top of this, we design HybridGAT, a neuro-symbolic architecture that combines topological reasoning with parametric feature analysis, achieving an F1-score of 0.94 and outperforming ensemble baselines and standard GNNs. HybridGAT further exhibits zero-shot generalization to unseen SPN ciphers (e.g., SKINNY) and variant (e.g., GIFT 128/128, SPARX 128/128) correctly identifies the "Medium" security level of SKINNY-64-64, with glass-box analysis confirming that its attention aligns with cipher semantics across Feistel and ARX families. Our approach has three main limitations. The core dataset of 55 primitive ciphers remains small and relies on augmentation; the model acts as a structural filter rather than a full cryptanalytic tool, missing purely parametric weaknesses such as the HIGHT rotation constants; and the feature set depends on manual domain engineering. Future work will expand the benchmark to include stream ciphers and post-quantum primitives and extend the prediction head to a multi-label diagnostic tool that identifies concrete vulnerability types (e.g., weak diffusion, insufficient nonlinearity), enabling more actionable feedback for automated cipher design.

CHAPTER NINE

CONCLUSION

This chapter synthesizes the key findings and conclusions and outlines the future work stemming from the body of research presented in this dissertation. The work progressed from an initial socio-technical investigation of the trust gap among users and policies in autonomous systems to the development of sophisticated frameworks for predictive resource optimization, quantitative risk assessment, and deployment feasibility, culminating in a neuro-symbolic architecture for the automated formal specification translation and security level evaluation of cryptographic primitives.

This chapter integrates the central findings and conclusions of this dissertation and identifies directions for future research arising from the work. The dissertation evolves from an initial socio-technical analysis of the trust gap between users and policy frameworks in autonomous systems to the design of advanced methods for predictive resource optimization, quantitative risk analysis, and deployment feasibility. It culminates in a neuro-symbolic architecture that enables automated formal specification translation and security-level assessment of cryptographic primitives.

9.1 Key Findings

The research yielded significant and quantifiable improvements across all stages of the assurance pipeline. In the initial exploration of socio-technical trust (RO1) [25], the analysis of unstructured social media data (Twitter/Reddit) versus government policy documents revealed a distinct divergence. While user sentiment was predominantly negative across the board, and driven by uncertainties regarding black-box decision-making, government policies were found to focus on liability and

hardware redundancy. This mismatch confirms a gap where regulatory frameworks are not synchronized with the foundational users' concerns, suggesting that acceptance and trust cannot be immediate without additional regulations.

The investigation into security assessment (RO2) produced a major finding. The risk assessment framework, based on Bayesian Attack Graph (BAG) and complex probabilities, successfully modeled the constructive and destructive interference of concurrent attacks, a capability absent in traditional risk models, and handled cyclic backtracking without altering the network architecture [26]. Additionally, the feasibility analysis (RO3) of 20 state-of-the-art ML-based IDS models using the novel Custom Feasibility Score (CFS) revealed a critical trade-off. While simple Category-2 models (e.g., Decision Trees) were consistently feasible, complex Category-1 models (e.g., VGG16, LSTM) often incurred high computational overhead that rendered them impractical for real-time Edge deployment, despite their higher theoretical accuracy [27].

Finally, this research produced a neuro-symbolic framework for automated formal translation and security level evaluation (RO4) for lightweight block ciphers. The Tiered Isomorphic Alignment (TIA) approach [28] enabled Large Language Models (LLMs) to achieve a 9.7 times increase in formal Syntax Validity and a 2.9 times increase in cryptographic Value Consistency compared to baseline LLMs, showing that hierarchical alignment is crucial for cross-paradigm translation. To support security evaluation, the CryptoGraph framework achieved an F1-score of 0.94 in classifying cipher security levels, outperforming ensemble baselines such as Random Forest and XGBoost. Glass-Box analysis further showed that the model correctly focused on key cryptographic components (e.g., S-boxes in GIFT, rotation operations in Simon), confirming that it learned meaningful structural security features [29].

9.2 Conclusions

This dissertation has successfully demonstrated the progressive development and integration of socio-technical trust, resource optimization, and formal assurance to address critical "Assurance Gaps" in the IoT ecosystem.

The initial phase of this research established that the barrier to AV adoption is not only technological but socio-technological. Our analysis concluded that while literature often cites positive sentiment, a deeper dive into non-event-driven data reveals persistent negative sentiment driven by uncertainty. We concluded that an Explainable AI (XAI) framework is not a luxury but a necessity to provide transparency and align user expectations with regulatory reality.

The subsequent research focused on the dual challenges of strategic risk quantification and tactical defense feasibility. We concluded that feasibility is a multi-dimensional constraint; a model that is feasible under normal conditions may become impractical during a high-traffic attack. The proposed CFS metric proved that evaluating IDS models solely on prediction accuracy is insufficient for safety-critical automotive environments, where resource scarcity can lead to failure.

Finally, to address the foundational correctness of the underlying software, we proposed a neuro-symbolic pipeline focused on cryptographic lightweight block ciphers. We concluded that while baseline LLMs struggle with the strict logic of formal code, notably Isabelle/HOL, our TIA framework transforms them into capable formalization assistants. By coupling this with Graph Attention Networks, extracting ciphers' representative AST and high-level vectors, we established that geometric deep learning can effectively serve as an automated tool for filtering out structurally weak ciphers before costly manual cryptanalysis is attempted.

9.3 Future Work

The findings from this dissertation open several promising avenues for future research, aiming to enhance the robustness, automation, and real-world applicability of the proposed frameworks.

A primary thrust of future work will be the development of the proposed Explainable AI (XAI) interface (RO1). This will involve creating a user-facing platform that maps specific user concerns to mitigating government policies in real-time, allowing for a dynamic assessment of how information transparency affects public trust levels.

