symmetry

MDPI

Article

Internet of Things Meets Vehicles: Sheltering In-Vehicle Network through Lightweight Machine Learning

Junchao Xiao [1,2], Hao Wu [3,*] and Xiangxue Li [2,*]

1 School of Systems Science and Engineering, Sun Yat-Sen University, Guangdong 510006, China; xiaojch3@mail2.sysu.edu.cn
2 School of Software Engineering, East China Normal University, Shanghai 200000, China
3 National Computer Network Emergency Response Technical Team/Coordination Center, Beijing 100029, China
* Correspondence: wuhao@cert.org.cn (H.W.); xxli@cs.ecnu.edu.cn (X.L.)

Received: 17 October 2019; Accepted: 7 November 2019; Published: 8 November 2019

Abstract: An internet of vehicles allows intelligent automobiles to interchange messages with other cars, traffic management departments, and data analysis companies about vehicle identification, accident detection, and danger warnings. The implementation of these features requires Internet of Things system support. Smart cars are generally equipped with many (hundreds or even thousands of) sensors and microcomputers so that drivers gain more information about travel. The connection between the in-vehicle network and the Internet can be leveraged by the attackers in a malicious manner and thus increases the number of ways the in-vehicle network can now be targeted. Protecting increasingly intelligent vehicle systems becomes more difficult, especially because a network of many different devices makes the system more vulnerable than ever before. The paper assumes a generic threat model in which attackers can access the controller area network (CAN) bus via common access points (e.g., Bluetooth, OBD-II, Wi-Fi, physical access, and cellular communication, etc). A machine learning based simplified attention (SIMATT)-security control unit (SECCU) symmetry framework is proposed towards a novel and lightweight anomaly detecting mechanism for the in-vehicle network. For this framework, we propose two new models, SECCU and SIMATT, and obtain state-of-the-art anomaly detecting performance when fusing the former to the latter. Regardless of the training phase or the detection phase, we strive to minimize the computational cost and thereby obtain a lightweight anomaly detection method. In particular, the SECCU model has only one layer of 500 computing cells and the SIMATT model has been improved to reduce its computational costs. Through substantial experiment comparisons (with various classical algorithms, such as LSTM, GRU, GIDS, RNN, or their variations), it is demonstrated that the SIMATT-SECCU framework achieves an almost optimal accuracy and recall rate.

Keywords: in-vehicle network; anomaly detecting; machine learning

1. Introduction

1.1. Internet of Things Meets Vehicles

Internet of vehicles (IoV) allows intelligent automobiles to interchange messages with other cars, traffic management departments, and data analysis companies about vehicle identification, accident detection, and danger warnings. The implementation of these features requires Internet of Things system support, as shown in Figure 1. Traffic sensors collect road information and send it to the driver's mobile phone through network centers. There are some Internet of Things' devices, for example Austria's Autobahn, which uses Cisco's connected devices to link about 80,000 microprocessors and

about 7000 HD cameras for data collection. Huawei deployed a city-level C-V2X network in Wuxi, comprised of RSUs and T-Boxes integrating HiSilicon chips, and this is the world's first end-to-end commercial solution scale deployment to successfully demontrate several C-V2X use cases.

Figure 1. Internet of vehicles: from the sensor web to the Internet of Things.

We are witnessing the upgrade from related websites (i.e., the data collected by the sensors can be obtained via the Internet.) to the Internet of Things (ie., the components embedded in the sensor can communicate with each other via the network, making use of the sensor more intelligently.). The IOT links to multiple smart terminals via the Internet. Driverless vehicles can update their algorithms with user data via links, such as IOT. These driverless cars require a large number of high quality data collection and processing systems. In this case, road information obtained using the IOT can be transmitted to driverless cars. This information includes navigation data and road traffic conditions. All of this data are shared with IOTs on driverless cars and cloud systems to improve intelligent algorithms.

Many (hundreds or even thousands of) electronic and intelligent sensors are equipped in modern vehicles to enhance driver experience. For example, GPS units provide vehicle location information, and the vehicle sound equipment is connected to the phone for functions such as calling and playing music. The Internet of vehicles meets the communication of various devices in the vehicle. The connection between the in-vehicle network and the Internet can be leveraged by attackers in a malicious manner and thus increases the number of ways the in-vehicle network can now be targeted [1]. As an example, through a cellular connection, one may likely gain a vehicle's GPS coordinates, vehicle identification number, its IP address, etc. Protecting increasingly intelligent vehicle systems becomes more difficult, especially because a network of many different devices makes the system more vulnerable than ever before. Recent work [2,3] show that attackers can take advantage of the vulnerabilities of an in-vehicle network to manipulate a vehicle. For example, attackers can attack a vehicle with wireless networks [4].

1.2. Attack Surfaces

If an attacker wants to crack a car they will first consider attack surfaces. Smart cars have many components that can be controlled by an attacker, for example, bluetooth, transaction processing management system (TPMS), Wi-Fi, cellular, infotainment console, USB, controller area network (CAN) bus, and onboard diagnostics II (OBD-II) connector. These components include the external and internal types, as shown in Figure 2, and it may be exploited by an attacker to change the behavior of vehicles. For example, by accessing communication modules such as GPS and cellular networks, an attacker can track the path of vehicles. Also, they can establish a connection via Wi-Fi to access the

remote in-vehicle network, (for up to 500 yards) or to crack the Wi-Fi password to install malware on the smart device.

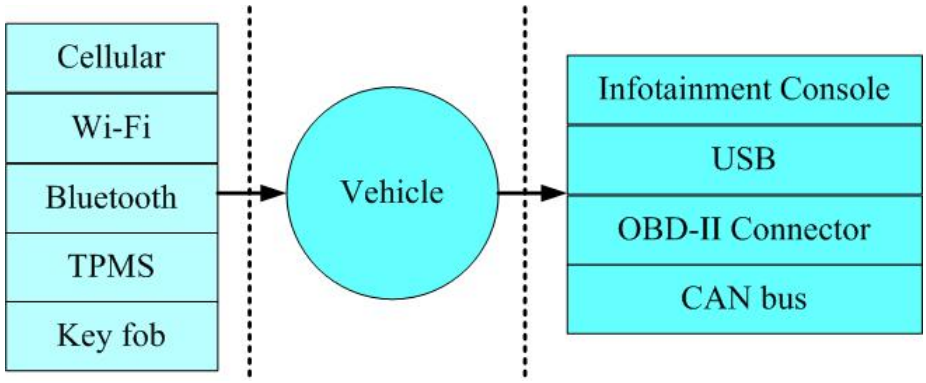

Figure 2. Internet of Things meets vehicles: attack surfaces.

1.3. Can Bus

Currently, electronic control units (ECUs) play an important role in intelligent cars, controlling multiple subsystems. Generally speaking, ECUs can replace traditional mechanical components to simplify automotive design [5]. In order to achieve vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication, the functions of ECUs are further imposed to facilitate larger-scale intra-vehicle [6] and inter-vehicle [7] communication.

A large number of communication protocols are designed for information exchange. The controller area network (CAN) is the de facto standard for communication networks of various vehicles on the market today [8]. On the CAN bus, an ECU can communicate with multiple sensors to accomplish different complex tasks. Important vehicle driving information such as diagnostics, sensors, and control data is transmitted via CAN bus for a variety of vehicle driving functions such as autonomous driving or assisted driving. Therefore, safe driving must be based on the safe transmission of information. However, with the complication of intelligent car components, the number of ECUs has increased exponentially. Protecting increasingly intelligent vehicle systems becomes more difficult, especially because vehicle data spans a variety of different connected devices to further weaken system security. Because vehicle communication security is not the primary concern of the original design of CAN, the in-vehicle network is therefore subject to multiple security threats [9]. For example, an ECU can obtain information from any other ECU on the CAN bus but cannot identify the real identity of the sending ECU. This can make the packet injection and data manipulation attacks easier and endanger the safety of the driver [10].

There are currently two approaches to solving the problem of CAN protocol security. One approach is the use of message authentication codes (MACs), and the other approach is to install an anomaly detecting system on the vehicle [11]. The MAC approach is more reliable, but occupies data fields in a CAN packet. The data fields in a CAN packet are only 8 bits and the maximum transmission rate is only 1 M/s, which does not meet MAC requirements [12]. Therefore, the application of MACs is bound to change the existing vehicle structure and increase production costs, which are currently difficult to widely popularize. In contrast, anomaly detecting [13] does not need to change the structure of the vehicle and can be installed in the vehicle's gateway to detect an attacker's malicious behavior in real time. Therefore, anomaly detecting is more suitable in practice for improving the security of an in-vehicle network.

