## RESEARCH ARTICLE

# Revolutionizing Cyber Threat Detection With Large Language Models: A Privacy-Preserving BERT-Based Lightweight Model for IoT/IIoT Devices

**MOHAMED AMINE FERRAG**[1], **(Senior Member, IEEE), MTHANDAZO NDHLOVU**[1],
**NORBERT TIHANYI**[1], **(Member, IEEE), LUCAS C. CORDEIRO**[2,3],
**MEROUANE DEBBAH**[4], **(Fellow, IEEE), THIERRY LESTABLE**[1],
**AND NARINDERJIT SINGH THANDI**[1]

[1]Technology Innovation Institute, Abu Dhabi, United Arab Emirates
[2]Department of Computer Science, The University of Manchester, M13 9PL Manchester, U.K.
[3]Department of Electronics and Computing, Federal University of Amazonas, Manaus 69067-005, Brazil
[4]KU 6G Research Center, Khalifa University of Science and Technology, Abu Dhabi, United Arab Emirates

Corresponding author: Mohamed Amine Ferrag (Mohamed.Ferrag@tii.ae)

**ABSTRACT** The field of Natural Language Processing (NLP) is currently undergoing a revolutionary transformation driven by the power of pre-trained Large Language Models (LLMs) based on groundbreaking Transformer architectures. As the frequency and diversity of cybersecurity attacks continue to rise, the importance of incident detection has significantly increased. IoT devices are expanding rapidly, resulting in a growing need for efficient techniques to autonomously identify network-based attacks in IoT networks with both high precision and minimal computational requirements. This paper presents SecurityBERT, a novel architecture that leverages the Bidirectional Encoder Representations from Transformers (BERT) model for cyber threat detection in IoT networks. During the training of SecurityBERT, we incorporated a novel privacy-preserving encoding technique called Privacy-Preserving Fixed-Length Encoding (PPFLE). We effectively represented network traffic data in a structured format by combining PPFLE with the Byte-level Byte-Pair Encoder (BBPE) Tokenizer. Our research demonstrates that SecurityBERT outperforms traditional Machine Learning (ML) and Deep Learning (DL) methods, such as Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs), in cyber threat detection. Employing the Edge-IIoTset cybersecurity dataset, our experimental analysis shows that SecurityBERT achieved an impressive 98.2% overall accuracy in identifying fourteen distinct attack types, surpassing previous records set by hybrid solutions such as GAN-Transformer-based architectures and CNN-LSTM models. With an inference time of less than 0.15 seconds on an average CPU and a compact model size of just 16.7MB, SecurityBERT is ideally suited for real-life traffic analysis and a suitable choice for deployment on resource-constrained IoT devices.

**INDEX TERMS** Cyber threat detection, IoT networks, generative AI, BERT, large language models.

## I. INTRODUCTION

According to a Statista report [1], it is projected that the global number of Internet of Things (IoT) connected devices could

The associate editor coordinating the review of this manuscript and approving it for publication was Genoveffa Tortora.

potentially reach 30 billion by the year 2030. With the rise in the number of IoT devices, there is also a growing incidence of cyber threats, posing substantial challenges to the security of diverse systems and networks [2]. As adversaries consistently evolve their tactics, the need for advanced and effective detection mechanisms becomes paramount. Manual

detection methods and conventional approaches are becoming outdated, and various Machine Learning (ML) techniques have emerged, combating these new threats more effectively. In this context, Natural Language Processing (NLP) techniques are gaining attention as a promising approach for cyber threat detection [3]. Among these techniques, the Bidirectional Encoder Representations from Transformers (BERT) model [4], a pre-trained transformer-based language model, has achieved remarkable success in several NLP applications. By exploiting BERT's contextual understanding, security researchers have found unique techniques to handle diverse cybersecurity concerns [5]. Researchers have recently been exploring using BERT and pre-trained language models in a wide range of cybersecurity applications, including malware detection in Android applications, identification of spam emails, intrusion detection in automotive systems, and anomaly detection in system logs [6], [7], [8]. Network-based traffic, such as port scans and packet floods, primarily consists of numerical data rather than textual information. This characteristic poses a challenge when attempting to leverage models like BERT to understand the semantic relationships between various types of network packets. While employing complex Large Language Models (LLMs) with billions of parameters can improve threat detection accuracy, their extensive computational needs render them impractical for implementation in embedded devices.

We present `SecurityBERT`, a novel lightweight privacy-preserving architecture for cyber threat detection in IoT networks. By employing a dedicated encoding technique designed for this specific purpose, we surpassed the performance of all existing ML algorithms and models in cyber threat detection. During the design of `SecurityBERT`, we had three primary goals in mind:

- To create an exceptionally compact model capable of executing rapid inferences without causing noticeable delays. This design choice enables real-time traffic analysis and facilitates embedding the model in IoT devices;
- To maintain the confidentiality of the extracted network data, ensuring that classification can be performed on untrusted servers;
- To surpass the accuracy levels of previous ML models in this field.

Achieving superior accuracy compared to existing hybrid solutions has proven a significant challenge in our architectural design. Striking the right balance is crucial. If the architecture becomes overly complex, it may become impractical for real-life traffic analysis. Conversely, if the model is overly simplistic, it may not provide the necessary accuracy for effective multi-classification, thus hindering its overall performance. Our original contributions are as follows:

- Our research introduces a novel privacy-preserving encoding approach called Privacy-Preserving Fixed-Length Encoding (PPFLE). By combining PPFLE with the Byte-level Byte-Pair Encoder (BBPE) tokenizer, we can effectively represent network traffic data in

a structured manner. By implementing this technique, we have achieved significant performance improvements compared to using text data with varying sizes;
- We have designed a 15-layer BERT-based architecture with only 11 million parameters for multi-category classification. We trained the model on PPFLE encoded data, which we refer to as `SecurityBERT`;
- We evaluated the efficiency of our proposed approach using the Edge-IIoTset cyber security dataset [9]. Various ML techniques have recently been tested on this dataset, providing a solid foundation for fair comparison. According to our experimental analysis, our method effectively identifies fourteen distinct types of attacks on an average CPU in less than 0.3 seconds, achieving an overall accuracy of 98.2%. To the best of our knowledge, this achievement showcases the highest accuracy ever attained among all ML algorithms, outperforming both the Convolutional Neural Network (CNN) and Transformer models.

This paper is organized as follows: Section II presents an exploration of the related work. Subsequently, Section III outlines the significant steps in developing `SecurityBERT`. In Section IV, we evaluate the performance of the proposed model. Lastly, we conclude our research and provide insight into potential future research directions of interest in Section V.

## II. RELATED WORK

As various researchers have already demonstrated, the BERT model proves to be an exceptional starting point for identifying cybersecurity threats. BERT has been utilized in various fields, from detecting log anomalies to identifying malicious web requests.

A noteworthy study by Alkhatib et al. [10] demonstrated the feasibility of using BERT for learning the sequence of arbitration identifiers (IDs) in a Controller Area Network (CAN) via a "masked language model" unsupervised training objective. They proposed the CAN-BERT transformer model for anomaly detection in current automotive systems and showed that the BERT model outperforms its predecessors regarding accuracy and F1-score. Rahali and Akhloufi [6] introduced MalBERT, a tool that conducts static analysis on the source code of Android applications. They used BERT to comprehend the contextual relationships of code words and classify them into representative malware categories. Their results further underscored the high performance of transformer-based models in malicious software detection.

Chen and Liao [8] introduced BERT-Log, an anomaly detection and fault diagnosis approach in large-scale computer systems that treat system logs as natural language sequences. They leveraged a pre-trained BERT model to learn the semantic representation of normal and anomalous logs, fine-tuning the model with a fully connected neural network to detect abnormalities. Seyyar et al. [7] proposed a model

for detecting anomalous HTTP requests in web applications, employing Deep Learning (DL) techniques and BERT. Aghaei et al. [11] presented SecureBERT,[1] a language model tailored explicitly for cybersecurity tasks, focusing on Cyber Threat Intelligence (CTI) and automation. The SecureBERT model offers a practical way of transforming natural language CTI into machine-readable formats, thereby minimizing the necessity for labor-intensive manual analysis. The authors devised a unique tokenizer and a method for adjusting pre-trained weights to ensure that SecureBERT understands general English and cybersecurity-related text. However, SecureBERT is not designed to process network-based cyber threat attacks.

CyBERT, introduced by Ranade et al. [12], is a custom version of BERT designed specifically for cybersecurity applications. This model has been fine-tuned using a vast corpus of cybersecurity data to enhance its ability to process intricate information concerning threats, attacks, and vulnerabilities. Yu et al. [13] explored a deep-learning-based approach for detecting advanced persistent threats (APTs) in the Industrial Internet of Things (IIoT), using the BERT model to address the challenges of long attack sequences. Their experimental results demonstrate the method's effectiveness, yielding high accuracy and a low false alarm rate in APT detection. Breve et al. [14] proposed using NLP techniques, specifically a BERT-based model, to detect potentially harmful automation rules in trigger-action IoT platforms that could breach user security or privacy. Their evaluation on the If-This-Then-That platform with over 76,000 rules demonstrated high accuracy, significantly outperforming traditional information flow analysis methods. Recently Wang et al. [15] developed BERT-of-Theseus, Vision Transformer, and PoolFormer (BT-TPF), an IoT intrusion detection model tailored for resource-limited IoT environments, using a knowledge-distillation approach. The model employs a Siamese network for feature reduction and a Vision Transformer to train a compact Poolformer model, achieving a significant parameter reduction while maintaining high accuracy. The aforementioned studies leverage pre-trained BERT models and customize them to meet their unique security needs by fine-tuning or using them as feature generators. These models benefit from the textual form and sequential nature of their security-related data, including sources such as code, emails, and log sequences. These studies effectively utilize BERT's ability to comprehend contextual relationships within sequences to carry out precise detection and classification tasks.