Another significant direction is the expansion of the CryptoGraph framework (RO4). While it successfully generalized to SPN, ARX, and Feistel architectures, the "HIGHT Paradox" revealed limitations in detecting purely parametric weaknesses (e.g., weak rotation constants). Future work will integrate a multi-label diagnostic tool to identify specific vulnerability types (e.g., insufficient diffusion) and extend the dataset to include post-quantum primitives.

Finally, the integration of Generative AI into the Risk Assessment framework (RO2) will be explored. We aim to use GenAI to improve the contextual understanding of attack graphs, aiding security administrators in identifying zero-day attack paths that traditional static databases might miss. This aligns with our broader goal of moving from static, manual security assessments to dynamic, automated, and continuous security.

REFERENCES

[1] M. Müter, A. Groll, and F. C. Freiling, "A structured approach to anomaly detection for in-vehicle networks," in *2010 Sixth International Conference on Information Assurance and Security*, pp. 92–98, 2010.

[2] T. Alam, "A reliable communication framework and its use in internet of things (iot)," vol. 3, 05 2018.

[3] H. K. Bharadwaj, A. Agarwal, V. Chamola, N. R. Lakkaniga, V. Hassija, M. Guizani, and B. Sikdar, "A review on the role of machine learning in enabling iot based healthcare applications," *IEEE Access*, vol. 9, pp. 38859–38890, 2021.

[4] V. P. Kour and S. Arora, "Recent developments of the internet of things in agriculture: A survey," *IEEE Access*, vol. 8, pp. 129924–129957, 2020.

[5] A. Oseni, N. Moustafa, G. Creech, N. Sohrabi, A. Strelzoff, Z. Tari, and I. Linkov, "An explainable deep learning framework for resilient intrusion detection in iot-enabled transportation networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 1, pp. 1000–1014, 2023.

[6] P. Radanliev, D. De Roure, R. Nicolescu, M. Huth, and O. Santos, "Artificial intelligence and the internet of things in industry 4.0," *CCF Transactions on Pervasive Computing and Interaction*, vol. 3, pp. 329–338, 2021.

[7] H. Rajab and T. Cinkelr, "Iot based smart cities," in *2018 International Symposium on Networks, Computers and Communications (ISNCC)*, pp. 1–4, 2018.

[8] T. Cohen, J. Stilgoe, S. Stares, N. Akyelken, C. Cavoli, J. Day, J. Dickinson, V. Fors, D. Hopkins, G. Lyons, N. Marres, J. Newman, L. Reardon, N. Sipe, C. Tennant, Z. Wadud, and E. Wigley, "A constructive role for social science in the development of automated vehicles," *Transportation Research Interdisciplinary Perspectives*, vol. 6, p. 100133, 2020.

[9] P. Coppola and F. Silvestri, "Autonomous vehicles and future mobility solutions," in *Autonomous Vehicles and Future Mobility* (P. Coppola and D. Esztergár-Kiss, eds.), pp. 1–15, Elsevier, 2019.

[10] S. Sindi and R. Woodman, "Autonomous goods vehicles for last-mile delivery: Evaluation of impact and barriers," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1–6, 2020.

[11] J. Dokic, B. Müller, and G. Meyer, "European roadmap smart systems for automated driving," 04 2015.

[12] V. Pyrialakou, C. Gkartzonikas, J. Gatlin, and K. Gkritza, "Perceptions of safety on a shared road: Driving, cycling, or walking near an autonomous vehicle," *Journal of Safety Research*, vol. 72, pp. 249–258, 2020.

[13] F. Hussain, S. A. Hassan, R. Hussain, and E. Hossain, "Machine learning for resource management in cellular and iot networks: Potentials, current solutions, and open challenges," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1251–1275, 2020.

[14] T. Dirsehan and C. Can, "Examination of trust and sustainability concerns in autonomous vehicle adoption," *Technology in Society*, vol. 63, p. 101361, 2020.

[15] L. M. Hulse, H. Xie, and E. R. Galea, "Perceptions of autonomous vehicles: Relationships with road users, risk, gender and age," *Safety Science*, vol. 102, pp. 1–13, 2018.

[16] Y. Ding, R. Korolov, W. (Al) Wallace, and X. C. Wang, "How are sentiments on autonomous vehicles influenced? an analysis using twitter feeds," *Transportation Research Part C: Emerging Technologies*, vol. 131, p. 103356, 2021.

[17] T. McDonald, B. Huang, R. Wei, H. Alambeigi, C. Arachie, A. Smith, and J. Jefferson, "Data mining twitter to improve automated vehicle safety," Feb 2021. Tech Report.

[18] N. Poolsappasit, R. Dewri, and I. Ray, "Dynamic security risk management using bayesian attack graphs," *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 1, pp. 61–74, 2011.

[19] A. Sen, K. Fletcher, and S. Madria, "A secure user-centric framework for dynamic service provisioning in iot environments," in *2019 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/13th IEEE International Conference on Big Data Science And Engineering (TrustCom/BigDataSE)*, pp. 334–341, IEEE, 2019.

[20] D. Behbehani, N. Komninos, K. Al-Begain, and M. Rajarajan, "Cloud enterprise dynamic risk assessment (cedra): a dynamic risk assessment using dynamic bayesian networks for cloud environment," *Journal of Cloud Computing*, vol. 12, no. 1, p. 79, 2023.

[21] M. Flores, D. Heredia, R. Andrade, and M. Ibrahim, "Smart home iot network risk assessment using bayesian networks," *Entropy*, vol. 24, no. 5, p. 668, 2022.

[22] J.-P. Aumasson, *Serious cryptography: a practical introduction to modern encryption*. No Starch Press, Inc, 2024.