We assume that the attacker can access the CAN bus and other device interfaces connected to it such as Bluetooth, OBD-II, Wi-Fi, physical access, and USB ports, as shown in Figure 3. Once an attacker gains access to in-vehicle network, it means that the attacker can perform many different modes of attack.

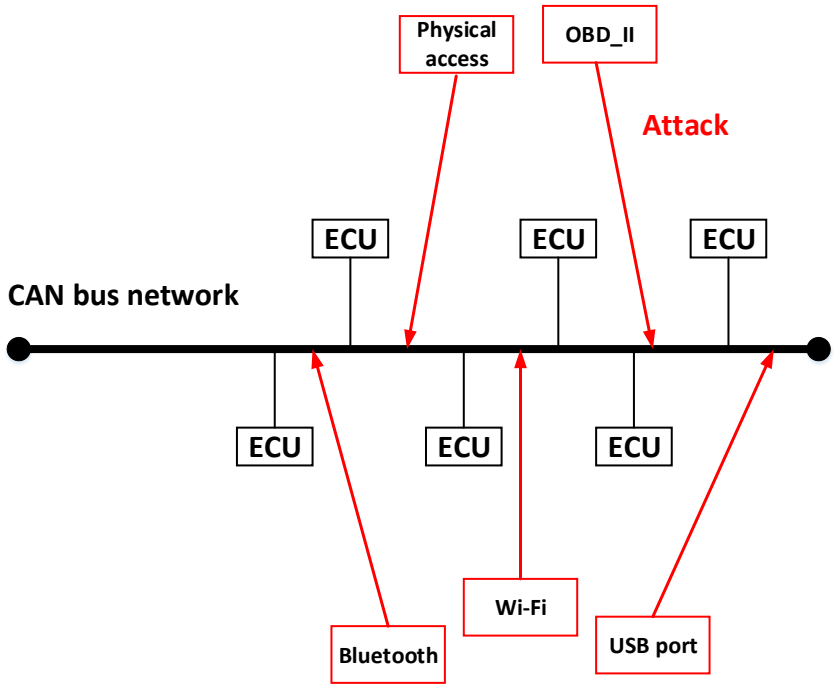

Figure 3. Internet of Things meets vehicles: threat model.

1.4. Our Contributions

In this paper, considering the computational power available in vehicle electronic equipment, we present a lightweight detecting model with the aim of ensuring detection accuracy. The proposed detecting model performs well and can monitor the status of the in-vehicle network in real time. In many anomaly detecting methods, performance is highly dependent on the attack data structure. Namely, most existing methods need to extract appropriate features from the attack data at the training stage and make judgments based on these data during the detecting process. However, the training attack data are generally constructed in an artificially simulated way, and thus quite likely to be different from actual attack data, which may hinder the practical performance of these methods. In contrast, the model we proposed is based on real data collected from the normal operation of a vehicle. The attack data are only used to create a criterion for model evaluation. Our contributions include the following:

1. To suit the actual situation of an in-vehicle network, we present a new recurrent neural network (RNN) model named SECCU.
2. We propose the simplified attention (SIMATT) model that can be computed using the computing power of vehicle electronics and improve detection accuracy.
3. The experimental results demonstrate that our proposed SIMATT-SECCU symmetry framework has the best performance in comparison with many classic models.
4. Our detecting model is fully data-driven and does not require any domain knowledge about the CAN protocol.
5. Our proposed model focuses on a lightweight anomaly detection method for in-vehicle network based on machine learning, which solves the problem of insufficient computing power of networked vehicle electronic devices when applying machine learning algorithms.

The detecting process of the SIMATT-SECCU framework is shown in Figure 4. First, the SIMATT-SECCU model is trained with attack-free (normal operation) and attack data sets to discover the feature distributions of the different data sets. Then, the trained SIMATT-SECCU model can be used for anomaly detecting.

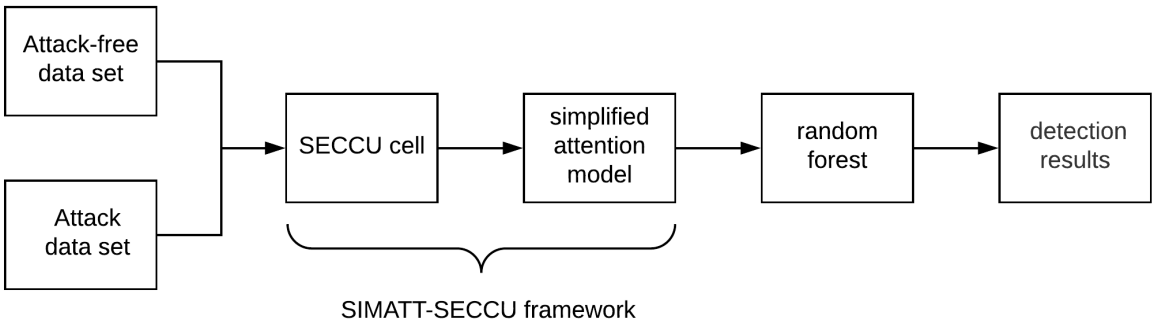

Figure 4. Detecting process of simplified attention (SIMATT)-security control unit (SECCU) framework.

As shown in Table 1, the following abbreviations are used in this manuscript and the rest of this paper is organized as follows. In Section 2, we review related work of the anomaly detecting for in-vehicle network. In Section 3, we present the RNN model and our proposed SECCU model. We further introduce attention models and propose our SIMATT model in Section 4. We descibe the CAN bus and the data sets used for evaluation in Section 5. Section 6 is devoted to the experiments and evaluation results. Finally, we conclude this paper in Section 7.

Table 1. The following abbreviations are used in this manuscript.

CAN	Controller Area Network
SECCU	Security Control Unit
LSTM	Long Short-Term Memory
GRU	Gated Recurrent Unit
HIEATT	HIErarchical ATTention model
SIMATT	SIMplified ention Model
GIDS	GAN Based Intrusion Detection System
RNN	Recurrent Neural Network
HE-RNN	HEuristics and RNNs
IOV	Internet of Vehicles
IOT	Internet of Things
OBD-II	OnBoard Diagnostics II
ECU	Electronics Control Unit
V2V	Vehicle-to-Vehicle
V2I	Vehicle-to-Infrastructure

2. Related Work

The industry and academia have begun to focus on security issues on the CAN bus and has proposed some defense options. The traditional method is to use the heuristic of in-vehicle network behavior anomaly detecting, which is an extension of the detecting mechanism of intrusion detection systems (IDS) for traditional computer networks. Such a detecting system can be installed in different parts on the in-vehicle network. (e.g., the OBD-II port, gateway, or T-box). The IDS monitor the CAN traffic and then obtains a detecting model. When anomalous behavior is detected and the system outputs warning information.

Due to the limitations of the internal computing units of the vehicle, the simple lightweight method is one of the research hotspots. Han et al. [14] proposed an anomaly detecting algorithm based on survival analysis. Song et al. [15] suggested statistical CAN information time interval in anomaly detecting. Although these methods have low computational overhead, they can only detect specific attacks and find unfavorable performance for many attack modes. Machine-learning-based anomaly detecting can solve this problem.

There are several machine-learning-based anomaly detecting methods in the literature. One significant advantage of this approach is that attacks are effectively detected without changing the existing in-vehicle network structure. Seo et al. [16] proposed a double discriminator using

the generative adversarial networks (GAN). The method uses two different data sets to train two discriminators, which greatly improves the detection accuracy of IDS. Tariq et al. [17] proposed an IDS combining heuristic algorithm and recurrent neural network (RNN). In this method, some features of CAN data are first counted to make a preliminary judgment, and then RNN is used for the final judgment. Larson [18] proposed a CAN attack detecting method based on security rules. The method is based on the object dictionary of the CANopen protocol; it uses protocol-level security rules to detect illegal ECU behavior, and Larson provides a set of example security rules. Wang et al. [19] proposed a time series prediction model that trains different types of instructions in the CAN protocol separately before combining them in the final system. Müter et al. [13] detect attacks by calculating the entropy of normal traffic and abnormal traffic on the CAN bus. Hu et al. [20] used a support vector machine model to detect the abnormal state of a vehicle. In [21], Ji et al. uses the clock drift of an ECU to detect abnormal conditions in abnormal vehicles. In [22], Xiao et al. proposed an early warning using convLSTM to predict time series deviations. Table 2 shows some state-of-the-art methods. However, these methods have inherent drawbacks. For example, in [19], the instruction ID of the CAN frame is separated, resulting in a loss of part of the information. In [13], the entropy-based method can only perform preliminary statistics and detect only some attack methods and missing many others. In [20], it was found that directly detecting the abnormal state of a vehicle with a support vector machine requires excessive computational resources, and it is difficult to ensure real-time monitoring. In [21], it was found that vehicles produced by different manufacturers have different ECU clock drift features, so it would be indispensable to reanalyze these for each vehicle brand. An extremely important problem is that the computational cost of machine-learning-based IDSs is considerably high. In contrast, we propose an IDS for in-vehicle network that can address these problems and the required computation shows advantage over preceding methods [16,17].