In cyber threat detection, it is vital to compare different research efforts. In real-world cyber threat detection scenarios, support is crucial for extracting features from network traffic, often relying on PCAP files. In addition to analyzing real packet data and detecting cyber threats on networks, it is important to consider privacy in training data, especially since

IoT devices and network data may contain sensitive information. Given the uniqueness of each network infrastructure and the need for high accuracy through fine-tuning or new training, sharing actual network traffic data for training purposes can raise privacy concerns. `SecurityBERT` has been developed as a pioneering, lightweight, privacy-preserving architecture specifically designed with this consideration in mind. TABLE 1 provides a comparison of various recent works on cyber threat detection in terms of four key parameters:

- **D = Detect:** Network-based Cyber Threat Detection
- **L = LLM:** Utilization of LLMs
- **N = Network PCAP:** Packet data analysis of a traffic
- **P = Privacy:** Privacy-preserving training data

**TABLE 1.** Comparison with recent works on cyber threat detection.

| Frameworks | Year | D | L | N | P |
|---|---|---|---|---|---|
| Alkhatib *et al.* [10] | 2022 | ✗ | ✓ | ✗ | ✗ |
| Rahali *et al.* [6] | 2022 | ✗ | ✓ | ✗ | ✗ |
| Aghaei *et al.* [11] | 2022 | ✗ | ✓ | ✗ | ✗ |
| Thapa *et al.* [4] | 2022 | ✗ | ✓ | ✗ | ✗ |
| Hamouda *et al.* [16] | 2022 | ✓ | ✗ | ✓ | ✗ |
| Friha *et al.* [17] | 2022 | ✓ | ✗ | ✓ | ✗ |
| Chen *et al.* [8] | 2022 | ✗ | ✓ | ✗ | ✗ |
| Seyyar *et al.* [7] | 2022 | ✗ | ✓ | ✗ | ✗ |
| Selvaraja *et al.* [18] | 2023 | ✓ | ✗ | ✓ | ✗ |
| B. Breve *et al.* [14] | 2023 | ✗ | ✓ | ✗ | ✗ |
| Chen *et al.* [19] | 2023 | ✗ | ✓ | ✗ | ✗ |
| Douiba *et al.* [20] | 2023 | ✓ | ✗ | ✓ | ✗ |
| Jahangir *et al.* [21] | 2023 | ✓ | ✗ | ✓ | ✗ |
| Hu *et al.* [22] | 2023 | ✓ | ✗ | ✓ | ✗ |
| K. Yu *et al.* [13] | 2023 | ✓ | ✓ | ✗ | ✗ |
| Hu *et al.* [22] | 2023 | ✓ | ✗ | ✓ | ✗ |
| Friha *et al.* [23] | 2023 | ✓ | ✗ | ✓ | ✗ |
| Chakraborty *et al.* [24] | 2023 | ✓ | ✗ | ✓ | ✗ |
| Wang *et al.* [25] | 2023 | ✓ | ✗ | ✓ | ✗ |
| Liu *et al.* [26] | 2023 | ✓ | ✗ | ✓ | ✗ |
| Aouedi *et al.* [27] | 2023 | ✓ | ✗ | ✓ | ✗ |
| Z. Wang *et al.* [15] | 2023 | ✓ | ✓ | ✓ | ✗ |
| **SecurityBERT** | 2023 | ✓ | ✓ | ✓ | ✓ |

✓: Supported, ✗: Not Supported.

The majority of the research conducted in 2022, including [6], [10], and [11], integrated LLMs, but they did not support detection, nor did they utilize packet data. However, contrary to the norm, Hamouda et al. [16] and Friha et al. [17] demonstrated support for cyber threat detection and utilized packet data but did not rely on the capabilities of LLMs. Works from 2023, such as [18], [20], and [25], emphasize more cyber threat detection and the use of packet data, but they largely lack in the application of LLMs.

## III. SecurityBERT ARCHITECTURE DESIGN

FIGURE 1 visually presents the comprehensive workflow of the model, encompassing all relevant steps from dataset preparation to classification. Each of these steps will be extensively covered in this section. Developing a BERT model from the ground up for network-based cyber threat detection demands a thorough and intricate approach. Below

---

[1] While the names may sound similar, it is important to note that SecureBERT is separate from our recently introduced SecurityBERT.
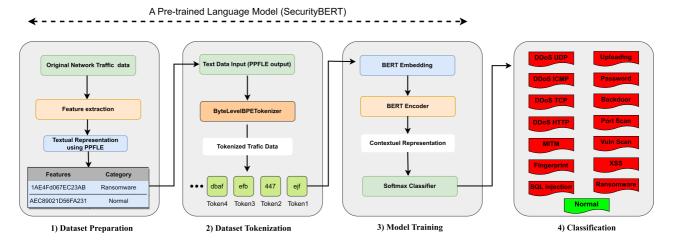
**FIGURE 1.** High-level workflow of our SecurityBERT model.

is a comprehensive outline detailing the main steps in the process:

---

**STEPS 1** Main steps of building SecurityBERT

---

1: **Dataset Utilization**
2: **Feature Extraction**
3: **Privacy-Preserving Fixed-Length Encoding (PPFLE)**
4: **Byte-level BPE (BBPE) Tokenizer**
5: **SecurityBERT Embedding**
6: **Contextual Representation**
7: **Training SecurityBERT**
   - Text Normalization
   - Text Tokenization
   - Frequency Filtering
   - Vocabulary Creation
   - Special Token Addition
   - Tokenizer Training
8: **Fine-tuning with Softmax activation function**

---

### A. DATASET UTILIZATION (EDGE-IIoTSET DATASET)

Generating our dataset through real-life traffic analysis would be time-consuming, and there's the risk of specific attacks not being adequately simulated or missing from our dataset. Hence, acquiring and utilizing realistic datasets for our research is crucial.

Cybersecurity and network security data can be gathered from various online sources using open-source databases and repositories. Notable examples include the Common Vulnerabilities and Exposures (CVE) database, the Open Web Application Security Project (OWASP), and numerous others for network security [28]. The primary challenge presented by these sources is their heavy reliance on artificial scenarios, which results in a deficiency of authentic data. Training a model exclusively on such data can potentially lead to unrealistic outcomes. Furthermore, most of these databases do not include packet network data, which poses a challenge

in simulating realistic scenarios. Our primary aim is to opt for a dataset that tackles this constraint by strongly emphasizing genuine network data. Furthermore, we intend to ensure maximum diversity within this dataset, encompassing a comprehensive range of attack types, including ransomware, XSS, SQL injection, DoS, and other widely recognized attack categories. This diversified dataset's rationale is to assess our newly proposed model's classification capabilities comprehensively. In 2022, Ferrag et al. introduced Edge-IIoTset [9], a new and extensive cybersecurity dataset specifically designed for IoT and IIoT applications. This dataset serves as a valuable resource for ML-based intrusion detection systems. The Edge-IIoTset dataset includes diverse devices, sensors, protocols, and cloud/edge configurations, rendering it highly representative of real-world scenarios and aligning perfectly with our research objectives. This dataset contains fifteen (15) attacks related to the Internet of Things (IoT) and Industrial IoT (IIoT) connectivity protocols, categorized into five threats: DoS/DDoS attacks, Information gathering, Man-in-the-middle (MITM), Injection attacks, and Malware attacks, which can be seen in Figure 2. The DoS/DDoS attack category encompasses TCP SYN Flood, UDP flood, HTTP flood, and ICMP flood attacks. The Information Gathering category includes attacks like port scanning, operating system fingerprinting, and vulnerability scanning. MITM attacks include tactics such as DNS Spoofing and ARP Spoofing. Injection attacks include Cross-Site Scripting (XSS), SQL injection, and file-uploading attacks. Lastly, the Malware category covers backdoors, password crackers, and ransomware attacks.

### B. FEATURES EXTRACTION

Given a PCAP file with a network traffic log, we extract relevant features from a specific time window and return them in a structured format suitable for analysis. Specifically, we identify and separate each network flow in the PCAP file. For each flow identified, we extract a set of predefined
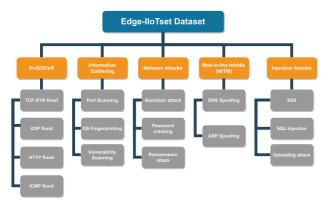
**FIGURE 2.** Categories of the Edge-IIoTset dataset.

features. Then, we organize the extracted features into a CSV file format for analysis. We removed null features from the Edge-IIoTset dataset during our initial exploration, identifying 61 distinct and diverse features. These features are sufficiently various to distinguish the distinctive patterns of network attacks exclusively.

The Edge-IIoTset dataset comprises features gathered from various sources, including network traffic, logs, system resources, and alerts. To better understand these features, the initial 15 can be seen in TABLE 2. For a comprehensive view of all 61 features, please see Table 7 in [9].

**TABLE 2.** The first 15 features gathered from PCAP files.

| N° | Name | Prot. Layer | Type |
|---|---|---|---|
| 1 | frame.time | Frame | Date and time |
| 2 | ip.src_host | IP | Character string |
| 3 | ip.dst_host | IP | Character string |
| 4 | arp.dst.proto_ipv4 | ARP | IPv4 address |
| 5 | arp.opcode | ARP | Unsigned integer |
| 6 | arp.hw.size | ARP | Unsigned integer |
| 7 | arp.src.proto_ipv4 | ARP | IPv4 address |
| 8 | icmp.checksum | ICMP | Unsigned integer |
| 9 | icmp.seq_le | ICMP | Unsigned integer |
| 10 | icmp.transmit_timestamp | ICMP | Unsigned integer |
| 11 | icmp.unused | ICMP | Sequence of bytes |
| 12 | http.file_data | HTTP | Character string |
| 13 | http.content_length | HTTP | Unsigned integer |
| 14 | http.request.uri.query | HTTP | Character string |
| 15 | http.request.method | HTTP | Character string |
| . | ... | ... | ... |
| . | ... | ... | ... |
| 61 | mbtcp.unit_id | Modbus/TCP | Unsigned Integer |

Numerous studies have already shown that these 61 distinct features are sufficient to detect specific network-based cyberattacks. This dataset, therefore, serves as an optimal foundation for comparing various ML algorithms [29]. After discussing the exact architectural design, the evaluation and comparison of `SecurityBERT` with other research will be detailed in Section IV.

### C. PRIVACY-PRESERVING FIXED-LENGTH ENCODING

A pivotal aspect of the design involves representing the unstructured network data in a manner that allows BERT to comprehend the context and relationships between various

features. BERT is designed to understand English proficiently but may not be the most suitable ML model for comprehending relationships between numbers. In our case, many features are numerical values, i.e., unsigned integers, not strings (as illustrated in TABLE 2), making it difficult to discern their interrelationships using natural language processing methods.