[23] M. Rana, Q. Mamun, and R. Islam, "Lightweight cryptography in iot networks: A survey," *Future Generation Computer Systems*, vol. 129, pp. 77–89, 2022.

[24] T. Nipkow, M. Wenzel, and L. C. Paulson, *Isabelle/HOL: a proof assistant for higher-order logic*. Springer, 2002.

[25] V. C. Wakam Younang, J. Yang, L. G. Jacuinde, and A. Sen, "A comparative analysis of user's concerns and government policies on autonomous vehicles," in *International Conference on Internet of Things*, pp. 47–61, Springer, 2022.

[26] V. C. Wakam Younang and A. Sen, "Security risk assessment using bayesian attack graphs and complex probabilities for large scale iot applications," *IEEE Transactions on Dependable and Secure Computing*, vol. 22, no. 6, pp. 7360–7371, 2025.

[27] V. C. Wakam Younang, O. Oyeniyi, R. Ayan, and A. Sen, "Evaluating the feasibility of machine learning approaches in anomaly-based intrusion detection system for in-vehicle networks," *ACM Transactions on Embedded Computing Systems*, pp. 1–38, June 2025, Under Submission.

[28] V. C. Wakam Younang and A. Sen, "Tiered isomorphic alignment for cross-paradigm logic: Inducing formal grammars from imperative code," *International Conference of Machine Learning*, pp. 1–10, January 2026, Under Submission.

[29] V. C. Wakam Younang and A. Sen, "Cryptograph: A neuro-symbolic automated framework for lightweight block ciphers security evaluation," *USENIX Security Symposium*, pp. 1–16, February 2026, Under Submission.

[30] P. Thirumavalavasethurayar and T. Ravi, "Implementation of replay attack in controller area network bus using universal verification methodology," in *2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS)*, pp. 1142–1146, IEEE, 2021.

[31] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, vol. 2015, no. S 91, pp. 1–91, 2015.

[32] S. Lee, W. Choi, I. Kim, G. Lee, and D. H. Lee, "A comprehensive analysis of datasets for automotive intrusion detection systems," *Computers, Materials & Continua*, vol. 76, no. 3, pp. 3413–3442, 2023.

[33] M. Marchetti and D. Stabili, "Anomaly detection of can bus messages through analysis of id sequences," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1577–1583, 2017.

[34] K. Iehira, H. Inoue, and K. Ishida, "Spoofing attack using bus-off attacks against a specific ecu of the can bus," in *2018 15th IEEE annual consumer communications & networking conference (CCNC)*, pp. 1–4, IEEE, 2018.

[35] M. Almehdhar, A. Albaseer, M. A. Khan, M. Abdallah, H. Menouar, S. Al-Kuwari, and A. Al-Fuqaha, "Deep learning in the fast lane: A survey on advanced intrusion detection systems for intelligent vehicle networks," *IEEE Open Journal of Vehicular Technology*, vol. 5, pp. 869–906, 2024.

[36] O. Avatefipour and H. Malik, "State-of-the-art survey on in-vehicle network communication (can-bus) security and vulnerabilities," *CoRR*, vol. abs/1802.01725, 2018.

[37] K. Scarfone, P. Mell, *et al.*, "Guide to intrusion detection and prevention systems (idps)," *NIST special publication*, vol. 800, no. 2007, p. 94, 2007.

[38] T. C. Dönmez, "Anomaly detection in vehicular can bus using message identifier sequences," *IEEE Access*, vol. 9, pp. 136243–136252, 2021.

[39] S. Sharmin, H. Mansor, A. F. A. Kadir, and N. A. Aziz, "Benchmarking frameworks and comparative studies of controller area network (can) intrusion detection systems: A review," *Journal of Computer Security*, vol. 32, no. 5, pp. 477–507, 2024.

[40] O.-R. A. D. O. Committee, *Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles*. SAE international, 2021.

[41] L. Muñoz-González and E. C. Lupu, "Bayesian attack graphs for security risk assessment," in *IST-153 Workshop on Cyber Resilience*, 2016.

[42] FIRST.org, "Common vulnerability scoring system v3.1: Specification document," tech. rep., Forum of Incident Response and Security Teams, 2019.

[43] K. McKay, L. Bassham, M. Sönmez Turan, and N. Mouha, "Report on lightweight cryptography," tech. rep., National Institute of Standards and Technology, 2016.

[44] Q. Hussain, W. K. Alhajyaseen, M. Adnan, M. Almallah, A. Almukdad, and M. Alqaradawi, "Autonomous vehicles between anticipation and apprehension: Investigations through safety and security perceptions," *Transport Policy*, vol. 110, pp. 440–451, 2021.

[45] V. C. Wakam Younang, "https://github.com/victorine07/av-social-mefia-analysis," Sept. 2022. Github link containing the documents, code and link source used for our analysis.

[46] J. Jefferson and A. D. McDonald, "The autonomous vehicle social network: Analyzing tweets after a recent tesla autopilot crash," *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 63, no. 1, pp. 2071–2075, 2019.

[47] X. Chen, H. Zeng, H. Xu, and X. Di, "Sentiment analysis of autonomous vehicles after extreme events using social media data," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pp. 1211–1216, 2021.

[48] R. Sadiq and M. Khan, "Analyzing self-driving cars on twitter," 2018.

[49] U. D. Gandhi, P. Malarvizhi Kumar, G. Chandra Babu, and G. Karthick, "Sentiment analysis on twitter data by using convolutional neural network (cnn) and long short term memory (lstm)," *Wireless Personal Communications*, May 2021.

[50] C. Phillips and L. P. Swiler, "A graph-based system for network-vulnerability analysis," in *Proceedings of the 1998 workshop on New security paradigms*, pp. 71–79, 1998.