Table 2. Summary of the intrusion detection systems (IDS) for controller area network (CAN) bus system literature in the automotive domain.

Key References	Detection Strategy	Method
Müter et al. [13]	Anomaly-based	Statistical-based (entropy-based)
Han et al. [14]	Anomaly-based	Survival
Song et al. [15]	Signature-based	Frequency-based
Seo et al. [16]	Anomaly-based	Generative adversarial networks
Tariq et al. [17]	Anomaly-based	Heuristic algorithm and RNN
Larson et al. [18]	Security rules	Object dictionary of the CANopen protocol
Wang et al. [19]	Anomaly-based	Time series prediction model
Hu et al. [20]	Anomaly-based	Support vector machine
Ji et al. [21]	Anomaly-based	Clock drift
Xiao et al. [22]	Anomaly-based	Time series prediction model

3. Recurrent Neural Network

In this section, we will introduce two classic recurrent neural networks (long short-term memory (LSTM) and gated recurrent unit (GRU)) and proposed a new structure (SECCU) for recurrent neural networks.

3.1. Long Short-Term Memory

Long short-term memory (LSTM) units are used in an RNN [23] to create a network that is often called an LSTM network (created to solve long-term memory dependence problems). A traditional LSTM network consists of multiple gates with different functions (Figure 5). The gate can remember the value of time interval, and the information flows in and out of the rules in the three gates.

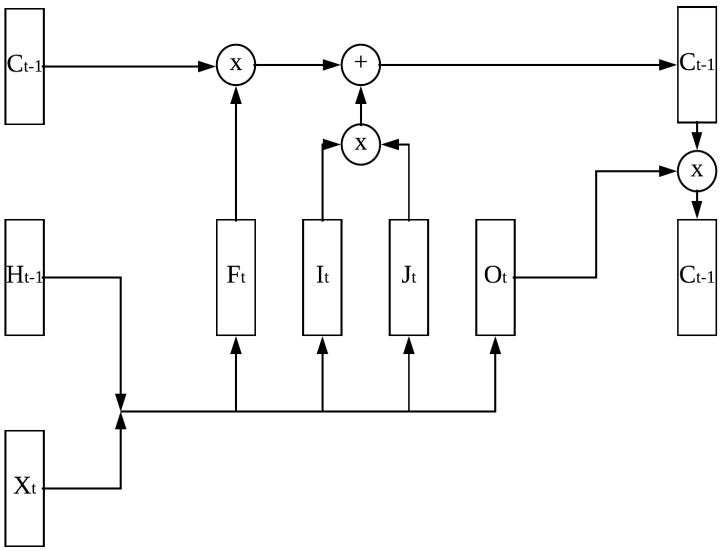

Figure 5. Long short-term memory (LSTM) cell.

LSTM networks are well-suited to the classification, processing, and prediction of time series data because There are undetectable delays in certain important events in the time series. The vanishing gradient problem in training traditional RNNs can be solved on LSTM units. Relative insensitivity to time intervals is an advantage of the LSTM network for other time series algorithms such as RNNs and hidden Markov models. Multiple LSTM networks can be stacked and temporally concatenated to form more complex structures.

The main innovation of LSTM is that memory cells c_t are added to collect state information. When new input data arrive, the information is accumulated in the cell if inputs gate i_t is activated. Simultaneously, if the forget gate f_t is open, past information c_t is forgotten. Output gate o_t determines whether the final output c_t is transmitted to h_t and j_t as the update coefficient of c_t . The main formulas of an LSTM cell are as follows:

$$i_t = tanh(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \tag{1}$$

$$j_t = sigm(W_{xj}x_t + W_{hj}h_{t-1} + b_j) \tag{2}$$

$$f_t = sigm(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \tag{3}$$

$$o_t = tanh(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \tag{4}$$

$$c_t = c_{t-1} \odot f_t + i_t \odot j_t \tag{5}$$

$$h_t = tanh(c_t) \odot o_t \tag{6}$$

In Equation (1), W_* denotes the weight matrices, b_* denotes the biases, and $\odot$ denotes the element-wise vector product.

3.2. Gated Recurrent Unit

In recent years, Cho et al. proposed the gated recurrent unit (GRU), which has a structure similar to that of an LSTM unit [24]. Chung et al. found that GRUs performed better than LSTM units on multiple data sets [25]. The GRU cell model is shown in Figure 6. The main innovation of a GRU is that it combines the input gate and forget gate. A GRU is defined by the following equations:

$$r_t = sigm(W_{xr}x_t + W_{hr}h_{t-1} + b_r) \tag{7}$$

$$z_t = sigm(W_{xz}x_t + W_{hz}h_{t-1} + b_z) \tag{8}$$

$$\widetilde{H}_t = tanh(W_{xh}x_t + W_{hh}(r_t \odot h_{t-1}) + b_h) \tag{9}$$

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \widetilde{H}_t \tag{10}$$

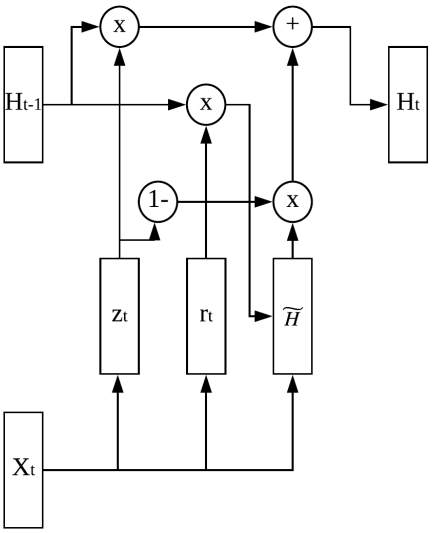

Figure 6. Gated recurrent unit (GRU) cell.

Here, r denotes the reset gate, z denotes the update gate, $\widetilde{H}$ denotes the hidden state of the cell, and H denotes the current state.

3.3. Security Control Unit

In [26], it was found that GRU and its many variants perform better than LSTM units on most tasks. However, LSTM units are more stable in most tasks. As mentioned in [26,27], the LSTM structure is almost optimal and cannot be improved, but the GRU uses multiple structures on different tasks to obtain better performance. Therefore, one can contemplate a model (as a variant of the GRU and combined with the structure of refined LSTM) that achieves stunning performance in dealing with CAN anomaly detecting tasks.

We combine the advantages of LSTM units and GRUs to propose the security control unit (SECCU). The SECCU cell model is shown in Figure 7. The SECCU is defined by the following equations:

$$r_t = relu(x_t + W_{hr}h_{t-1} + b_r) \tag{11}$$

$$z_t = W_{xz}x_t + b_z \tag{12}$$

$$\widetilde{H}_t = relu(W_{hh}(r \odot h_{t-1}) + W_{xh}x_t + b_h) \odot \tag{13}$$

$$z_t + h_{t-1} \odot (1 - z_t) \tag{14}$$

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \widetilde{H}_t \tag{15}$$

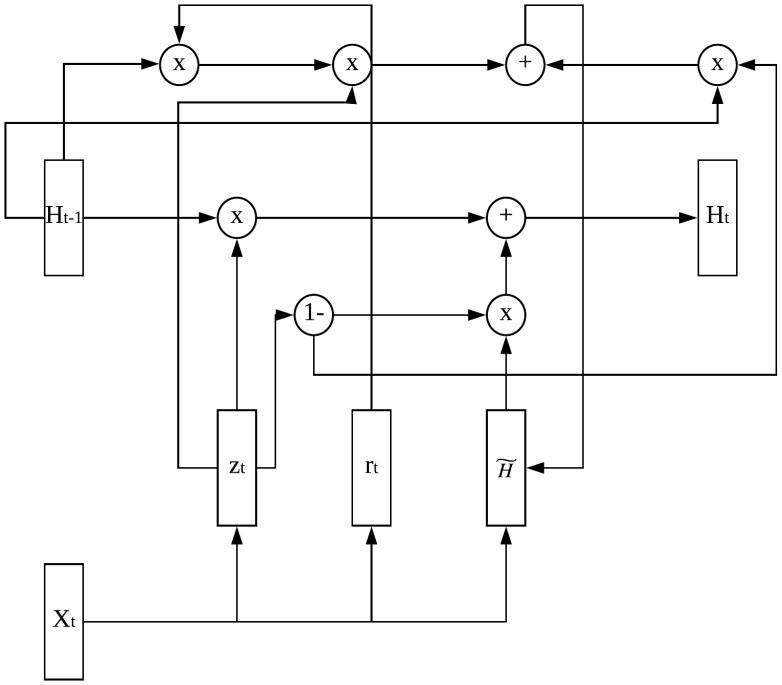

Figure 7. Security control unit (SECCU) cell.