To leverage the power of BERT natural language understanding, we process the dataset, comprising numerical and categorical values, and transform it into textual representation. Specifically, we added context to the data by incorporating column names and concatenating them with their respective values. Then, each new value is hashed and combined with other hashed values within the same instance, resulting in the generation of a sequence. By employing this technique, we have developed a new language comprehensible to BERT and introduced privacy into the training data through cryptographic hash functions. We call this novel textual representation technique as *Privacy-Preserving Fixed-Length Encoding (PPFLE)*.

Significant similarities in log files, TCP scans, and memory dumps may lead to misinterpretation and incorrect classification of various attacks. Employing a hash function allows for handling even minor deviations in the data, effectively representing them as distinct data points for ML. Moreover, specific attacks, like UDP scans and others that are challenging to represent as plain text, can be better understood by the model when they are converted into hashed values. Put simply, we have developed a new linguistic format that the BERT model comprehends much more effectively than mere numerical data, and it aligns more closely with the natural English language for which the BERT model is specifically tailored.

Through this method, we fashioned a representation of the numbers that closely align with natural language, allowing the model to attain enhanced classification accuracy, as detailed in Section IV-A. Correctly converting network data and applying PPFLE can achieve higher accuracy than using the original pre-trained BERT model architecture.

#### 1) PPFLE DESCRIPTION

The objectives of PPFLE are twofold. On the one hand, it is designed to convert unstructured network data into a structured format that better mimics the natural English language, aligning well with the BERT model's specialization. On the other hand, it focuses on maintaining privacy by ensuring that only encoded data is observed, thereby hiding sensitive information in the network data while preserving key classification features.

Let us define a matrix denoted by $\mathbf{M}$ with $i$ rows and $j$ columns. Here, $\mathbf{M}[i, j]$ represents the matrix element at the intersection of the ith row and jth column in $\mathbf{M}$. We denote the ith row of $\mathbf{M}$ by $r_i = \mathbf{M}[i, :]$. In $\mathbf{M}$, the first row contains the column names, which serve as labels or identifiers for each column. We denote these column names as $c_j$, where $j$ represents the column index, i.e., $\mathbf{M}[1, j] = c_j$.

Let us define $s(i, j)$ as a concatenation operation where the column name $c_j$, a dollar sign, and the value of the $j$th column in the $(i + 1)$th row $r_i$ are concatenated into a single string, excluding the first row which contains the column names, i.e.,

$$s(i, j) = c_j \; \| \; \text{"\$"} \; \| \; \mathbf{M}[i + 1, j] \tag{1}$$

Next, define $H(x)$ as a hashing operation on a string $x$ and let $L$ is a list where each element is separated by a space, i.e., $L = \{l_1 \, l_2 \, l_3 \, \ldots \, l_k\}$.

Then, the textual representation of each row $i$ in the matrix $\mathbf{M}$ can be expressed as follows:

$$L \leftarrow L \cup \{H(s(i, n))\} \; \forall \, (1 \le n \le j) \tag{2}$$

Repeating this procedure for each row in $\mathbf{M}$, we obtain a new matrix called DataList denoted as $\mathcal{DL}$. In $\mathcal{DL}$, each row represents an $L$ list, i.e.;

$$\mathcal{DL} = \begin{bmatrix} L_1 = [H(s(1, 1)) \, H(s(1, 2)) \ldots H(s(1, j))] \\ L_2 = [H(s(2, 1)) \, H(s(2, 2)) \ldots H(s(2, j))] \\ \vdots \\ L_i = [H(s(i, 1)) \, H(s(i, 2)) \ldots H(s(i, j))] \end{bmatrix}$$
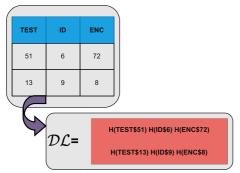


**FIGURE 3.** Creating dataList example.

In other words, the DataList $\mathcal{DL}$ lists where each inner list contains the hashed, concatenated column values for each row in the matrix $\mathbf{M}$. The $\mathbf{M}$ matrix in ML is commonly called a DataFrame. FIGURE 3 showcases a simple DataList creation example. The pseudocode of the PPFLE algorithm can be seen in Algorithm 1.

The PPFLE algorithm, despite its simplicity, effectively translates unstructured data into a fixed-length format. This representation mirrors the characteristics of natural languages, offering considerable advantages when utilized by ML algorithms.

### 2) REMOVED FEATURES FOR PPFLE ENCODING

A natural question arises as to whether all 61 features are suitable for PPFLE encoding. For instance, features like "ip.src_host" and "ip.dst_host" contain IP addresses, which can lead to overfitting, especially if they have unique identifiers or particular details that don't generalize well in different network. Similarly, hashing timestamps with millisecond precision could introduce confusion during training, so it may

---

**Algorithm 1** Privacy-Preserving Fixed-Length Encoding

**Require:** Matrix $\mathbf{M}$ with $i$ rows and $j$ columns
1: **procedure** PPFLE($\mathbf{M}$)
2:     $\mathcal{DL} \leftarrow []$            ▷ Initialize $\mathcal{DL}$ to be empty
3:     **for** $m = 1$ to $i$ **do**        ▷ Iterate through rows in $\mathbf{M}$
4:         $L = []$        ▷ Initialize $L$ to be an empty list
5:         **for** $n = 1$ to $j$ **do**            ▷ Column iteration
6:             $L \leftarrow H(s(m, n))$        ▷ Append $H(x)$ to $L$
7:         **end for**
8:         $\mathcal{DL} \leftarrow L$            ▷ Append $L$ to $\mathcal{DL}$
9:     **end for**
10:     **return** $\mathcal{DL}$
11: **end procedure**

---

be necessary to remove such features if one intends to apply PPFLE encoding. For this reason, several features related to network traffic and packet captures were excluded. High-cardinality features such as "http.request.full_uri" can be challenging to encode effectively and might not offer generalizable patterns. Features with potential redundancy, like the presence of both "ip.src_host" and "arp.src.proto_ipv4", could introduce multicollinearity, affecting model stability. Features such as "frame.time", indicating packet capture timestamps, might not directly relate to the predictive modeling task. Other columns like "tcp.payload" and "http.file_data" represent raw data payloads, which, without extensive preprocessing, could introduce noise rather than clarity. Removing these columns streamlines the dataset, enhancing computational efficiency and ensuring the model focuses on the most relevant and generalizable patterns while maintaining user privacy.

### D. BYTE-LEVEL BPE (BBPE) TOKENIZER

Tokenization is performed on the PPFLE-encoded data. This ensures that no sensitive information is fed to the model during training. A natural question arises: Doesn't the PPFLE compromise the semantics of network data, rendering tokenization unfeasible? By applying PPFLE encoding, we convert numerical values to align with the characteristics of natural language more closely. Each feature is encoded independently, allowing the adjacent hashed values to provide the model with sufficient information about the type of attacks it encounters. Hashing all 61 features together, however, would destroy the semantics of the attacks.

Figure 4 demonstrates the functionality of the PPFLE algorithm, including tokenization.

For instance, using PPFLE to encode a feature for an attack on port 443 with the GET method would appear as: `H(TCP.DSTPORT$443) H(HTTP.METHOD$GET)`. Conversely, a DNS poisoning attack would have a distinct representation, lacking any `HTTP.METHOD` and thus consistently hashed as `H(HTTP.METHOD$0)`. It has turned out during our experimental analysis that these 61 features are highly effective in representing different types of
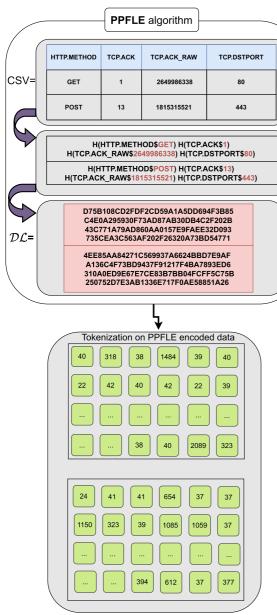
**FIGURE 4.** PPFLE encoding and BBPE tokenization example.

**Algorithm 2** Encode Evaluation Data Sequences

1: $chunk\_size \leftarrow 5000$
2: $num\_chunks \leftarrow \lceil$ len($eval\_data$) / $chunk\_size \rceil$
3: $input\_ids\_eval \leftarrow$ [] ▷ Initialize as an empty list
4: $attention\_masks\_eval \leftarrow$ [] ▷ Initialize as an empty list
5: **for** $i = 0$ to $num\_chunks$ **do**
6:     $start\_idx \leftarrow i \times chunk\_size$
7:     $end\_idx \leftarrow (i + 1) \times chunk\_size$
8:     $chunk \leftarrow eval\_data[start\_idx : end\_idx]$
9:     $encoded\_seqs \leftarrow$ encode($chunk$)
10:     $iic, amc \leftarrow$ UNPACK($encoded\_seqs$)
11:     **append** $icc$ to $input\_ids\_eval$
12:     **append** $amc$ to $attention\_masks\_eval$
13: **end for**
14: **concatenate** the input IDs and attention masks as $input\_ids\_eval, attention\_masks\_eval$

presentation of network traffic data into smaller subwords likely present in the tokenizer's vocabulary as a sequence of bytes, we can efficiently process traffic data by leveraging the power of BERT.

During the training of the tokenizer, a vocabulary size of 5000 was employed, along with a set of specific tokens, including `["<s>", "<pad>", "</s>", "<unk>", "<mask>"]`. The tokenizer's training involved utilizing the file name, setting the vocabulary size, establishing a minimum frequency of 2, and incorporating the list of special tokens. For a visual representation of the various tokens within the PPFLE encoded data, refer to FIGURE 4. Understanding the semantics of these tokens functions similarly to interpreting a typical sentence. The hash output for a specific attack remains constant; thus, if these subword hexadecimal values appear in a particular sequence, BERT can recognize that this unique order corresponds to a hash output and categorize it as a specific attack.

### E. SecurityBERT EMBEDDING

Algorithm 2 showcases the `SecurityBERT` embedding. The algorithm starts by setting the *chunk_size* to 5000.