[51] S. Mauw and M. Oostdijk, "Foundations of attack trees," in *International Conference on Information Security and Cryptology*, pp. 186–198, Springer, 2005.

[52] I. Ray and N. Poolsapassit, "Using attack trees to identify malicious attacks from authorized insiders," in *European Symposium on Research in Computer Security*, pp. 231–246, Springer, 2005.

[53] J. Lee, H. Lee, and H. P. In, "Scalable attack graph for risk assessment," in *2009 International Conference on Information Networking*, pp. 1–5, IEEE, 2009.

[54] L. Gallon and J. J. Bascou, "Using cvss in attack graphs," in *2011 Sixth International Conference on Availability, Reliability and Security*, pp. 59–66, IEEE, 2011.

[55] E. FIRST, "Common vulnerability scoring system version 3.1: Specification document," 2019.

[56] S. Unger, E. Arzoglou, M. Heinrich, D. Scheuermann, and S. Katzenbeisser, "Risk assessment graphs: Utilizing attack graphs for risk assessment," *arXiv preprint arXiv:2307.14114*, 2023.

[57] E.-M. Kalogeraki, S. Papastergiou, and T. Panayiotopoulos, "An attack simulation and evidence chains generation model for critical information infrastructures," *Electronics*, vol. 11, no. 3, p. 404, 2022.

[58] U. Garg, G. Sikka, and L. K. Awasthi, "Empirical analysis of attack graphs for mitigating critical paths and vulnerabilities," *Computers & Security*, vol. 77, pp. 349–359, 2018.

[59] J. Wang, M. Neil, and N. Fenton, "A bayesian network approach for cybersecurity risk assessment implementing and extending the fair model," *Computers Security*, vol. 89, p. 101659, 2020.

[60] P. Xie, J. H. Li, X. Ou, P. Liu, and R. Levy, "Using bayesian networks for cyber security analysis," in *2010 IEEE/IFIP international conference on dependable systems & networks (DSN)*, pp. 211–220, IEEE, 2010.

[61] N. d'Ambrosio, G. Perrone, and S. P. Romano, "Including insider threats into risk management through bayesian threat graph networks," *Computers & Security*, vol. 133, p. 103410, 2023.

[62] H. Y. L. B. Gao, Ni, "Exploring attack graphs for security risk assessment: A probabilistic approach," *Wuhan University Journal of Natural Sciences*, vol. 23, 2018.

[63] M. Alhomidi and M. Reed, "Attack graph-based risk assessment and optimisation approach," *International Journal of Network Security & Its Applications*, vol. 6, no. 3, p. 31, 2014.

[64] M. Frigault and L. Wang, "Measuring network security using bayesian network-based attack graphs," in *2008 32nd Annual IEEE International Computer Software and Applications Conference*, pp. 698–703, 2008.

[65] R. Dantu, K. Loper, and P. Kolan, "Risk management using behavior based attack graphs," in *International Conference on Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004.*, vol. 1, pp. 445–449 Vol.1, 2004.

[66] Y. Liu and H. Man, "Network vulnerability assessment using bayesian networks," in *Data mining, intrusion detection, information assurance, and data networks security 2005*, vol. 5812, pp. 61–71, SPIE, 2005.

[67] F.-X. Aguessy, O. Bettan, G. Blanc, V. Conan, and H. Debar, "Hybrid risk assessment model based on bayesian networks," in *International Workshop on Security*, pp. 21–40, Springer, 2016.

[68] A. Sen and S. Madria, "Risk assessment in a sensor cloud framework using attack graphs," *IEEE Transactions on Services Computing*, vol. 10, no. 6, pp. 942–955, 2016.

[69] M. Zolanvari, M. A. Teixeira, L. Gupta, K. M. Khan, and R. Jain, "Machine learning-based network vulnerability analysis of industrial internet of things," *IEEE internet of things journal*, vol. 6, no. 4, pp. 6822–6834, 2019.

[70] S. Mouti, S. K. Shukla, S. A. Althubiti, M. A. Ahmed, F. Alenezi, and M. Arumugam, "Cyber security risk management with attack detection frameworks using multi connect variational auto-encoder with probabilistic bayesian networks," *Computers and Electrical Engineering*, vol. 103, p. 108308, 2022.

[71] P. Radanliev, D. De Roure, C. Maple, J. R. Nurse, R. Nicolescu, and U. Ani, "Ai security and cyber risk in iot systems," *Frontiers in Big Data*, vol. 7, p. 1402745, 2024.

[72] C. Dezan, S. Zermani, and C. Hireche, "Embedded bayesian network contribution for a safe mission planning of autonomous vehicles," *Algorithms*, vol. 13, no. 7, p. 155, 2020.

[73] D. Dominic, S. Chhawri, R. M. Eustice, D. Ma, and A. Weimerskirch, "Risk assessment for cooperative automated driving," in *Proceedings of the 2nd ACM workshop on cyber-physical systems security and privacy*, pp. 47–58, 2016.

[74] B. Sheehan, F. Murphy, M. Mullins, and C. Ryan, "Connected and autonomous vehicles: A cyber-risk classification framework," *Transportation research part A: policy and practice*, vol. 124, pp. 523–536, 2019.

[75] A. Bezemskij, G. Loukas, D. Gan, and R. J. Anthony, "Detecting cyber-physical threats in an autonomous robotic vehicle using bayesian networks," in *2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pp. 98–103, IEEE, 2017.

[76] A. Behfarnia and A. Eslami, "Risk assessment of autonomous vehicles using bayesian defense graphs," in *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, pp. 1–5, IEEE, 2018.

[77] S. Tariq, S. Lee, and S. S. Woo, "Cantransfer: transfer learning based intrusion detection on a controller area network using convolutional lstm network," in *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, SAC '20, (New York, NY, USA), p. 1048–1055, Association for Computing Machinery, 2020.