Here, r, z, $\widetilde{H}$, and H have the same meanings as they do in the GRU formulas.

4. Attention Model

In this section, we first introduce the hierarchical attention model and then proposed the simplified attention model to reduce computational overhead.

4.1. Hierarchical Attention Model

An RNN usually calculates a hidden state h_t as a function at time t of previous hidden state h_{t-1}, capturing the correlation and absolute position information along the time dimension directly through its sequential structure. In recent years, attention models based on position coding or distance deviation have performed well on time-series tasks [28]. In particular, the hierarchical attention (HIEATT) model has made great strides in natural language processing. In this paper, we introduce attention models into in-vehicle network for anomaly detecting.

Recently, natural language processing has made substantial advances thanks to the support of the HIEATT model [29]. The instruction sequence generated by the CAN protocol can be regarded as a language, so we can adopt the HIEATT model for vehicle network anomaly detecting. The HIEATT model (shown in Figure 8) is defined by the following equations:

$$u_i = tanh(W_w h_{it} + b_w) \tag{16}$$

$$a_i = \frac{exp(u_{it}^\top u_w)}{\sum_t exp(u_{it}^\top u_w)} \tag{17}$$

$$v = \sum_t a_{it} h_{it} \tag{18}$$

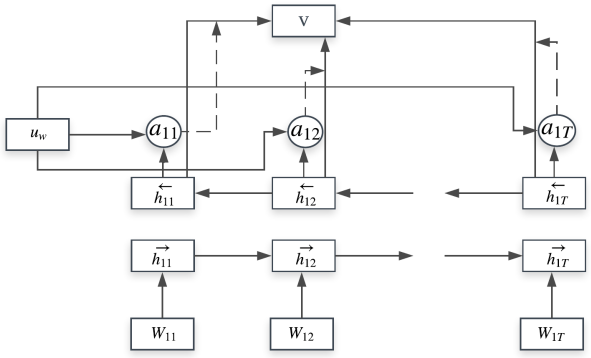

Figure 8. Hierarchical attention model.

That is, we first feed instruction sequence h_{it} through a single-layer RNN to obtain u_{it} as a hidden representation of h_{it}. We then measure the similarity of u_{it} with instruction sequence vector u_w, which indicates its importance. We compute instruction sequence vector a_i as a weighted sum of the instruction sequence. The context vector u_w can be thought of as a high-level representation in memory networks [30].

4.2. Simplified Attention Model

Although the HIEATT model has led to outstanding progress in natural language processing tasks, its application is limited by the computing power of the vehicle electronic devices and the characteristics of the CAN protocol itself. Hence, for an IDS, the HIEATT model must be modified for the environment of an in-vehicle network.

The CAN protocol is not as complex as a natural language, and a better result can be obtained with a slightly simplified attention model (the SIMATT model) for analyzing the instruction sequence generated by the CAN protocol, as shown in Figure 9. The vector of hidden layer state h_t is used to produce a probability vector a of weights. Vector c is the weighted sum of h_t using a. The SIMATT model is defined by the following equations:

$$a_t = softmax(tanh(x_t h_t + b_t)) \tag{19}$$

$$z_t = a_t \odot a_t^{\top} \tag{20}$$

$$c_t = \sum_t z_t \tag{21}$$

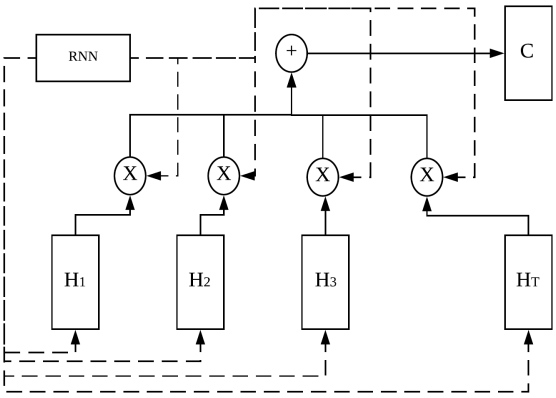

Figure 9. Simplified attention model.

Here, $a^{\top}$ denotes the tensor after the time step axis is interchanged with the sequence length axis in the a tensor and $\odot$ denotes the element-wise vector product. In contrast to the HIEATT model, we added the softmax function and removed the exponential calculation to simplify it and reduce the computational cost.

5. Can Bus and Dataset

In this section, we first briefly describe the implementation principle of CAN BUS, then introduce the datasets and attack methods used in the paper.

5.1. Can Bus

A CAN is a half-duplex high-speed communication bus network with a maximum transmission rate of 1 Mbps. Most automotive companies use the CAN protocol to communicate among ECUs and embedded devices. Figure 10 shows an example CAN packet.

1 bit	11 bits	6 bits	0~8 bytes	16 bits	2 bits	7 bits
Start of Frame	Arbitration Field	Control Field	Data Field	CRC Field	ACK Field	End of Frame

Figure 10. Controller area network (CAN) packet format.

The arbitration ID field consists of 11 bits for the ID, and The ID is the identity of the ECU. The arbitration ID field provides two main functions: (1) The priority of the packet transmission is determined by the ID number. (smaller ID numbers have higher priority) and (2) an ECU can use it to filter corresponding messages. There are data information of 8 bytes in the data field. The control field is used to identify the size of the data field. The cyclic redundancy check field is used to detect errors in data packet transmission. Finally, the acknowledgment field is used to confirm the receipt of a valid CAN packet.

In a CAN, without the concept of address, the data transmitted on the bus can be received by all ECUs. The information is identified by the value of the arbitration fields. The arbitration ID indicates the target device ID of the received message, and any device can communicate with multiple devices as long as multiple arbitration IDs are attached to the package. When multiple packets are transmitted on the bus, packets with a lower arbitration ID would be sent first.

The CAN bus considers the possibility of adding new nodes.Therefore, all messages is transmitted on the bus. Due to CAN is transmitting data by broadcast, the controller on the network can accept all packets without regard to the sender's arbitration ID. Therefore, Malicious devices on the bus can easily send malicious packets.

5.2. Can Bus Dataset

Data is a vital part of every machine learning model. To train these models, it is necessary to employ a data set. However, every modern business must also protect its users' data. To solve the problem of data availability, many institutions and organizations are providing publicly available data sets so that anyone can get them free to build models for educational or commercial use.

Our experimental data is downloaded from a public datasets at http://ocslab.hksecurity.net/Dataset/CAN-intrusion-dataset [31]. The selected data were constructed by logging CAN traffic via the OBD-II port from a real vehicle while message injection attacks were performed.

In this study, we consider two types of attacks, one that directly modifies a vehicle's ECU node and the other that injects unauthenticated information directly into the in-vehicle network. First, we assume that the attacker can physically control and modify the target ECU node in the vehicle so that it stops transmitting information and allows the malicious impersonation node to send information

instead. With respect to the overall information, the system has not changed. We call this attack an impersonation attack [31,32]. In the second type of attack, we assume that the attacker cannot control the target ECU node, but can inject malicious information to induce a fault or remotely maneuver the vehicle. We consider two types of injection attacks, DoS and fuzzy attacks [31], described in detail below.