It then calculates the number of chunks, *num_chunks*, by dividing the length of the *eval_data* by *chunk_size*, and rounding up to the nearest integer. Two empty lists, *input_ids_eval* and *attention_masks_eval*, are initialized to hold the encoded input IDs and attention masks, respectively. The algorithm then enters a loop, iterating from 0 to `num_chunks`. This loop determines the start and end indices for each chunk of the `eval_data`. It retrieves a chunk of data using these indices and encodes each sequence in the chunk, storing the result in `encoded_seqs`. This encoded data is then unpacked into two components: `input_ids_chunk` and `attention_masks_chunk`, denoted by *iic* and *amc*. These components are appended to the `input_ids_eval` and `attention_masks_eval` lists. Once all chunks

network attacks with great accuracy. Furthermore, the model can recognize all attack patterns based on these features, even when hashed. For the data encoded with PPFLE, we employed the `ByteLevelBPETokenizer` from the Hugging Face Transformers library. This tokenizer, initially utilized for GPT-2 [30], breaks down text into subword units for tokenization. It is based on the Byte-Pair Encoding (BPE) algorithm [31], a data compression technique that replaces the most frequent pair of consecutive bytes in a sequence with a single, unused byte. The `ByteLevelBPETokenizer` is particularly useful for handling out-of-vocabulary (OOV) words, which are not present in the tokenizer's vocabulary of human language [32]. By breaking down our language

have been processed, the algorithm concatenates all the input IDs and attention masks in `input_ids_eval` and `attention_masks_eval` respectively, along dimension 0, thereby creating a complete set of input IDs and attention masks for the evaluation data. Here, we note that the `input_ids_eval` and `attention_masks_eval` are important components of the input to transformer-based models. The `input_ids_eval` is a sequence of integers representing the input data after being tokenized. Each integer maps to a token in the model's vocabulary.

The `attention_masks_eval` informs the model about which tokens should be attended to and which should not. In many cases, sequences are padded with special tokens to make all sequences the same length for batch processing. Attention masks prevent the model from attending to these padding tokens. Typically, an attention mask has the same length as the corresponding `input_ids` sequence and contains a 1 for real tokens and a 0 for padding tokens.

### F. CONTEXTUAL REPRESENTATION

We adopted the BERT architecture, which leverages transformers for textual representation and cyber threat classification. Specifically, we pre-trained our `SecurityBERT` using our newly created tokenized dataset. In this process, `SecurityBERT` takes each token from the tokenized text and represents it as an embedding vector, denoted as $X \in R^d$, where $d$ represents the dimensionality of the embedding space. Then `SecurityBERT` utilizes a transformer-based architecture consisting of multiple encoder layers. Each encoder layer comprises multi-head self-attention mechanisms and position-wise feed-forward neural networks. The self-attention mechanism [33] allows the model to capture dependencies and relationships between words within a sentence, thus facilitating contextual understanding. The self-attention mechanism in BERT can be mathematically expressed as follows:

$$\text{Attention(Q, K, V)} = \sigma \left( \frac{QK^T}{\sqrt{d_k}} \right) V, \tag{3}$$

where $\sigma$ is the softmax function, $Q$, $K$, and $V$ are the query, key, and value matrices, respectively, $d_k$ represents the dimensionality of the keys vector, and $T$ denotes the transpose operation. Through self-attention, BERT encodes contextual representations by capturing the importance of different words within a sentence based on their semantic and syntactic relationships. The resulting contextual embeddings are obtained through feed-forward operations and layer normalization.

### G. TRAINING SecurityBERT

The training of `SecurityBERT` involves several crucial steps, each carefully calibrated to ensure optimal performance in security-centric tasks. These steps include data collection and preprocessing, tokenizer training, model configuration, and the training process itself. `SecurityBERT` works with PPFLE-encoded data, simplifying certain steps

in the tokenizer training process and requiring alternative approaches for other steps. Here, we detail the distinct aspects of `SecurityBERT`'s training process.

#### 1) TEXT NORMALIZATION

$$n(D) = \{n(d)|d \in D\} \tag{4}$$

In this function, $n(D)$ represents the normalization process applied to each document $d$ in the set of all documents $D$. Text normalization typically involves converting all text to lowercase, removing punctuation, and sometimes even stemming or lemmatizing words (reducing them to their root form). This process is part of the original BERT architecture; however, when working with PPFLE-encoded data, this element becomes unnecessary and does not provide any extra value to our architecture.

#### 2) TOKENIZATION

$$t(d) = \{t(w)|w \in d, d \in D\} \tag{5}$$

The tokenization function $t(d)$ breaks down each document $d$ in the set $D$ into its constituent words or tokens $w$. These tokens are the basic units of text that a machine-learning model can understand and process.

#### 3) FREQUENCY FILTERING

$$f(D, F) = \{w \in D|\text{freq}(w, D) \geq F\} \tag{6}$$

This function $f(D, F)$ defines a High-Pass filter, cutting out tokens $w$ that have a frequency of occurrence, $\text{freq}(w, D)$ less than the minimum frequency $F$ in the set of all documents $D$. This is to remove rare words that might not provide much informational value for further processing or model training.

#### 4) VOCABULARY CREATION

$$v(D, V) = \{w|w \in D, \text{rank}(w, D) \leq V\} \tag{7}$$

Here, the function $v(D, V)$ creates a vocabulary by choosing the top $V$ words $w$ from the set of all documents $D$ based on their frequency rank $\text{rank}(w, D)$. This forms the vocabulary that the model will recognize.

#### 5) SPECIAL TOKEN ADDITION

$$v' = v \cup S \tag{8}$$

This states that the new vocabulary, $v'$, is a union of the original vocabulary $v$ and the set of special tokens $S$. These special tokens typically include markers for the start and end of sentences, unknown words, padding, etc., and are essential for certain SecurityBERT operations.

**TABLE 3.** Comparison of original BertBase, BertLarge, and SecurityBERT.

| Feature | BertBase | BertLarge | SecurityBERT |
|---|---|---|---|
| Model Type | Pretrained model on English language | Pretrained model on English language | Pretrained model on Network Data |
| Word Embeddings Size | 30522 x 768 | 30522 x 1024 | 30522 x 256 |
| Position Embeddings Size | 512 x 768 | 512 x 1024 | 512 x 256 |
| Token Type Embeddings Size | 2 x 768 | 2 x 1024 | 2 x 256 |
| Number of Layers in Encoder | 12 | 24 | 4 |
| Size of Query, Key, Value in Attention | 768 | 1024 | 256 |
| Size of Intermediate Layer in Transformer | 3072 | 4096 | 1024 |
| Output Size of Transformer Layer | 768 | 1024 | 256 |
| Pooler Output Size | 768 | 1024 | 256 |
| Number of Parameters | 110M | 340M | 11M |
| Additional Components | None (General-purpose model) | None (General-purpose model) | Dropout (for regularization) and Classifier (linear layer for classification tasks) |

## 6) TOKENIZER TRAINING

$$T_d = \text{map}(v')  \qquad (9)$$

Finally, the function $\text{map}(v')$ trains the tokenizer $T_d$. The trained tokenizer now maps future text inputs to the established vocabulary $v'$, effectively turning unstructured text into a form that `SecurityBERT` can process. The trained tokenizer $T_d$ can take any text segment from the document $'d'$ into a series of tokens that `SecurityBERT` can understand.

These steps allow us to transform the raw text into a numerical representation that `SecurityBERT` can process effectively.

## H. FINE TUNING SecurityBERT

After the pre-training stage, we fine-tuned `SecurityBERT` for the cyber threat detection classification task. We added one linear layer followed by a Softmax activation function on top of the pre-trained `SecurityBERT` model, and the entire network is fine-tuned using our labeled data. This process enables `SecurityBERT` to adapt its learned contextual representations to the specific threat detection requirements, improving performance.

### 1) TRAINING SETUP

The training and fine-tuning were conducted on an Intel Xenon(R) 2.20 GHz CPU and an Nvidia A100 GPU with 40GB of RAM. The training on this specific hardware configuration was completed in 1 hour and 47 minutes. The following Section will extensively discuss a comprehensive performance evaluation of our novel `SecurityBERT` model.

## I. LAYERS OF THE SecurityBERT MODEL

Throughout the research, the primary objective was to attain exceptional accuracy in data classification while ensuring the model's size remained compact, with a focus on optimizing performance. After extensive experiments, the final model comprises 15 layers, specifically engineered to accurately

comprehend PPFLE data while mitigating overfitting issues by incorporating suitable dropout layers. The comprehensive structure of the 15-layered `SecurityBERT` is illustrated in FIGURE 5. In the architectural design of `SecurityBERT`, we utilized just 4 Encoder Layers and modified the original parameters to suit our problem better. Additionally, we introduced a new layer in the final stage, comprising a Dropout layer, another new layer, and a Classifier Layer. TABLE 3 highlights the key parameter differences between the original two BERT models and `SecurityBERT`.

### 1) BERT EMBEDDINGS

The BERT Embeddings section starts with Word embeddings, succeeded by Position embeddings and then Token type embeddings. To stabilize the activations, there is a Layer Normalization with a size of 128, followed by a Dropout layer with a rate of 0.1.

### 2) BERT SELF ATTENTION

The BERT Self Attention comprises three primary linear transformations for the Key, Query, and Value. Each of these transformations has input and output features sized at 128. Another Dropout layer with a rate of 0.1 is included to prevent overfitting.

### 3) BERT SELF OUTPUT

The BERT Self Output section features a Linear dense layer with an input and output feature size of 128. A Layer Normalization complements this, also sized at 128, and a Dropout layer with a rate of 0.1 for regularization.

### 4) BERT INTERMEDIATE

In the BERT Intermediate part, there's a dense layer with input features of 128 and output features expanded to 512. This section employs the GELU activation function.

### 5) BERT OUTPUT

In the BERT Output segment of the model, the final layer is a Linear dense layer that transforms the 512 features back to 128.
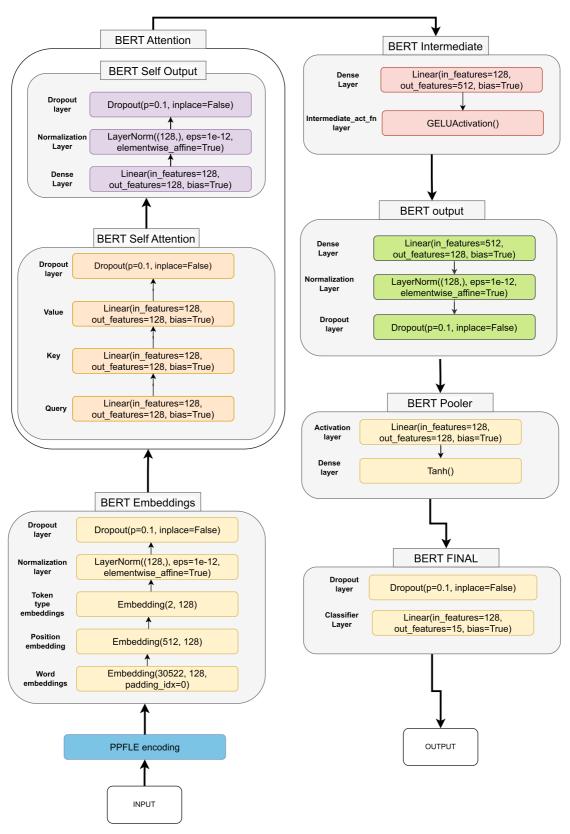
**FIGURE 5.** SecurityBERT architecture.