[78] H. Lee, S. H. Jeong, and H. K. Kim, "Otids: A novel intrusion detection system for in-vehicle network by using remote frame," in *2017 15th Annual Conference on Privacy, Security and Trust (PST)*, pp. 57–5709, 2017.

[79] U. E. Larson, D. K. Nilsson, and E. Jonsson, "An approach to specification-based attack detection for in-vehicle networks," in *2008 IEEE Intelligent Vehicles Symposium*, pp. 220–225, IEEE, 2008.

[80] S. Fang, G. Zhang, Y. Li, and J. Li, "Windowed hamming distance-based intrusion detection for the can bus," *Applied Sciences*, vol. 14, no. 7, 2024.

[81] Y. Hamada, M. Inoue, H. Ueda, Y. Miyashita, and Y. Hata, "Anomaly-based intrusion detection using the density estimation of reception cycle periods for

in-vehicle networks," *SAE International Journal of Transportation Cybersecurity and Privacy*, vol. 1, no. 11-01-01-0003, pp. 39–56, 2018.

[82] M. Marchetti, D. Stabili, A. Guido, and M. Colajanni, "Evaluation of anomaly detection for in-vehicle networks through information-theoretic algorithms," in *2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*, pp. 1–6, 2016.

[83] D. Stabili, M. Marchetti, and M. Colajanni, "Detecting attacks to internal vehicle networks through hamming distance," in *2017 AEIT International Annual Conference*, pp. 1–6, 2017.

[84] M. Bozdal, M. Samie, and I. K. Jennions, "Winds: A wavelet-based intrusion detection system for controller area network (can)," *IEEE Access*, vol. 9, pp. 58621–58633, 2021.

[85] A. Tomlinson, J. Bryans, S. A. Shaikh, and H. K. Kalutarage, "Detection of automotive can cyber-attacks by identifying packet timing anomalies in time windows," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pp. 231–238, 2018.

[86] R. U. D. Refat, A. A. Elkhail, A. Hafeez, and H. Malik, "Detecting can bus intrusion by applying machine learning method to graph based features," in *Intelligent Systems and Applications* (K. Arai, ed.), (Cham), pp. 730–748, Springer International Publishing, 2022.

[87] D. Swessi and H. Idoudi, "Comparative study of ensemble learning techniques for fuzzy attack detection in in-vehicle networks," in *Advanced Information Networking and Applications* (L. Barolli, F. Hussain, and T. Enokido, eds.), (Cham), pp. 598–610, Springer International Publishing, 2022.

[88] A. R. Javed, S. u. Rehman, M. U. Khan, M. Alazab, and T. R. G, "Canintelliids: Detecting in-vehicle intrusion attacks on a controller area network using cnn and attention-based gru," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 1456–1466, 2021.

[89] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[90] N. Leslie, "An unsupervised learning approach for in-vehicle network intrusion detection," in *2021 55th Annual Conference on Information Sciences and Systems (CISS)*, pp. 1–4, 2021.

[91] V. S. Barletta, D. Caivano, A. Nannavecchia, and M. Scalera, "Intrusion detection for in-vehicle communication networks: An unsupervised kohonen som approach," *Future Internet*, vol. 12, no. 7, 2020.

[92] Y. Wu, A. Q. Jiang, W. Li, M. Rabe, C. Staats, M. Jamnik, and C. Szegedy, "Autoformalization with large language models," *Advances in neural information processing systems*, vol. 35, pp. 32353–32368, 2022.

[93] A. Q. Jiang, S. Welleck, J. P. Zhou, T. Lacroix, J. Liu, W. Li, M. Jamnik, G. Lample, and Y. Wu, "Draft, sketch, and prove: Guiding formal theorem provers with informal proofs," in *The Eleventh International Conference on Learning Representations*, 2022.

[94] T. H. Trinh, Y. Wu, Q. V. Le, H. He, and T. Luong, "Solving olympiad geometry without human demonstrations," *Nature*, vol. 625, no. 7995, pp. 476–482, 2024.

[95] B. Roziere, M.-A. Lachaux, L. Chanussot, and G. Lample, "Unsupervised translation of programming languages," *Advances in neural information processing systems*, vol. 33, pp. 20601–20611, 2020.

[96] M. Jiao, T. Yu, X. Li, G. Qiu, X. Gu, and B. Shen, "On the evaluation of neural code translation: Taxonomy and benchmark," in *Proceedings of the 38th IEEE/ACM International Conference on Automated Software Engineering*, ASE '23, p. 1529–1541, IEEE Press, 2024.

[97] J.-K. Zinzindohoué, K. Bhargavan, J. Protzenko, and B. Beurdouche, "Hacl*: A verified modern cryptographic library," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1789–1806, 2017.

[98] A. Erbsen, J. Philipoom, J. Gross, R. Sloan, and A. Chlipala, "Simple high-level code for cryptographic arithmetic: With proofs, without compromises," *ACM SIGOPS Operating Systems Review*, vol. 54, no. 1, pp. 23–30, 2020.

[99] R. Chapman, A. Petcher, T. Hansen, Y. Peng, T. Lepoint, C. Bytheway, and P. Kampanakis, "Formal verification of cryptographic software at aws-current practices and future trends," 2024.

[100] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24824–24837, 2022.

[101] S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, and K. Narasimhan, "Tree of thoughts: Deliberate problem solving with large language models," *Advances in neural information processing systems*, vol. 36, pp. 11809–11822, 2023.

[102] M. El-Hajj, H. Mousawi, and A. Fadlallah, "Analysis of lightweight cryptographic algorithms on iot hardware platform," *Future Internet*, vol. 15, no. 2, p. 54, 2023.