The data are divided into the following four data sets:

- Attack-free: This data set consists of CAN data generated under normal conditions after the vehicle has been started.
- Impersonation attack: After an attacker has attacked an ECU and disabled it, the attacker inserts his/her ECU for a specific purpose. The inserted ECU is disguised as the disabled ECU, and it can periodically reply to request from other ECU [31,32]. Impersonation attack does not change the original frequency of CAN messages, so IDSs based on frequency characteristics cannot detect this attack. If the attacker does not change the contents of the CAN messages, the inserted ECU looks like a legitimate ECU, and the attacker can then launch other types of attacks. We refer the reader to [31] for further details about the construction and distribution characteristics of the impersonation attack data set. This attack mode is shown in Figure 11.
- DoS attack: This dataset consists of data generated while an attacker periodically injected high-priority CAN instructions so that legitimate instructions are not sent in time. In this data set, the attacker injected the highest priority instruction (ID 0000). This attack mode is shown in Figure 12.
- Fuzzy attack: This dataset consists of data generated while an attacker randomly sends instructions to cause the vehicle to perform unexpectedly. To implement a fuzzy attack, the attacker needs to collect detailed information about the target vehicle. For example, instruction ID that can produce unexpected behavior. It is different from DoS attack by occupancy bus to delay normal information transmission. The Fuzzy attack caused the vehicle function to be unavailable. This attack mode is shown in Figure 13.

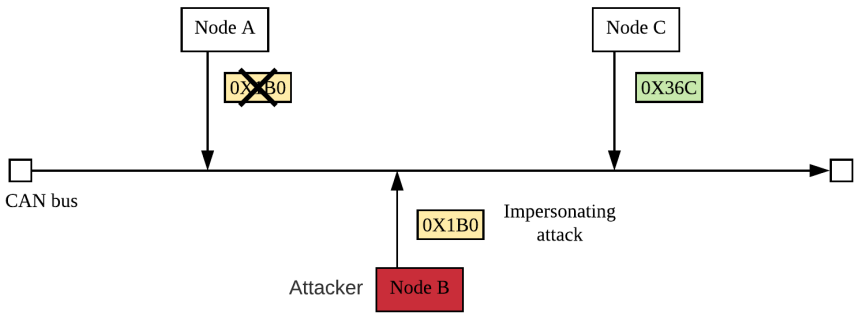

Figure 11. Impersonation attack.

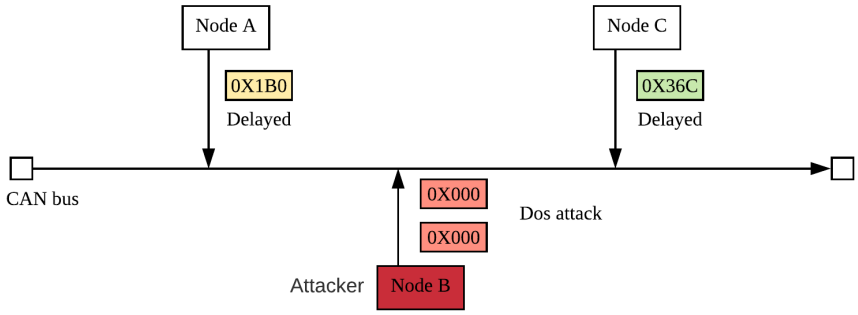

Figure 12. DoS attack.

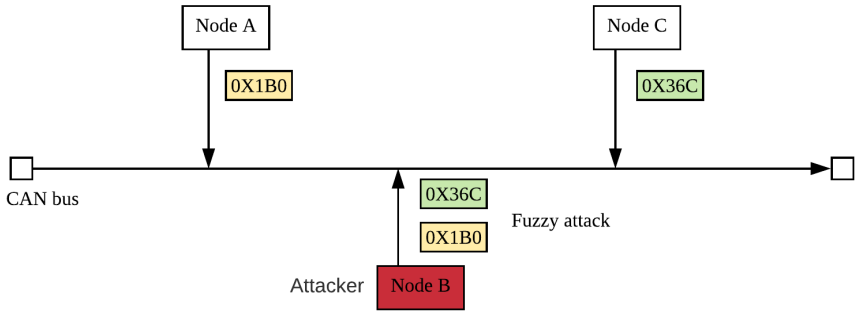

Figure 13. Fuzzy attack.

6. Experiment Procedure and Results

In this section, we explain the method of data normalization and detailed the experimental process, introduced the evaluation method of the model, and finally compared the various models.

6.1. Experiment Procedure

The formatting of the CAN data set used in this paper is illustrated in Figure 14. The timestamp denotes the time elapsed from the start of ignition of the vehicle to the generation of the instruction, and the ID indicates the identity of the instruction. The data length code (DCL) indicates the length of the data field of the instruction. The remainder of the information is the data carried by the instruction. The attack-free data set consists of instructions 10,001 to 2,369,000 (the first 10,000 instructions were removed because they were used to train the SIMATT-SECCU model). We removed the timestamp field in the public dataset, as we only need to ensure that the instructions flow out in order, and the exact time of the outflow is beyond our concern. Nothing is changed in the rest of the public dataset. The DoS attack, fuzzy attack, and impersonation attack data sets employ the first 6,56,500, 591,000, and 995,000 instructions of their original data sets, respectively. All data set values were normalized to a range between 0 and 1 by the following formula:

$$x_{normali} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{22}$$

where x denotes the data that needs to be normalized, x_{min} denotes the minimum value of an attribute in the data set, and x_{max} denotes the maximum value of that attribute in the data set.

```
Timestamp:        0.001928        ID: 0153    000    DLC: 8    00 80 10 ff 00 ff 40 ce
Timestamp:        0.002167        ID: 0260    000    DLC: 8    05 20 00 30 ff 93 5f 35
Timestamp:        0.002402        ID: 02a0    000    DLC: 8    62 00 60 9d db 0c ba 02
Timestamp:        0.002638        ID: 0370    000    DLC: 8    ff 20 00 80 ff 00 00 ec
Timestamp:        0.002882        ID: 0382    000    DLC: 8    40 fe 0f 00 00 00 00 08
Timestamp:        0.003123        ID: 043f    000    DLC: 8    10 50 60 ff 4b 98 09 00
Timestamp:        0.003370        ID: 0440    000    DLC: 8    ff f0 00 00 ff a0 09 00
Timestamp:        0.003617        ID: 0517    000    DLC: 8    00 00 00 00 00 00 00 00
Timestamp:        0.003852        ID: 0545    000    DLC: 8    d8 10 00 8c 32 00 32 00
Timestamp:        0.004095        ID: 059b    000    DLC: 8    00 00 00 00 00 00 00 c4
Timestamp:        0.005486        ID: 02b0    000    DLC: 5    15 00 00 07 30
Timestamp:        0.005713        ID: 0165    000    DLC: 8    0f e8 7f 00 00 00 00 98
```

Figure 14. The formatting of the CAN data set used in this paper.

As shown in Figure 14, in addition to the timestamp field, other fields are entered as features into the SIMATT-SECCU model. An instruction has 10 feature attributes (the data field has 8 feature attributes). We used 50 instructions (500 feature attributes) as an input sample and used the first 10,000 instructions as the training set for the SIMATT-SECCU model. The model was iteratively trained 50 times. Only one layer of 500 SECCU cells is employed. All CAN instruction sequences were further analyzed by the SIMATT-SECCU model, and finally classified using a random forest [33] to complete the attack detecting task.

We used TensorFlow to build the SIMATT-SECCU model and scikit-learn toolkit to implement a random forest, in which all parameters were set to their default values. We use ten-fold cross-validation [34,35] methods to verify the performance of the model, which consists of the following specific steps:

1. We take the first 80% of the instruction sequences in all data sets and divide them into 10 equal parts.
2. Each part is treated as a test set, and the remaining nine parts are used as a training set. A total of 10 test results are obtained.

To evaluate the performance of our SIMATT-SECCU model, we conducted a series of experiments, as follows:

1. The SIMATT model was removed and the SECCU cell was replaced with an LSTM cell or a GRU cell to compare the proposed model with the LSTM and GRU models, respectively. Only the SIMATT model was removed to obtain the SECCU-based model.
2. The SIMATT model was replaced by the HIEATT model, and the SECCU cell was replaced by an LSTM cell or a GRU cell to obtain the HIEATT-LSTM and HIEATT-GRU models, respectively. The HIEATT-SECCU model was obtained by replacing the SIMATT model with the HIEATT model.
3. The SECCU cell was replaced with an LSTM cell or GRU cell to obtain the SIMATT-LSTM and SIMATT-GRU models, respectively.