### 6) BERT POOLER+BERT FINAL

After a Tanh activation in the Bert Pooler, the output is streamlined through another Linear layer, further reducing the feature size to 15, representing the final output. This reduction is a crucial aspect of the model, preparing it for the 15 distinct classification tasks (14 attacks + 1 normal traffic).

### J. MODEL PARAMETERS

The precise parameter choices are among the most critical aspects of a BERT model. Incorrectly selected parameters can significantly influence the model's performance. The model uses a Byte-Pair Encoding Tokenizer, which provides a reliable and effective means of splitting input data into manageable tokens. The training data utilized by the model amounts to 661,767,168 tokens, with a limited vocabulary size of 5000. The minimum token frequency for `SecuriyBERT` is set at 2, while the model supports a maximum sequence length of 737 and a minimum sequence length of 619. The truncation settings limit the sequence length to a maximum 512, ensuring data consistency and model stability. Special tokens used by the model include <s>, <pad>, </s>, <unk>, and <mask>. Regarding processing power, the model works with a batch size of 128 and a hidden size of 128. The model architecture comprises two hidden layers and utilizes four attention heads to process the input data. The intermediate size is set at 512, and the maximum position embeddings at 512, providing enough room for extensive and complex computations. The model can identify and respond to 14 different attacks, demonstrating its versatility and broad applicability. The `SecuriyBERT` is based on 11,174,415 parameters that are fine-tuned for optimal performance. Lastly, the model runs on an Nvidia A100 GPU, a powerful hardware accelerator that enables rapid data processing and real-time response capabilities.

TABLE 4 summarizes the experimental parameter configuration, carefully designed to optimize performance and functionality. With the application of these parameters, our new `SecurityBERT` model exhibits the capability to identify fourteen distinct types of attacks with remarkable accuracy.

## IV. PERFORMANCE EVALUATION OF SecurityBERT

In this section, we evaluate the performance of the newly proposed `SecurityBERT` model, through rigorous testing and comparative analysis. We show that the newly proposed model achieves a remarkable accuracy of 98.2%, which, to the best of our knowledge, stands as the highest accuracy ever attained using an ML algorithm detecting IoT attacks on realistic real-world network traffic.

### A. EXPERIMENTAL RESULTS

To ensure appropriate comparisons with results from other models, we rely on standard measurements, namely Precision, Recall, F1-Score, and Support measurements. These

**TABLE 4.** Configuration and parameres of SecurityBERT.

| Parameter | Value |
|---|---|
| Tokenizer type | Byte-Pair Encoding Tokenizer |
| Training data | 661,767,168 tokens |
| Vocabulary size | 5000 |
| Minimum token frequency | 2 |
| Maximum sequence length | 737 |
| Minimum sequence length | 619 |
| Truncation settings | Max length=512 |
| Special tokens | <s>, <pad>, </s>, <unk>,<mask> |
| Batch size | 128 |
| Hidden size | 128 |
| Number of hidden layers | 2 |
| Number of attention heads | 4 |
| Intermediate-size | 512 |
| Maximum position embeddings | 512 |
| Number of attacks | 14 |
| Total number of parameters | 11,174,415 |
| Hardware accelerator GPU | Nvidia A100 |

metrics are crucial in comprehensively evaluating the model's performance and providing a meaningful assessment of its capabilities.

We partitioned the Edge-IIoTset dataset conventionally, allocating 80% of the samples for training and reserving 20% for evaluation. The model has not previously been exposed to the evaluation data, and we assess its effectiveness using those samples. TABLE 5 presents the distribution of different types of cyber attack samples across training and evaluation data sets.

**TABLE 5.** Distribution of data across 14 attack types.

| Attack Type | Nb. of Samples | Train. Data | Eval. Data |
|---|---|---|---|
| Normal | 1,615,643 | 1,292,514 | 323,129 |
| DDoS_UDP | 121,568 | 88,027 | 22,007 |
| DDoS_ICMP | 116,436 | 93,149 | 23,287 |
| SQL_injection | 51,203 | 40,962 | 10,241 |
| Password | 50,153 | 40,122 | 10,031 |
| Vulnerability_scanner | 50,110 | 40,088 | 10,022 |
| DDoS_TCP | 50,062 | 40,050 | 10,012 |
| DDoS_HTTP | 49,911 | 39,929 | 9,982 |
| Uploading | 37,634 | 30,107 | 7,527 |
| Backdoor | 24,862 | 19,890 | 4,972 |
| Port_Scanning | 22,564 | 18,051 | 4,513 |
| XSS | 15,915 | 12,732 | 3,183 |
| Ransomware | 10,925 | 8,740 | 2,185 |
| MITM | 1,214 | 320 | 80 |
| Fingerprinting | 1,001 | 801 | 200 |
| Max Count | 2,219,201 | 1,765,482 | 441,371 |

TABLE 6 presents the detailed classification report for the `SecuriyBERT` model on various network attack classes.

FIGURE 6 shows the accuracy and loss history during the `SecuriyBERT` training changes over four epochs.

### 1) ROC AUC SCORES FOR CYBER THREAT CLASSIFICATION

FIGURE 7 presents various classes' Receiver Operating Characteristic Area Under the Curve (ROC AUC) scores. These scores indicate the `SecurityBERT` model's performance, with a value of 1.0 being perfect. Classes "Normal", "UDP", "TCP", and "MITM" demonstrate perfect classification with an AUC score of 1.0, which suggests that the model can flawlessly differentiate these
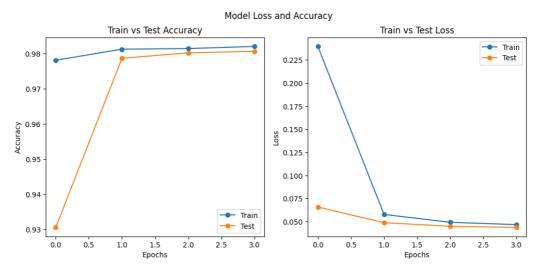
**FIGURE 6.** Accuracy and loss history of SecurityLLM training in 4 epochs.

**TABLE 6.** Classification report of SecurityBERT.

| Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| Normal | 1.00 | 1.00 | 1.00 | 323,129 |
| DDoS_UDP | 1.00 | 1.00 | 1.00 | 22,007 |
| DDoS_ICMP | 1.00 | 1.00 | 1.00 | 23,287 |
| SQL_injection | 0.85 | 0.83 | 0.84 | 10,241 |
| Password | 0.85 | 0.81 | 0.83 | 10,031 |
| DDoS_TCP | 1.00 | 1.00 | 1.00 | 10,012 |
| DDoS_HTTP | 0.89 | 0.99 | 0.94 | 9,982 |
| Vul_scanner | 1.00 | 0.94 | 0.97 | 10,022 |
| Uploading | 0.79 | 0.86 | 0.83 | 7,527 |
| Backdoor | 0.82 | 0.94 | 0.87 | 4,972 |
| Port_Scanning | 0.87 | 1.00 | 0.93 | 4,513 |
| XSS | 0.94 | 0.76 | 0.84 | 3,183 |
| Ransomware | 1.00 | 0.40 | 0.57 | 2,185 |
| Fingerprinting | 0.00 | 0.00 | 0.00 | 200 |
| MITM | 1.00 | 1.00 | 1.00 | 80 |
| Macro Avg | 0.87 | 0.84 | 0.84 | 441,371 |
| Weighted Avg | 0.98 | 0.98 | 0.98 | 441,371 |
| **Accuracy** | **0.982 (98.2%)** | | | |

"XSS", and "Rans" all have AUC scores exceedingly close to 1.0, ranging from approximately 0.9976 to 0.999988. This implies a near-perfect classification for these classes, with very minor misclassifications. On the lower end of the performance spectrum, the class "Fing" has an AUC score of 0.991569, which, while still indicative of strong performance, means it has a slightly higher misclassification rate than the other classes. The `SecurityBERT` model generally exhibits stellar performance across all classes, with almost all of them achieving near-perfect or perfect classification.

### 2) CONFUSION MATRIX

A visual depiction of the confusion matrix from the `SecuriyBERT` classification is presented in FIGURE 8.



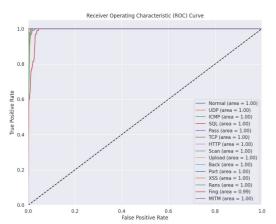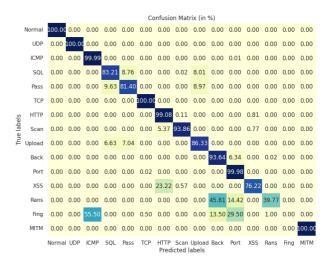**FIGURE 7.** ROC AUC Scores for Cyber Threat Classification.



**FIGURE 8.** Confusion matrix of SecurityBERT classification.

classes from the others. The classes "ICMP", "SQL", "Pass", "HTTP", "Scan", "Upload", "Back", "Port",

For the 'Normal' class and most types of DDoS attacks, including 'DDoS_UDP', 'DDoS_ICMP', and 'DDoS_TCP', the model achieved perfect scores in terms of precision,

recall, and F1-score, showing a high accurate classification performance on these types (c.f. TABLE 6). It is noteworthy to mention the high support count for the 'Normal' class, which amounts to 323, 129 instances. The performance on 'SQL_injection', 'Password', 'DDoS_HTTP', 'Uploading', 'Backdoor', and 'Port_Scanning' classes was relatively lower but still commendable, with F1-scores ranging from 0.83 to 0.94. Notably, 'DDoS_HTTP' and 'Port_Scanning' achieved a remarkably high recall of 0.99 and 1.00, respectively, indicating that the model could identify almost all instances of these attacks when they occur. 'Vul_scanner' had a high precision and a slightly lower recall of 0.94, resulting in an F1-score of 0.97, showing good performance in identifying this type of attack. 'Ransomware' showed a high precision of 1.00, but with a significantly lower recall of 0.40, resulting in an F1-score of 0.57, suggesting that while the model made correct predictions for the 'Ransomware' class, it missed a significant portion of actual instances.