[103] N. Gookyi, D. Agyemanh, G. Kanda, and K. Ryoo, "Nist lightweight cryptography standardization process: Classification of second round candidates, open challenges, and recommendations.," *Journal of Information Processing Systems*, vol. 17, no. 2, 2021.

[104] J. B. Almeida, M. Barbosa, G. Barthe, F. Dupressoir, and M. Emmi, "Verifying {Constant-Time} implementations," in *25th USENIX Security Symposium (USENIX Security 16)*, pp. 53–70, 2016.

[105] K. Bhargavan, B. Bond, A. Delignat-Lavaud, C. Fournet, C. Hawblitzel, C. Hritcu, S. Ishtiaq, M. Kohlweiss, R. Leino, J. R. Lorch, *et al.*, "Everest: Towards a verified, drop-in replacement of https," in *2nd Summit on Advances in Programming Languages*, 2017.

[106] J. B. Almeida, M. Barbosa, G. Barthe, B. Grégoire, A. Koutsos, V. Laporte, T. Oliveira, and P.-Y. Strub, "The last mile: High-assurance and high-speed cryptographic implementations," in *2020 IEEE Symposium on Security and Privacy (SP)*, pp. 965–982, IEEE, 2020.

[107] A. W. Appel, "Verification of a cryptographic primitive: Sha-256," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 37, no. 2, pp. 1–31, 2015.

[108] A. Lochbihler, S. R. Sefidgar, D. Basin, and U. Maurer, "Formalizing constructive cryptography using crypthol," in *2019 IEEE 32nd Computer Security Foundations Symposium (CSF)*, pp. 152–15214, IEEE, 2019.

[109] M. C. Tschantz, D. Kaynar, and A. Datta, "Formal verification of differential privacy for interactive systems," *Electronic Notes in Theoretical Computer Science*, vol. 276, pp. 61–79, 2011.

[110] G. Barthe, F. Dupressoir, B. Grégoire, C. Kunz, B. Schmidt, and P.-Y. Strub, "Easycrypt: A tutorial," *International School on Foundations of Security Analysis and Design*, pp. 146–166, 2012.

[111] A. Gohr, "Improving attacks on round-reduced speck32/64 using deep learning," in *Annual International Cryptology Conference*, pp. 150–179, Springer, 2019.

[112] A. Benamira, D. Gerault, T. Peyrin, and Q. Q. Tan, "A deeper look at machine learning-based cryptanalysis," in *Annual international conference on the theory and applications of cryptographic techniques*, pp. 805–835, Springer, 2021.

[113] F. Hameed and H. Alkhzaimi, "Deep learning-based profiling side-channel attacks in speck cipher," *Scientific Reports*, vol. 15, no. 1, p. 26149, 2025.

[114] T. R. Lee, J. S. Teh, N. Jamil, J. L. S. Yan, and J. Chen, "Lightweight block cipher security evaluation based on machine learning classifiers and active s-boxes," *IEEE Access*, vol. 9, pp. 134052–134064, 2021.

[115] U. Alon, M. Zilberstein, O. Levy, and E. Yahav, "code2vec: Learning distributed representations of code," *Proceedings of the ACM on Programming Languages*, vol. 3, no. POPL, pp. 1–29, 2019.

[116] Y. Wu, J. Lu, Y. Zhang, and S. Jin, "Vulnerability detection in c/c++ source code with graph representation learning," in *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 1519–1524, IEEE, 2021.

[117] A. Paliwal, S. Loos, M. Rabe, K. Bansal, and C. Szegedy, "Graph representations for higher-order logic and theorem proving," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 2967–2974, 2020.

[118] B. Asvija, R. Eswari, and M. Bijoy, "Bayesian attack graphs for platform virtualized infrastructures in clouds," *Journal of Information Security and Applications*, vol. 51, p. 102455, 2020.

[119] A. Kazeminajafabadi and M. Imani, "Optimal monitoring and attack detection of networks modeled by bayesian attack graphs," *Cybersecurity*, vol. 6, no. 1, p. 22, 2023.

[120] R. S. Ross, "Guide for conducting risk assessments," 2012.

[121] S. H. Houmb and V. N. Franqueira, "Estimating toe risk level using cvss," in *2009 International Conference on Availability, Reliability and Security*, pp. 718–725, IEEE, 2009.

[122] N. S. Yanofsky and M. A. Mannucci, *Quantum computing for computer scientists*. Cambridge University Press, 2008.

[123] V. C. W. Younang, ""iot-risk-assessment-with-complex-probabilities"," 2025. Github link containing the documents, code and link source used for our analysis.

[124] P. Oser, R. W. van der Heijden, S. Lüders, and F. Kargl, "Risk prediction of iot devices based on vulnerability analysis," *ACM Transactions on Privacy and Security*, vol. 25, no. 2, pp. 1–36, 2022.

[125] F. Arat and S. Akleylek, "A new method for vulnerability and risk assessment of iot," *Computer Networks*, vol. 237, p. 110046, 2023.

[126] M. D. Hossain, H. Inoue, H. Ochiai, D. Fall, and Y. Kadobayashi, "Lstm-based intrusion detection system for in-vehicle can bus communications," *IEEE Access*, vol. 8, pp. 185489–185502, 2020.

[127] S. T. Mehedi, A. Anwar, Z. Rahman, and K. Ahmed, "Deep transfer learning based intrusion detection system for electric vehicular networks," *Sensors (Basel, Switzerland)*, vol. 21, 2021.

[128] I. Nazakat and K. Khurshid, "Intrusion detection system for in-vehicular communication," in *2019 15th International Conference on Emerging Technologies (ICET)*, pp. 1–6, 2019.