The training parameter values of the comparative models were set to be the same as those of the SIMATT-SECCU model.

6.2. Introduction to Experimental Evaluation Indicators

Precision, accuracy, recall, F1, the receiver operating characteristic (ROC) curve, and the area under the ROC curve (AUC) are commonly used as performance criteria for evaluating a detecting model. We evaluate the performance of the models using the four terms defined in Table 3.

Table 3. The metrics.

true positives(TP)	Positive class is judged as positive class
false negatives(FN)	Positive class is judged as negative class
false positives(FP)	Negative class is judged as positive class
true negatives(TN)	Negative class is judged as negative class

Precision reflects the proportion of the true positive samples in the positive case determined by the classifier. Precision can be defined as follows:

$$precision = \frac{TP}{TP + FP}. \tag{23}$$

Accuracy reflects the ability of the classifier to determine the entire sample. Accuracy can be defined as follows:

$$accuracy = \frac{TP + TN}{TP + FP + FN + TN}. \tag{24}$$

Recall rate reflects the proportion of positive cases that are correctly judged to the total positive case. In addition, recall can be defined as follows:

$$recall = \frac{TP}{TP + FN}. \tag{25}$$

F1 reflects the harmonic mean of the accuracy and recall rate. F1 can be defined as follows:

$$F1 = \frac{precision \times recall}{precision + recall} \times 2. \tag{26}$$

The true positive rate (TPR) is defined as the ratio of true positives to all positive samples. Its formula is the same as that of the recall rate:

$$TPR = \frac{TP}{TP + FN}. \tag{27}$$

The false positive rate (FPR) is defined as the ratio of false positives to all negative samples. Its formula is as follows:

$$FPR = \frac{FP}{FP + TN}. \tag{28}$$

The ROC curve is a plot of the FPR on the X axis and the TPR on the Y axis. In addition, the AUC value is the area under the ROC curve. Obviously, a larger AUC indicates better classification results.

6.3. Model Performance Analysis

The ten-fold cross-validation accuracy results for the comparison models are shown in Table 4. The results show that SIMATT-SECCU has the highest accuracy, followed by SIMATT-LSTM. In addition, the models that include SIMATT have a high degree of accuracy. The models that include HIEATT and the SECCU cell have different levels of performance degradations. This may be because the HIEATT model is too complex for relatively simple CAN protocol instructions, and this degrades its performance.

Table 4. The ten-fold cross-validation accuracy results for the comparison models.

Model	1	2	3	4	5	6	7	8	9	10	Mean
SECCU	0.916	0.900	0.876	0.898	0.864	0.684	0.587	0.521	0.540	0.593	0.738
LSTM	0.964	0.922	0.909	0.922	0.921	0.817	0.711	0.787	0.752	0.729	0.843
GRU	0.960	0.922	0.904	0.926	0.903	0.774	0.675	0.702	0.673	0.686	0.813
HIEATT-SECCU	0.929	0.887	0.867	0.883	0.863	0.738	0.648	0.686	0.632	0.656	0.779
HIEATT-LSTM	0.958	0.924	0.906	0.925	0.903	0.796	0.677	0.723	0.664	0.687	0.816
HIEATT-GRU	0.946	0.909	0.901	0.914	0.899	0.763	0.626	0.678	0.628	0.635	0.790
SIMATT-SECCU	**1.000**	**1.000**	**0.993**	**1.000**	**1.000**	**1.000**	**0.993**	**1.000**	**1.000**	**0.993**	**0.997**
SIMATT-LSTM	1.000	1.000	1.000	0.993	1.000	0.993	0.958	1.000	1.000	0.993	0.993
SIMATT-GRU	1.000	1.000	0.979	0.993	0.993	0.986	0.938	1.000	0.993	0.972	0.985

For the detecting task, the recall rate of the model is considered more important. Table 5 shows the recall rate of each model. The SIMATT-SECCU recall rate is only greater than that of SIMATT-LSTM by a value of 0.003. However, the CAN is usually able to generate 685,000 instructions in 300 s, which can reduce false positives by more than 40 times.

Table 5. The ten-fold cross-validation recall results for the comparison models.

Model	1	2	3	4	5	6	7	8	9	10	Mean
SECCU	0.840	0.798	0.760	0.803	0.740	0.663	0.858	0.852	0.823	0.872	0.801
LSTM	0.936	0.853	0.820	0.851	0.840	0.863	0.915	0.937	0.938	0.943	0.890
GRU	0.936	0.846	0.821	0.853	0.815	0.834	0.927	0.928	0.915	0.943	0.882
HIEATT-SECCU	0.862	0.787	0.745	0.799	0.751	0.765	0.877	0.884	0.869	0.898	0.824
HIEATT-LSTM	0.921	0.852	0.809	0.857	0.831	0.861	0.928	0.917	0.915	0.925	0.882
HIEATT-GRU	0.892	0.834	0.819	0.832	0.804	0.826	0.905	0.906	0.895	0.909	0.862
SIMATT-SECCU	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
SIMATT-LSTM	1.000	1.000	1.000	1.000	1.000	0.986	0.986	1.000	1.000	1.000	0.997
SIMATT-GRU	1.000	1.000	0.986	0.973	0.986	1.000	1.000	1.000	0.986	1.000	0.993

The ROC curve (Figure 15) enables us to intuitively evaluate the performance of each model. As the figure shows, the SIMATT model substantially improves the performance of the models (the three curves closely follow each other). In contrast, the HIEATT model improves the performance of the SECCU cell only: the performance of the LSTM and GRU cells declines when HIEATT is used.

We used the first 80% of all datasets as the training set and the last 20% as the test set to evaluate the performance of each model. The experimental results are shown in Figure 16. The impersonation attack is difficult to detect for most models because it mimics a normal ECU, sending packets without generating much change in the overall instruction sequence. In contrast, the detecting of an impersonation attack when the SIMATT model is employed is no longer a challenge. Most models also have some detection errors on the attack-free dataset. This may be because the sudden braking of the vehicle during driving can caused the features of the attack-free dataset to change and affect the detecting results. Each model performs well on the DoS and fuzzy attack datasets because the data these data sets have obvious features. For example, in the DoS attack data set, the same data packet is periodically sent and is easily detected. In addition, although the performance of the SECCU cell is slightly worse, it has higher plasticity. When HIEATT is added to the models, the performance of LSTM and GRU decreased, whereas the performance of SECCU increased. Moreover, when SIMATT is added to the models, the performance of SIMATT-SECCU exceeds those of all other models.

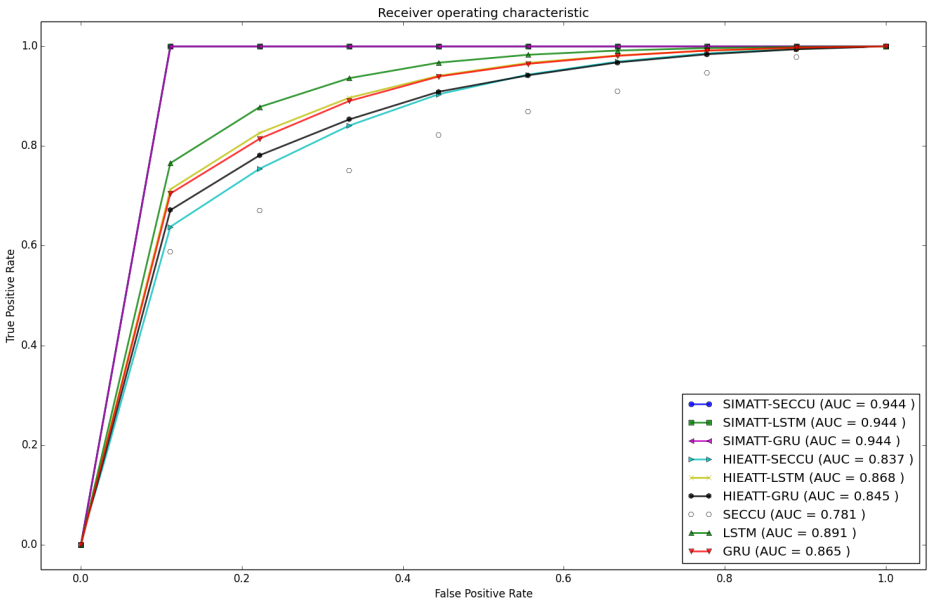

Figure 15. Receiver operating characteristic (ROC) curve for the comparison models.