An examination of the confusion matrix reveals that, while the ransomware classification did experience misclassification in a substantial proportion of instances, most misclassifications occurred within the 'Backdoors' category. This category bears notable similarities with the ransomware category in real-life traffic data. Consequently, if our model misclassifies ransomware as a backdoor, it will not have a significant impact, maintaining satisfactory results in practical applications. The classes 'XSS' and 'MITM' showed good performance with F1-scores of 0.84 and 1.00, respectively, demonstrating that the model handled these classes well. Interestingly, the 'Fingerprinting' class had a precision, recall, and F1-score of 0, indicating a complete misclassification for these instances by the model. We again highlight that, much like the backdoor-ransomware misclassification scenario, a considerable proportion of the 'Fingerprint' misclassifications pertain to the 'ICMP' class. Misclassifying 'Fingerprint' as 'XSS,' for example, would ordinarily be a substantial issue in misclassification. However, in practical applications, these misclassifications bear no real consequence since the 'Fingerprint' and 'ICMP' classes closely resemble each other.

The average recall and F1-score were all 0.84 on the macro level. The weighted average was considerably higher at 0.98 for all three metrics, suggesting a good performance overall. The slight difference between these two averages may be due to the imbalanced nature of the dataset, as classes with larger support have a greater influence on the weighted average. The overall accuracy of the model, measuring the proportion of correct predictions made out of all predictions, was 0.982, showing a high degree of the predictive power of the `SecurityBERT` model in identifying different types of network attacks.

### 3) WeightWatcher - EMPIRICAL SPECTRAL DENSITY (ESD)

WeightWatcher (WW) is an open-source diagnostic tool designed for examining Deep Neural Networks (DNNs) and can analyze various layers within a model. WeightWatcher can assist in identifying signs of overfitting and underfitting within particular layers of pre-trained or trained DNNs. We employed WW to optimize performance throughout our experiments, modifying the model's parameters to achieve optimal results. FIGURE 9 presents the Power Law (PL) exponent ($\alpha$) values when plotted against layer IDs, revealing intriguing insights into the weight matrix properties of `SecurityBERT`. FIGURE 10 presents the Empirical Spectral Density (ESD) for Layer 14. FIGURE 11 presents the Log-Lin Empirical Spectral Density (ESD) for Layer 14.
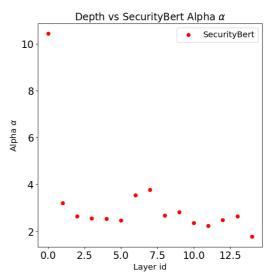


**FIGURE 9.** Power Law (PL) exponent ($\alpha$) values.

Initial layers, especially the first, show a significantly high $\alpha$ value of around 10.43, suggesting a distinct weight initialization or early layer behavior. As we progress deeper into the network, the $\alpha$ values stabilize around 2 to 3, with many layers hovering close to the 2 mark. An $\alpha$ value near 2 indicates weight matrices possessing heavy-tailed properties, which, according to [34], smaller values ($\alpha \approx 2$) are associated with models that generalize better.

According to the measurements presented, `Security-BERT` can generalize effectively to new data that closely resembles the patterns observed during testing on the training dataset.

### B. PERFORMANCE COMPARISON

Numerous research studies have assessed the accuracy of detecting the 14 attacks in the Edge-IIoTset dataset. In this section, we have specifically analyzed research conducted by various authors.
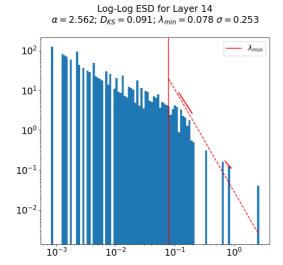
The creators of the Edge-IIoTset dataset tested various traditional ML algorithms on it, including Decision Tree (DT), Random Forest (RF), Support Vector Machines (SVM), and K-Nearest Neighbor (KNN). Among these traditional methods, DT exhibited the lowest performance with an accuracy of only 67.11%, while RF outperformed the others with an accuracy of 80.83%. In addition to these traditional
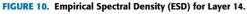
**TABLE 7.** Comparison of SecurityBERT with traditional ML and DL models.

| AI type | Authors | Year | AI Model | Accuracy |
|---|---|---|---|---|
| Traditional ML | Ferrag et al. [9] | 2022 | Decision Tree (DT) | 67.11% |
| | Ferrag et al. [9] | 2022 | Random Forest (RF) | 80.83% |
| | Ferrag et al. [9] | 2022 | Support Vector Machines (SVM) | 77.61% |
| | Aouedi et al. [27] | 2023 | DT + RF / FL | 90.91% |
| | Zhang et al. [35] | 2023 | K-Nearest Neighbor (KNN) | 93.78% |
| | Ferrag et al. [9] | 2022 | K-Nearest Neighbor (KNN) | 79.18% |
| Deep learning models | Friha et al. [23] | 2023 | CNN / CL / No-DP | 94.84% |
| | Friha et al. [23] | 2023 | CNN / FL / No-DP | 93.96% |
| | Aljuhani et al. [36] | 2023 | CSAE + ABiLSTM | 94.40% |
| | Friha et al. [17] | 2022 | Recurrent Neural Network (RNN) | 94% |
| | Ding et al. [37] | 2023 | Long short-term memory (LSTM) | 94.96% |
| | Ferrag et al. [9] | 2022 | Deep Neural Network (DNN) | 94.67% |
| | Friha et al. [17] | 2022 | Deep Neural Network (DNN) | 93% |
| | E. M.d. Elias et al. [38] | 2022 | CNN-LSTM | 97.14% |
| | Ferrag et al. [29] | 2023 | Transformer model w/o Tokenization and Embedding | 94.55% |
| Large language model | - | - | BERT without PPFLE | 51.3% |
| | This work | 2024 | SecurityBERT with PPFLE | **98.20%** |

CNN: Convolutional Neural Network, CL: Centralized Learning, FL: federated learning, DP: Differential Privacy, CSAE: Contractive Sparse AutoEncoder, ABiLSTM: Attention-based Bidirectional Long Short Term Memory.
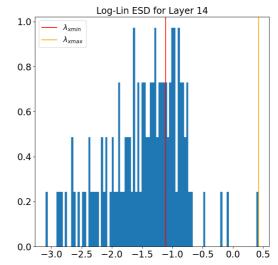


**FIGURE 10.** Empirical Spectral Density (ESD) for Layer 14.



**FIGURE 11.** Log-Lin Empirical Spectral Density (ESD) for Layer 14.

algorithms, a Deep Neural Network (DNN) test was conducted, which outperformed the others, boasting an accuracy of 94.67%. The ultimate objective of this research is to develop a model capable of achieving nearly flawless real-time accuracy while maintaining a relatively compact model size suitable for deployment on IoT-embedded devices. This requirement explicitly rules out resource-intensive solutions like utilizing pre-trained LLMs, which, although capable of delivering high accuracy, are impractical for constrained devices regarding real-time packet analysis due to their significant resource demands. Following the initial dataset release, numerous authors tried to enhance accuracy using various model combinations and novel architectural designs. TABLE 7 presents the comparative accuracy of the proposed model, namely `SecurityBERT`, against the traditional ML models and Deep Learning (DL) models.

Friha et al. [23] explored the potential of CNNs to exceed the 95% accuracy. They experimented with various setups, including Centralized Learning (CL) and Federated Learning (FL), both with and without Differential Privacy (DP). Using CL without DP, their best model attained an accuracy of 94.84%. E. M.d. Elias et al. combined the CNN approach with Long-Short Term Memory (LSTM). This combination surpassed the 95% threshold, reaching an impressive accuracy of 97.14% on the Edge-IIoTset dataset, solely by extracting features from the transport and network layers. A recent research paper by Ferrag et al. [29] explored an innovative method. They introduced a simple GAN and Transformer-based architecture without any tokenization or embeddings. The model obtained a 94.55% accuracy rate, and in this study, they raised the question of whether tokenization could pose challenges when applied to IoT datasets due to the unstructured nature of IoT network data, making it

challenging to capture the semantics of closely resembling patterns like TCP scans or UDP scans.

The main objective of our paper was to create a novel model capable of exceeding an already exceptionally high level of accuracy. To the best of our knowledge, we have reached a record-breaking accuracy of 98.2% in classifying the 14 types of attacks, as showcased in TABLE 6. This achievement represents the highest accuracy ever achieved in the multiclassification of these attack categories.

### C. REAL-LIFE ENVIRONMENT INTEGRATION

In the original Edge-IIoTset dataset, feature extraction is derived from genuine PCAP files. This implies that replicating the same results in real-life scenarios becomes feasible if we possess real traffic data. We can substitute the PCAP files used in the Edge-IIoTset dataset [9] with real-life internal network traffic, employing a suitable sniffing tool to generate the PCAP file. FIGURE 12 provides a visual representation of the experimental setup where `SecurityBERT` is seamlessly integrated into a real-life network environment. This system is designed to detect network incidents with remarkable accuracy, leveraging real-time network packet data.

#### 1) INFERENCE TIME

To implement the model on IoT devices, evaluating whether the inference time is sufficiently fast is essential. If the model is overly complex and exhibits slow inference times on an average CPU, its viability in real-world environments becomes questionable. TABLE 8 offers a detailed comparative analysis of computation times across different hardware platforms, specifically focusing on the inference task of the `SecuriyBERT` model. The figures in the table denote the average inference time derived from 1000 measurements.

**TABLE 8.** Inference time of SecuriyBERT across different hardware platforms.

| Hardware | Average Inference Time (sec) |
|---|---|
| A100 GPU | 0.0164 |
| T4 GPU | 0.0244 |
| V100 GPU | 0.0277 |
| TPU | 0.0327 |
| CPU* | 0.1582 |

*Intel(R) Xeon(R) CPU @ 2.20GHz

The devices evaluated include three NVIDIA GPU models (A100, T4, and V100), Google's Tensor Processing Unit (TPU), and a general-purpose CPU. Each entry, denoted in seconds, reflects each hardware platform's time to perform the inference using `SecuriyBERT`. The A100 GPU is the most efficient in this context, completing the inference quickly. The pivotal metric here is the 0.15 sec CPU inference time, signifying that the model can be efficiently deployed on resource-limited devices for analyzing real-life traffic. Additionally, given its compact size of just 16.7 MB, the model is well-suited for deployment in embedded devices.
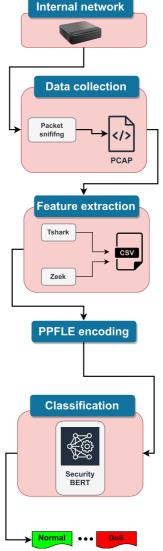


**FIGURE 12.** Real-life experimental setup using SecurityBERT.

#### 2) REDUCING MTTR

Integrating `SecurityBERT` into an embedded device and deploying it within an IoT network makes it possible to substantially enhance detection accuracy and leverage its high-speed performance to identify malicious activities within internal networks in real-time quickly. This, in turn, can lead to a notable reduction in Mean Time to Remediate (MTTR).