[129] H.-C. Lin, P. Wang, K.-M. Chao, W.-H. Lin, and J.-H. Chen, "Using deep learning networks to identify cyber attacks on intrusion detection for in-vehicle networks," *Electronics*, vol. 11, p. 2180, 07 2022.

[130] M. Hanselmann, T. Strauss, K. Dormann, and H. Ulmer, "Canet: An unsupervised intrusion detection system for high dimensional can bus data," *Ieee Access*, vol. 8, pp. 58194–58205, 2020.

[131] A. K. Desta, S. Ohira, I. Arai, and K. Fujikawa, "Rec-cnn: In-vehicle networks intrusion detection using convolutional neural networks trained on recurrence plots," *Vehicular Communications*, vol. 35, p. 100470, 2022.

[132] A. Alshammari, M. A. Zohdy, D. Debnath, and G. Corser, "Classification approach for intrusion detection in vehicle systems," *Wireless Engineering and Technology*, vol. 9, no. 4, pp. 79–94, 2018.

[133] B. Lampe and W. Meng, "can-train-and-test: A curated can dataset for automotive intrusion detection," *Computers & Security*, vol. 140, p. 103777, 2024.

[134] M. L. Han, B. I. Kwak, and H. K. Kim, "Anomaly intrusion detection method for vehicular networks based on survival analysis," *Vehicular Communications*, vol. 14, pp. 52–63, 2018.

[135] H. Kang, B. I. Kwak, Y. H. Lee, H. Lee, H. Lee, and H. K. Kim, "Car hacking: Attack defense challenge 2020 dataset," *(No Title)*, 2021.

[136] Unknown, "Hacking dataset." Last accessed: 03.20.2025.

[137] H. M. Song, J. Woo, and H. K. Kim, "In-vehicle network intrusion detection using deep convolutional neural network," *Vehicular Communications*, vol. 21, p. 100198, 2020.

[138] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *Journal of Machine Learning Research*, vol. 18, no. 185, pp. 1–52, 2018.

[139] M. Hanselmann, T. Strauss, K. Dormann, and H. Ulmer, "Syncan: Synthetic can bus dataset generator," 2020.

[140] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[141] A. Taylor, S. Leblanc, and N. Japkowicz, "Anomaly detection in automobile control network data with long short-term memory networks," in *2016 IEEE international conference on data science and advanced analytics (DSAA)*, pp. 130–139, IEEE, 2016.

[142] M. Weber, G. Wolf, E. Sax, and B. Zimmer, "Online detection of anomalies in vehicle signals using replicator neural networks," *Proc. 6th Escar USA*, pp. 1–14, 2018.

[143] B. Lampe and W. Meng, "Can benchmark dataset: A curated can dataset for automotive intrusion detection." `https://bitbucket.org/brooke-lampe/can-benchmark/src/master/`, 2024.

[144] N. Ninja, "Scaling and normalization: Standardizing numerical data." Last accessed: 04.22.2025.

[145] Nvidia, "Autonomous vehicles." Last accessed: 03.24.2025.

[146] Wikipedia, "Tesla autopilot hardware." Last accessed: 03.24.2025.

[147] Samsung, "An ideal solution for the next-generation in-vehicle experience." Last accessed: 03.24.2025.

[148] Variscite, "Var-som-mx8: Nxp imx8." Last accessed: 03.24.2025.

[149] C. Electronics, "Can bus explained - a simple introduction," 2025.

[150] "Road vehicles — controller area network (can) — part 1: Data link layer and physical signalling," 2015.

[151] Kvaser, "Can frame types - user guide," Updated 2025.

[152] C. T. Wilfried Voss, "Controller area network (can) bus tutorial - message frame format," 2021.

[153] O. Oyeniyi and V. W. Younang, "Model deployment feasibility metrics." `github.com/oluwafeyisayo/Model-Deployment-Feasibility-Metrics.git`, 2025.

[154] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, "The simon and speck lightweight block ciphers," in *Proceedings of the 52nd annual design automation conference*, pp. 1–6, 2015.

[155] G. Yang, B. Zhu, V. Suder, M. D. Aagaard, and G. Gong, "The simeck family of lightweight block ciphers," in *International workshop on cryptographic hardware and embedded systems*, pp. 307–329, Springer, 2015.

[156] R. M. Needham and D. J. Wheeler, "Tea extensions," *Report (Cambridge University, Cambridge, UK, 1997)*, 1997.

[157] D. Dinu, L. Perrin, A. Udovenko, V. Velichkov, J. Großschädl, and A. Biryukov, "Sparx: a family of arx-based lightweight block ciphers provably secure against linear and differential attacks," in *NIST Lightweight Cryptography Workshop 2016*, 2016.

[158] D. Hong, J.-K. Lee, D.-C. Kim, D. Kwon, K. H. Ryu, and D.-G. Lee, "Lea: A 128-bit block cipher for fast encryption on common processors," in *international workshop on information security applications*, pp. 3–27, Springer, 2013.

[159] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. Robshaw, Y. Seurin, and C. Vikkelsoe, "Present: An ultra-lightweight block cipher," in *International workshop on cryptographic hardware and embedded systems*, pp. 450–466, Springer, 2007.

[160] S. Banik, S. K. Pandey, T. Peyrin, Y. Sasaki, S. M. Sim, and Y. Todo, "Gift: A small present: Towards reaching the limit of lightweight encryption," in *International conference on cryptographic hardware and embedded systems*, pp. 321–345, Springer, 2017.

[161] W. Zhang, Z. Bao, D. Lin, V. Rijmen, B. Yang, and I. Verbauwhede, "Rectangle: a bit-slice lightweight block cipher suitable for multiple platforms," *Science China Information Sciences*, vol. 58, no. 12, pp. 1–15, 2015.