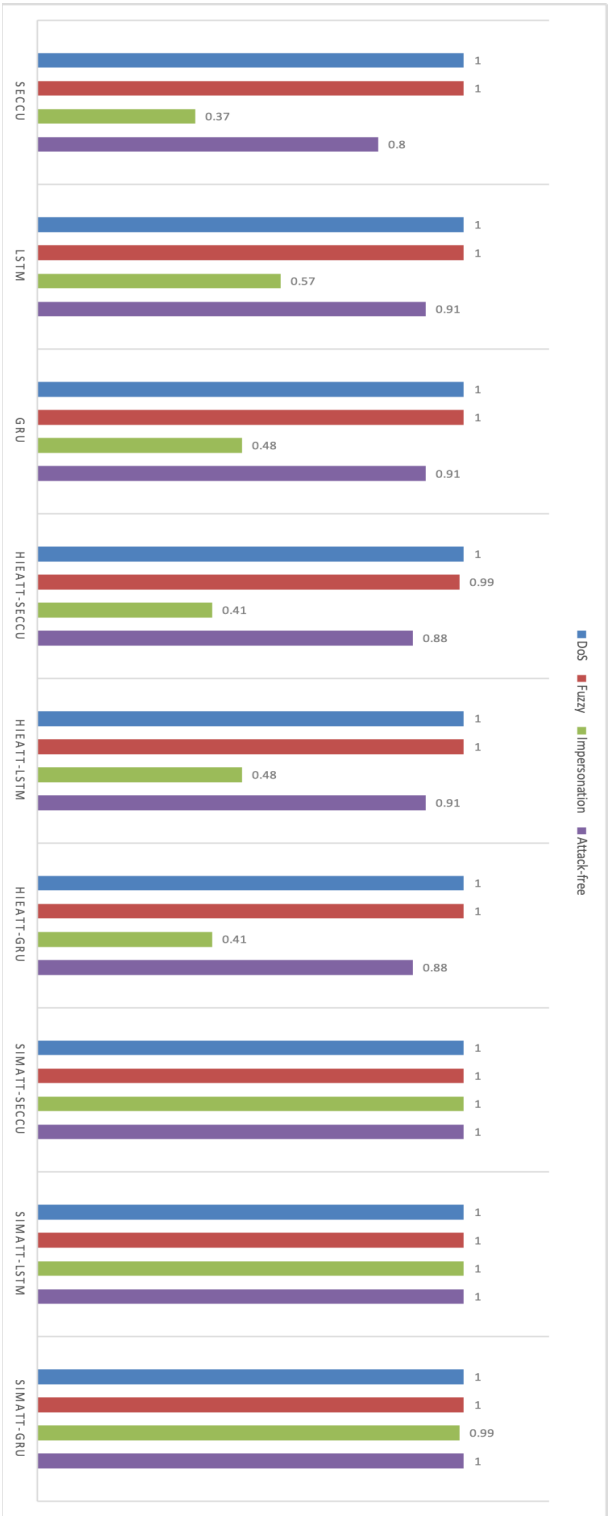

Figure 16. The recall rate for four datasets of the comparison models.

Based on the above experimental results, it can be concluded that the SIMATT model can help SECCU-, LSTM-, and GRU-based RNNs substantially improve anomaly detecting. Moreover, the SIMATT-SECCU model obtains state-of-the-art detection performance.

To further demonstrate the performance advantages of SIMATT-SECCU, we compare it with the GIDS [16] and HE-RNN [17] models. In the comparison of GIDS, we use the data set provided in [16] (which added two attack data sets related to RPM attack and GEAR attack) to check the ability of

SIMATT-SECCU in detecting multiple attacks. Detailed attack steps of GIDS can be found in [16]. We take the first 1,000,000 instructions in the data set and the first 80% of all datasets as the training set and the rest as the test set to evaluate the performance. The recall of GIDS and SIMATT-SECCU are shown in Figure 17. The detection performance of SIMATT-SECCU for three attacks is better than that of GIDS, and particularly the detection capability of GEAR attack is significantly improved. Whereas, SIMATT-SECCU does not show comparable performance in detecting fuzzy attacks.

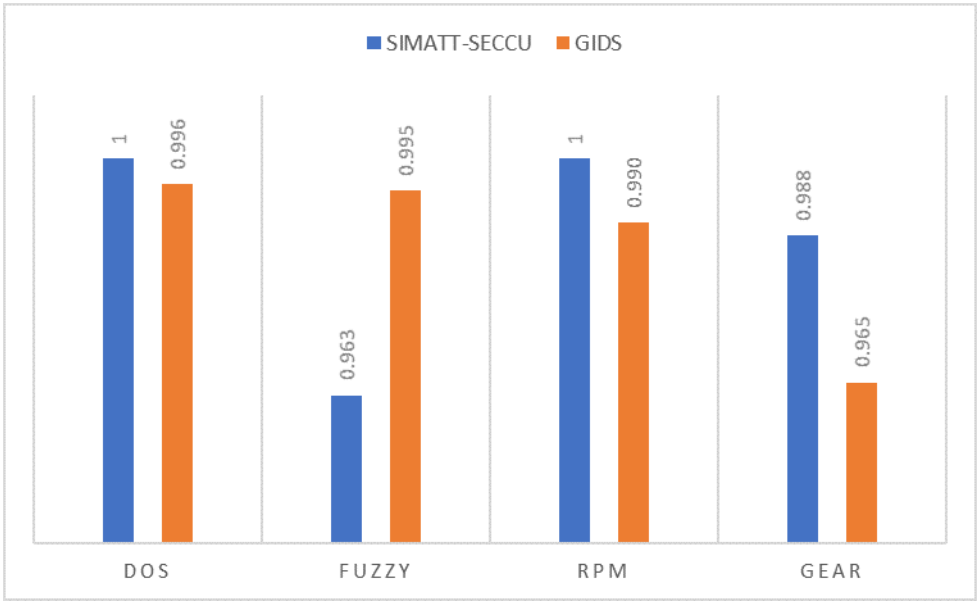

Figure 17. The recall rate for four datasets of the SIMATT-SECCU and generative adversarial networks (GAN)-based intrusion detection system (GIDS).

HE-RNN [17] uses the same data set of this paper to directly compare the detection capabilities of the model. From Tables 6 and 7, we can see that the SIMATT-SECCU model is more powerful for detecting both fuzzy and DoS attacks.

Table 6. Heuristics and recurrent neural networks (RNNs) model (HE-RNN) detecting results.

Attack	Precision	Recall	F1
DoS	0.9930	0.9927	0.9929
Fuzzy	0.9901	0.9935	0.9918

Table 7. Heuristics and recurrent neural networks (RNNs) model (HE-RNN) detecting results.

Attack	Precision	Recall	F1
DoS	1.0000	1.0000	1.0000
Fuzzy	1.0000	1.0000	1.0000

To compare the computational costs of each models, we take 20,000 instructions generated in about 10 s for time testing. Our configuration is as follows: CPU: Intel Core i7-6700 3.40 GHZ, RAM: 8 G, operating system: Windows 7 without GPU computing components. We built these programs using python 3.6 and TensorFlow 1.12.

Figure 18 shows the detecting time consumption of the 11 models. It can be seen that the SIMATT optimally adapts to the SECCU because SIMATT-SECCU only has 0.003 s more detecting time than SECCU, and HIEATT-X about 0.02 s slower than SIMATT-X (X represents SECCU, LSTM, or GRU, respectively). Most time consumptions of the GIDS detecting are spent on one-hot-vector

encoding conversion. The arbitration ID in each CAN instruction needs to be converted into three one-hot-vectors, which not only idles away time but also increases space overhead. HE-RNN leads to excessive detection time overhead due to the complex neural network structure. The RNN network of HE-RNN has four layers, with a total of 1280 computing cells, and contrarily we only have one layer of 500 computing cells. Moreover, in HE-RNN, we only calculate the time spent by data passing through RNN, and the heuristic algorithm for measuring CAN data features is not covered yet, meaning that HE-RNN actually needs more calculations.