Implementing AI in software security and incident handling is not a recent development in software security. For instance, companies like Rubrik[2] and Microsoft have adopted generative AI models to optimize operations and enhance efficiency. For instance, if Rubrik's Security Cloud machine detects abnormal behavior, it automatically generates an incident in Microsoft's Sentinel. By employing this proactive

[2]https://www.rubrik.com/products

approach, they can achieve faster response times and more effective management of potential security threats.

Similarly, `SecurityBERT` can be seamlessly integrated into existing real-world systems, thereby augmenting the overall accuracy and detection rate of these pre-existing systems.

## V. CONCLUSION

The innovative application of BERT architecture for cyber threat detection, embodied in `SecurityBERT`, demonstrated remarkable efficiency, contradicting initial assumptions regarding its incompatibility due to the reduced significance of syntactic structures. Experimental results underscored the superiority of this approach over conventional ML and DL models, including CNNs, deep learning networks, and recurrent neural networks. The `SecurityBERT` model, tested on a collected cybersecurity dataset, exhibited an outstanding capability to identify fourteen distinct types of attacks with an accuracy rate of 98.2%.

While this paper has made significant progress in advancing the use of LLMs in cybersecurity, future research directions can take several routes to enhance these promising findings further. One potential avenue involves fine-tuning and expanding the `SecurityBERT` model to enhance its performance across various attack types, incorporating adversarial attacks and more complex threats. In addition, due to the evolving nature of cyber threats, continuous updating and training of `SecurityBERT` model on the latest real-world datasets will be imperative to maintain its efficacy.

An exciting and promising avenue for future research is delving into methods to autonomously implement mitigations based on the classification of `SecurityBERT`. This advancement could lead to automated patch management, antivirus management, network reconfiguration, port management, and numerous other facets of cybersecurity management.

## REFERENCES

[1] Statista Report, *Internet of Things (IoT)—Statistics & Facts*, Blog Post, Sep. 2023. [Online]. Available: https://www.statista.com/aboutus/

[2] N. Moustafa, N. Koroniotis, M. Keshk, A. Y. Zomaya, and Z. Tari, "Explainable intrusion detection for cyber defences in the Internet of Things: Opportunities and solutions," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 3, pp. 1775–1807, 3rd Quart., 2023.

[3] S. Silvestri, S. Islam, S. Papastergiou, C. Tzagkarakis, and M. Ciampi, "A machine learning approach for the NLP-based analysis of cyber threats and vulnerabilities of the healthcare ecosystem," *Sensors*, vol. 23, no. 2, p. 651, Jan. 2023.

[4] C. Thapa, S. I. Jang, M. E. Ahmed, S. Camtepe, J. Pieprzyk, and S. Nepal, "Transformer-based language models for software vulnerability detection," in *Proc. 38th Annu. Comput. Secur. Appl. Conf.*, Dec. 2022, pp. 481–496.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.

[6] A. Rahali and M. A. Akhloufi, "MalBERT: Malware detection using bidirectional encoder representations from transformers," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2021, pp. 3226–3231.

[7] Y. E. Seyyar, A. G. Yavuz, and H. M. Ünver, "An attack detection framework based on BERT and deep learning," *IEEE Access*, vol. 10, pp. 68633–68644, 2022.

[8] S. Chen and H. Liao, "BERT-log: Anomaly detection for system logs based on pre-trained language model," *Appl. Artif. Intell.*, vol. 36, no. 1, Dec. 2022, Art. no. 2145642.

[9] M. A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, and H. Janicke, "Edge-IIoTset: A new comprehensive realistic cyber security dataset of IoT and IIoT applications for centralized and federated learning," *IEEE Access*, vol. 10, pp. 40281–40306, 2022.

[10] N. Alkhatib, M. Mushtaq, H. Ghauch, and J.-L. Danger, "CAN-BERT do it? Controller area network intrusion detection system based on BERT language model," in *Proc. IEEE/ACS 19th Int. Conf. Comput. Syst. Appl. (AICCSA)*, Dec. 2022, pp. 1–8.

[11] E. Aghaei, X. Niu, W. Shadid, and E. Al-Shaer, "SecureBERT: A domain-specific language model for cybersecurity," in *Proc. Int. Conf. Secur. Privacy Commun. Syst.* Cham, Switzerland: Springer, 2022, pp. 39–56.

[12] P. Ranade, A. Piplai, A. Joshi, and T. Finin, "CyBERT: Contextualized embeddings for the cybersecurity domain," in *Proc. IEEE Int. Conf. Big Data*, Dec. 2021, pp. 3334–3342.

[13] K. Yu, L. Tan, S. Mumtaz, S. Al-Rubaye, A. Al-Dulaimi, A. K. Bashir, and F. A. Khan, "Securing critical infrastructures: Deep-learning-based threat detection in IIoT," *IEEE Commun. Mag.*, vol. 59, no. 10, pp. 76–82, Oct. 2021.

[14] B. Breve, G. Cimino, and V. Deufemia, "Identifying security and privacy violation rules in trigger-action IoT platforms with NLP models," *IEEE Internet Things J.*, vol. 10, no. 6, pp. 5607–5622, Mar. 2023.

[15] Z. Wang, J. Li, S. Yang, X. Luo, D. Li, and S. Mahmoodi, "A lightweight IoT intrusion detection model based on improved BERT-of-Theseus," *Exp. Syst. Appl.*, vol. 238, Mar. 2024, Art. no. 122045.

[16] D. Hamouda, M. A. Ferrag, N. Benhamida, and H. Seridi, "PPSS: A privacy-preserving secure framework using blockchain-enabled federated deep learning for industrial IoTs," *Pervas. Mobile Comput.*, vol. 88, Jan. 2023, Art. no. 101738.

[17] O. Friha, M. A. Ferrag, L. Shu, L. Maglaras, K.-K. R. Choo, and M. Nafaa, "FELIDS: Federated learning-based intrusion detection system for agricultural Internet of Things," *J. Parallel Distrib. Comput.*, vol. 165, pp. 17–31, Jul. 2022.

[18] S. Selvarajan, G. Srivastava, A. O. Khadidos, A. O. Khadidos, M. Baza, A. Alshehri, and J. C.-W. Lin, "An artificial intelligence lightweight blockchain security model for security and privacy in IIoT systems," *J. Cloud Comput.*, vol. 12, no. 1, p. 38, Mar. 2023.

[19] Y. Chen, Z. Ding, L. Alowain, X. Chen, and D. Wagner, "DiverseVul: A new vulnerable source code dataset for deep learning based vulnerability detection," 2023, *arXiv:2304.00409*.

[20] M. Douiba, S. Benkirane, A. Guezzaz, and M. Azrour, "An improved anomaly detection model for IoT security using decision tree and gradient boosting," *J. Supercomput.*, vol. 79, no. 3, pp. 3392–3411, Feb. 2023.

[21] H. Jahangir, S. Lakshminarayana, C. Maple, and G. Epiphaniou, "A deep learning-based solution for securing the power grid against load altering threats by IoT-enabled devices," *IEEE Internet Things J.*, vol. 10, no. 12, pp. 10687–10697, Jun. 2023.

[22] F. Hu, W. Zhou, K. Liao, H. Li, and D. Tong, "Towards federated learning models resistant to adversarial attacks," *IEEE Internet Things J.*, vol. 10, no. 19, pp. 16917–16930, Oct. 2023.

[23] O. Friha, M. A. Ferrag, M. Benbouzid, T. Berghout, B. Kantarci, and K.-K.-R. Choo, "2DF-IDS: Decentralized and differentially private federated learning-based intrusion detection system for industrial IoT," *Comput. Secur.*, vol. 127, Apr. 2023, Art. no. 103097.

[24] C. Chakraborty, S. M. Nagarajan, G. G. Devarajan, T. V. Ramana, and R. Mohanty, "Intelligent AI-based healthcare cyber security system using multi-source transfer learning method," *ACM Trans. Sen. Netw.*, May 2023. [Online]. Available: https://doi.org/10.1145/3597210

[25] X. Wang, C. Fang, M. Yang, X. Wu, H. Zhang, and P. Cheng, "Resilient distributed classification learning against label flipping attack: An ADMM-based approach," *IEEE Internet Things J.*, vol. 10, no. 17, pp. 15617–15631, Sep. 2023.

[26] J. Liu, Y. Tang, H. Zhao, X. Wang, F. Li, and J. Zhang, "CPS attack detection under limited local information in cyber security: An ensemble multi-node multi-class classification approach," *ACM Trans. Sensor Netw.*, vol. 20, no. 2, pp. 1–27, Mar. 2024.

[27] O. Aouedi and K. Piamrat, "F-BIDS: Federated-blending based intrusion detection system," *Pervas. Mobile Comput.*, vol. 89, Feb. 2023, Art. no. 101750.

[28] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *J. Inf. Secur. Appl.*, vol. 50, Feb. 2020, Art. no. 102419.

[29] M. A. Ferrag, M. Debbah, and M. Al-Hawawreh, "Generative AI for cyber threat-hunting in 6G-enabled IoT networks," in *Proc. IEEE/ACM 23rd Int. Symp. Cluster, Cloud Internet Comput. Workshops (CCGridW)*, May 2023, pp. 16–25.

[30] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, and M. Funtowicz, "HuggingFace's transformers: State-of-the-art natural language processing," 2019, *arXiv:1910.03771*.

[31] Y. Shibata, T. Kida, S. Fukamachi, M. Takeda, A. Shinohara, T. Shinohara, and S. Arikawa, "Byte pair encoding: A text compression scheme that accelerates pattern matching," Tech. Rep., 1999. [Online]. Available: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=1e944 1bbad598e181896349757b82af42b6a6902

[32] A. Araabi, C. Monz, and V. Niculae, "How effective is byte pair encoding for out-of-vocabulary words in neural machine translation?" 2022, *arXiv:2208.05225*.