[162] C. McCoy, "Simon_Speck_Ciphers." `https://github.com/inmcm/Simon_Speck_Ciphers`, 2018. GitHub repository, Accessed: 2025-08-27.

[163] C. McCoy, "present_cipher." `https://github.com/inmcm/present_cipher`, 2016. GitHub repository, Accessed: 2025-09-11.

[164] A. Ranea, Y. Liu, and T. Ashur, "An easy-to-use tool for rotational-xor cryptanalysis of arx block ciphers," *Proceedings of the Romanian Academy, Series A*, vol. 18, no. 4, pp. 307–314, 2017.

[165] G. C. Team, "GIFT: A small present - GitHub repository." https://github.com/giftcipher/gift, 2017. Accessed: 2026-01-26.

[166] M. Walid, "Rectangle-lightweight-block-cipher: Implementation of the RECTANGLE block cipher." https://github.com/m-walid/Rectangle-lightweight-block-cipher, Dec. 2021.

[167] D. Guo, Q. Zhu, D. Yang, Z. Xie, K. Dong, W. Zhang, G. Chen, X. Bi, Y. Wu, Y. Li, *et al.*, "Deepseek-coder: When the large language model meets programming–the rise of code intelligence," *arXiv preprint arXiv:2401.14196*, 2024.

[168] A. Lozhkov, R. Li, L. B. Allal, F. Cassano, J. Lamy-Poirier, N. Tazi, A. Tang, D. Pykhtar, J. Liu, Y. Wei, *et al.*, "Starcoder 2 and the stack v2: The next generation," *arXiv preprint arXiv:2402.19173*, 2024.

[169] B. Hui, J. Yang, Z. Cui, J. Yang, D. Liu, L. Zhang, T. Liu, J. Zhang, B. Yu, K. Lu, *et al.*, "Qwen2. 5-coder technical report," *arXiv preprint arXiv:2409.12186*, 2024.

[170] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[171] Anonymous, "TIA framework for cross-paradigm logic translation Repository." https://anonymous.4open.science/r/TIA-framework-for-cross-paradigm-logic-translation-402E/, 2026.

[172] Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, *et al.*, "Lora: Low-rank adaptation of large language models.,"

[173] D. Hong, J. Sung, S. Hong, J. Lim, S. Lee, B.-S. Koo, C. Lee, D. Chang, J. Lee, K. Jeong, *et al.*, "Hight: A new block cipher suitable for low-resource device," in *International workshop on cryptographic hardware and embedded systems*, pp. 46–59, Springer, 2006.

[174] C. Beierle, J. Jean, S. Kölbl, G. Leander, A. Moradi, T. Peyrin, Y. Sasaki, P. Sasdrich, and S. M. Sim, "The skinny family of block ciphers and its low-latency variant mantis," in *Annual international cryptology conference*, pp. 123–153, Springer, 2016.

[175] National Institute of Standards and Technology, "Finalists - lightweight cryptography." https://csrc.nist.gov/projects/lightweight-cryptography/finalists.

[176] C. Guo, T. Iwata, M. Khairallah, K. Minematsu, and T. Peyrin, "Romulus as nist lwc finalist," in *NIST Lightw. Cryptogr. Workshop*, 2019.

[177] S. Banik, A. Chakraborti, A. Inoue, T. Iwata, K. Minematsu, M. Nandi, T. Peyrin, Y. Sasaki, S. M. Sim, and Y. Todo, "Gift-cofb," *Cryptology ePrint Archive*, 2020.

[178] Y. Bertot and P. Castéran, *Interactive theorem proving and program development: Coq'Art: the calculus of inductive constructions*. Springer Science & Business Media, 2013.

[179] M. Kaufmann, P. Manolios, and J. S. Moore, *Computer-aided reasoning: ACL2 case studies*, vol. 4. Springer Science & Business Media, 2013.

[180] E. Biham and A. Shamir, "Differential cryptanalysis of des-like cryptosystems," *Journal of CRYPTOLOGY*, vol. 4, no. 1, pp. 3–72, 1991.

[181] A. Biryukov *et al.*, "SPARX reference implementation." `https://github.com/cryptolu/SPARX`, 2016. Official C implementation by the University of Luxembourg research team.

[182] C. McCoy, "skinny_cipher: Implementations of the Skinny block cipher." `https://github.com/inmcm/skinny_cipher`, 2017.

[183] H. Chen and X. Wang, "Improved linear hull attack on round-reduced simon with dynamic key-guessing techniques," in *International Conference on Fast Software Encryption*, pp. 428–449, Springer, 2016.

[184] L. Qin, H. Chen, and X. Wang, "Linear hull attack on round-reduced simeck with dynamic key-guessing techniques," in *Australasian Conference on Information Security and Privacy*, pp. 409–424, Springer, 2016.

[185] E. Barker and A. Roginsky, "Transitioning the use of cryptographic algorithms and key lengths," tech. rep., National Institute of Standards and Technology, 2018.

[186] A. Kerckhoffs, *La cryptographie militaire, ou, Des chiffres usités en temps de guerre: avec un nouveau procédé de déchiffrement applicable aux systèmes à double clef*. Librairie militaire de L. Baudoin, 1883.

[187] E. B. Barker, W. C. Barker, W. E. Burr, W. T. Polk, and M. E. Smid, "Sp 800-57. recommendation for key management, part 1: General (revised)," 2007.

[188] Z. Chu, H. Chen, X. Wang, X. Dong, and L. Li, "Improved integral attacks on simon32 and simon48 with dynamic key-guessing techniques," *Security and Communication Networks*, vol. 2018, no. 1, p. 5160237, 2018.

[189] J. Lu, "Cryptanalysis of reduced versions of the hight block cipher from ches 2006," in *International Conference on Information Security and Cryptology*, pp. 11–26, Springer, 2007.