[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–11.

[34] C. H. Martin, T. Peng, and M. W. Mahoney, "Predicting trends in the quality of state-of-the-art neural networks without access to training or testing data," *Nature Commun.*, vol. 12, no. 1, p. 4122, Jul. 2021.

[35] X. Zhang, L. Hao, G. Gui, Y. Wang, B. Adebisi, and H. Sari, "An automatic and efficient malware traffic classification method for secure Internet of Things," *IEEE Internet Things J.*, early access, Sep. 22, 2023, doi: 10.1109/JIOT.2023.3318290.

[36] A. Aljuhani, P. Kumar, R. Alanazi, T. Albalawi, O. Taouali, A. K. M. Islam, N. Kumar, and M. Alazab, "A deep learning integrated blockchain framework for securing industrial IoT," *IEEE Internet Things J.*, early access, Sep. 18, 2024, doi: 10.1109/JIOT.2023.3316669.

[37] W. Ding, M. Abdel-Basset, and R. Mohamed, "Deepak-IoT: An effective deep learning model for cyberattack detection in IoT networks," *Inf. Sci.*, vol. 634, pp. 157–171, Jul. 2023.

[38] E. M. D. Elias, V. S. Carriel, G. W. De Oliveira, A. L. Dos Santos, M. Nogueira, R. H. Junior, and D. M. Batista, "A hybrid CNN-LSTM model for IIoT edge privacy-aware intrusion detection," in *Proc. IEEE Latin-Amer. Conf. Commun. (LATINCOM)*, Nov. 2022, pp. 1–6.

**MTHANDAZO NDHLOVU** is currently a Senior Security Researcher with the Digital Security Unit, Technology Innovation Institute (TII), Abu Dhabi, United Arab Emirates. He is also an Alumni of 42 Paris, France, and the National University of Science and Technology, Bulawayo, Zimbabwe. He holds the Offensive Security Certified Professional (OSCP) and Certified Red Team Professional (CRTP) Certifications. He is highly skilled in penetration testing, reverse engineering, software development, n-day research, and malware research. His research interests include mobile and windows security, red team operations, NLP, software security, malware analysis, program analysis, and AI for cyber security.

**NORBERT TIHANYI** (Member, IEEE) received the B.Sc. degree in security engineering, the M.Sc. degree in safety engineering, the M.Sc. degree (Hons.) in IT engineering, and the Ph.D. degree in information science and technology from Eötvös Loránd University, Budapest, Hungary, in 2020. He is currently a Public Body Member of the Hungarian Academy of Science. He is also a Lead Researcher with the Technology Innovation Institute (TII), Abu Dhabi, United Arab Emirates. His research interests include cryptanalysis, security of embedded devices, and cryptography-related prime number theory.

**MOHAMED AMINE FERRAG** (Senior Member, IEEE) received the bachelor's, master's, Ph.D., and Habilitation degrees in computer science from Badji Mokhtar—Annaba University, Annaba, Algeria, in 2008, 2010, 2014, and 2019, respectively. He was an Associate Professor with the Department of Computer Science, Guelma University, Algeria, from 2014 to 2022. Concurrently, from 2019 to 2022, he held the position of a Senior Researcher with the NAU-Lincoln Joint Research Center of Intelligent Engineering, Nanjing Agricultural University, China. In 2022, he was a Lead Researcher with the Artificial Intelligence & Digital Science Research Center, Technology Innovation Institute, Abu Dhabi, United Arab Emirates. His research interests include spectrum of topics within the cyber security domain, including wireless network security, network coding security, applied cryptography, blockchain technology, generative AI, software security, and the application of AI in cyber security. His scholarly output includes over 140 papers published in international journals and conference proceedings. He has spearheaded numerous projects in research and development, fostering collaborative ties with academic institutions in the U.K., Australia, USA, Canada, and China. His contributions to the field include the creation of two cybersecurity datasets, namely, the edge-IIoT dataset and the FormAI dataset, which have become essential resources for AI researchers worldwide. His academic contributions have been recognized with the 2021 IEEE TEM Best Paper Award, the 2022 Scopus Algeria Award, and many best paper conference awards. He has consistently been named on Stanford University's list of the world's top 2% of scientists four times, from 2020 to 2023. He also contributes to the academic community as an Associate Editor for prestigious journals, such as the IEEE INTERNET OF THINGS JOURNAL and *ICT Express* (Elsevier).

**LUCAS C. CORDEIRO** received the B.Sc. degree in electrical engineering and the M.Sc. degree in computer engineering from the Federal University of Amazonas, Brazil, in 2005 and 2007, respectively, and the Ph.D. degree in computer science from the University of Southampton, U.K., in 2011. He is currently a Reader with the Department of Computer Science, The University of Manchester, U.K., where he leads the Systems and Software Security (S3) Research Group. He is also the Arm Centre of Excellence Director of UoM. In addition, he is affiliated with the Trusted Digital Systems Cluster, Centre for Digital Trust and Society, the Formal Methods Group, The University of Manchester, and the Post-Graduate Programs in Electrical Engineering (PPGEE) and Informatics (PPGI), Federal University of Amazonas, Brazil. Before joining The University of Manchester, he was a Postdoctoral Researcher with the University of Oxford and a Research Engineer with Diffblue. In addition, he worked for five years as a Software Engineer with Siemens/BenQ Mobile and CTPIM/NXP Semiconductors. His work focuses on software model checking, automated testing, program synthesis, software security, and embedded and cyber-physical systems. He has coauthored more than 150 peer-reviewed publications in the most prestigious venues (e.g., ICSE, CAV, TACAS, FSE, ASE, ISSTA, TSE, TR, and TC). He has received various international awards, including the Most Influential Paper Award at ASE'23, the Distinguished Paper Award at ICSE'11, and 46 Awards from the International Competitions on Software Verification (SV-COMP) and Testing (Test-COMP) (2012–2024). He has a proven track record of securing research funding from EPSRC, Intel, Motorola, Samsung, Nokia Institute of Technology, CNPq, FAPEAM, British Council, and Royal Society (career total over USD 13M).

**MEROUANE DEBBAH** (Fellow, IEEE) received the M.Sc. and Ph.D. degrees from École Normale Supérieure Paris-Saclay, France. He was with the Motorola Laboratory, Saclay, France, from 1999 to 2002, and then with the Vienna Research Center for Telecommunications, Vienna, Austria, until 2003. From 2003 to 2007, he was an Assistant Professor with the Mobile Communications Department, Institut Eurecom, Sophia Antipolis, France. Since 2007, he has been a Full Professor with CentraleSupelec, Gif-sur-Yvette, France. From 2007 to 2014, he was the Director of the Alcatel-Lucent Chair on Flexible Radio. From 2014 to 2021, he was the Vice-President of the Huawei France Research Center. He was jointly the Director of the Mathematical and Algorithmic Sciences Laboratory and the Director of the Lagrange Mathematical and Computing Research Center. From 2021 to 2023, he was a Chief Researcher with the Technology Innovation Institute and led the AI and Digital Science Research Center, Technology Innovation Institute. He was also an Adjunct Professor with the Department of Machine Learning, Mohamed bin Zayed University of Artificial Intelligence, Abu Dhabi. Since 2023, he has been a Professor with the Khalifa University of Science and Technology, Abu Dhabi, and the founding Director of the 6G Center. He is currently a Professor with the Khalifa University of Science and Technology. He has managed eight EU projects and over 24 national and international projects. He holds more than 40 patents. His research interests include fundamental mathematics, algorithms, statistics, information, and communication sciences. He is fellow of WWRF, EURASIP, AAIA, and Institut Louis Bachelier; and a Membre Émérite SEE. He received the ERC Grant MORE (Advanced Mathematical Tools for Complex Network Engineering), from 2012 to 2017. He received multiple prestigious distinctions, prizes, and best paper awards (more than 35 best paper awards) for his contributions to both fields and, according to research.com, is ranked as the best scientist in France in electronics and electrical engineering.

connected devices and platforms for the industry (smart city, asset management, and smart grids), rolling out LPWAN networks worldwide. Earlier, he was the Head of IoT Business Development & Strategic Partnerships, he actively participated to the Launch of IoT LoRaTM Alliance, being granted the position of the Vice-Chair of the Strategy Committee, in June 2015. Then, from October 2015 to 2017, he was the Vice-Chairman of the Alliance. He has more than 25 years' experience in the telecom, mobility, and energy industry, steering industrial innovation from spark of ideas to prototypes, products, standards, and systems, rolling out networks while leading new businesses within organizations, such as Alcatel, Samsung, and Sagemcom. He is the author of more than 35 international publications, more than 20 patents, editor of one Wiley book on advanced channel coding, the coauthor in two Wiley books on B3G/4G systems, was a Telecom Expert of the European Commission, Eureka Cluster CELTIC, and the French National Research Agency (ANR). He was the Chair of the M2M Expert Group in eMobility European Technology Platform. In 2010, he was a member of the Telecom Steering Committee of Systematic, a competitiveness pole, and created the IoT/M2M course in Supélec, where he was a Visiting Lecturer for seven years.

**THIERRY LESTABLE** received the Engineering degree from Supélec, France, in 1997, and Ph.D. degree from Supélec/Paris Sud-Orsay, in 2003. He is currently an Acting Chief Researcher with the Digital Security & Telecommunications Units, Artificial Intelligence & Digital Science Research Center (AIDRC), Technology Innovation Institute, United Arab Emirates, covering multi-disciplinary units, such as AI, telecom (B5G/6G), and (cyber) security. Before that, he was the Director, heading the e-Mobility & Automotive Business Department, Sagemcom, France, addressing market and products challenges raised from intelligent and connected vehicles (telematics, DMS, and ADAS) to micro-mobility (e-Bikes). He was a member of the Steering Committee from FIEV Federation, representing major Automotive Equipment Suppliers, where he promoted the benefits of A.I for road safety. Before Automotive, he headed the Industrial IoT Business Department, designing and manufacturing

**NARINDERJIT SINGH THANDI** extensive experience in defense with 23 years across U.K. MoD research and defense companies in the U.K. and Europe as a Technical Expert focusing on land and air secure systems. He worked across strategy, operations, capability development, technology, and implementation, but his real strengths are C4ISR and data exploitation across secure communication systems. He has worked extensively in the U.K., USA, Europe, United Arab Emirates, and Saudi Arabia, focusing on capability development within defense security.

● ● ●