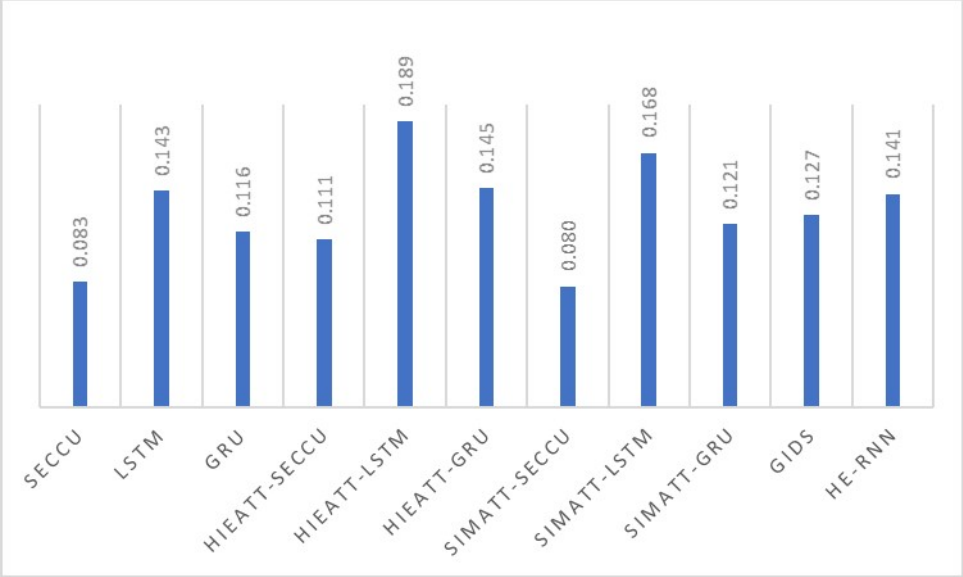

Figure 18. The time required for each model to detect 20,000 instructions.

Putting the above analysis altogether, we can argue that the proposed SIMATT-SECCU model can significantly reduce the calculation cost as well as ensure the detection accuracy.

7. Conclusions

Internet of vehicles allows intelligent automobiles to interchange messages with other cars, traffic management departments, and data analysis companies about vehicle identification, accident detection, and danger warnings. The implementation of these features requires Internet of Things system support. smart cars are generally equipped with many (hundreds or even thousands of) sensors and microcomputers so that drivers gain a more information about travel. The connection between the in-vehicle network and the Internet can be leveraged by the attackers in a malicious manner and thus increases the number of ways the in-vehicle network can now be targeted. Protecting increasingly intelligent vehicle systems becomes more difficult, especially because a network of many different devices makes the system more threatening than ever before.

In this paper, the SIMATT-SECCU framework was introduced that can be used to detect instruction sequence anomalies on in-vehicle network. This IDS monitors vehicle status in real time and does not require expert knowledge of the CAN protocol. For this framework, we propose a high-plasticity recurrent neural network called the security control unit (SECCU) that can achieve higher performance through the addition of the attention model. At the same time, we also propose the simplified attention (SIMATT) model, which has low computational cost and can significantly improve the performance of LSTM, GRU, and SECCU models in vehicle network anomaly detecting tasks.

Our contributions include the following.

1. To suit the actual situation of an in-vehicle network, we present a new recurrent neural network (RNN) model named SECCU.

2. We propose the simplified attention (SIMATT) model that can be computed using the computing power of vehicle electronics and improve detection accuracy.
3. The experimental results demonstrate that our proposed SIMATT-SECCU symmetry framework has the best performance in comparison with many classic models.
4. Our detecting model is fully data-driven and does not require any domain knowledge about the CAN protocol.
5. Our proposed model focuses on a lightweight anomaly detecting for in-vehicle network based on machine learning, which solves the problem of insufficient computing power of networked vehicle electronic devices when applying machine learning algorithms.

More work should be done to improve the performance of the in-vehicle network IDS in more complex situations. One significant direction of future research is to determine whether the detection accuracy of the model can be further improved and the false positive rate reduced. Another interesting direction would be to construct a lightweight heuristic that matches the computational requirements of real-time detecting. Furthermore, when IDS detects that the vehicle is behaving abnormally, how to launch a reasonable response is another challenge. All these aspects are worth further study.

Author Contributions: Conceptualization, J.X.; methodology, J.X.; writing—original draft preparation, J.X.; writing—review and editing, X.L. and H.W.; project administration, X.L. and H.W.; funding acquisition, X.L.

Funding: The paper was supported by the National Cryptography Development Fund (Grant No. MMJJ20180106) and the National Natural Science Foundation of China (Grant Nos. 61572192, 61971192).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Woo, S.; Jo, H.J.; Lee, D.H. A practical wireless attack on the connected car and security protocol for in-vehicle CAN. *IEEE Trans. Intell. Transp. Syst.* **2015**, *16*, 993–1006. [CrossRef]
2. Foster, I.; Prudhomme, A.; Koscher, K.; Savage, S. Fast and vulnerable: A story of telematic failures. In Proceedings of the USENIX Workshop on Offensive Technologies, Washington, DC, USA, 10–11 August 2015.
3. Golde, N.; Redon, K.; Borgaonkar, R. Weaponizing femtocells: The effect of rogue devices on mobile telecommunications. In Proceedings of the NDSS Symposium, San Diego, CA, USA, 5–8 Feburary 2012.
4. Kim, J.H.; Seo, S.-H.; Hai, N.T.; Cheon, B.M.; Lee, Y.S.; Jeon, J.W. Gateway framework for in-vehicle networks based on CAN, FlexRay, and Ethernet. *IEEE Trans. Veh. Technol.* **2015**, *64*, 4472–4486. [CrossRef]
5. Park, T.J.; Han, C.S.; Lee, S.H. Development of the electronic control unit for the rack-actuating steer-by-wire using the hardware-in-the-loop simulation system. *Mechatronics* **2005**, *15*, 899–918, . [CrossRef]
6. Tuohy, S.; Glavin, M.; Ciarán, H. Intra-vehicle networks: A review. *IEEE Trans. Intell. Transp. Syst.* **2015**, *16*, 534–545. [CrossRef]
7. Biswas, S.; Tatchikou, R.; Dion, F. Vehicle-to-vehicle wireless communication protocols for enhancing highway traffic safety. *IEEE Commun. Mag.*, **2006**, *44*, 74–82, [CrossRef]
8. Farsi, M.; Ratcliff, K.; Barbosa, M. An overview of controller area network. *Comp. Control Eng. J.* **1999**, *10*, 113–120. [CrossRef]
9. Li, X.; Yu, Y.; Sun, G.; Guannan, S.; Kefei, C. Connected vehicles' security from the perspective of the In-vehicle network. *IEEE Netw.* **2018**, *32*, 58–63. [CrossRef]
10. Tang, T.; Shi, W.; Shang, H.; Wang, Y. A new car-following model with consideration of inter-vehicle communication. *Nonlinear Dyn.* **2014**, *76*, 2017–2023. [CrossRef]
11. Groza, B.; Murvay, S. Efficient protocols for secure broadcast in controller area networks. *IEEE Trans. Ind. Inform.* **2013**, *9*, 2034–2042. [CrossRef]
12. Woo, S.; Jo, H.J.; Kim, I.S.; Lee, D.H. A practical security architecture for in-vehicle CAN-FD. *IEEE Trans. Intell. Transp. Syst.* **2016**, *17*, 2248–2261. [CrossRef]
13. Muter, M.; Asaj, N. Entropy-based anomaly detection for in- vehicle networks. In Proceedings of the 2011 IEEE Intelligent Vehicles Symposium (IV), Baden-Baden, Germany, 5–9 June 2011; pp. 1110–1115.
14. Lan, H.M.; Kwak, B.I.; Kim, H.K. Anomaly intrusion detection method for vehicular networks based on survival analysis. *Veh. Commun.* **2018**, *14*, 52–63.

15. Song, H.M.; Kim, H.K. Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network. In Proceedings of the International Conference on Information Networking (ICOIN), Kota Kinabalu, Malaysia, 13–15 January 2016.
16. Seo, E.; Song, H.M.; Kim, H.K. GIDS: GAN based intrusion detection system for in-vehicle network. In Proceedings of the 16th Annual Conference on Privacy, Security and Trust (PST), Belfast, UK, 28–30 August 2018.
17. Tariq, S.; Lee, S.; Kim, H.K.; Woo, S.S. Detecting In-vehicle CAN message attacks using heuristics and RNNs. In Proceedings of the International Workshop on Information and Operational Technology Security Systems, Heraklion, Greece, 13 September 2018.
18. Larson, U.E.; Nilsson, D.K.; Jonsson, E. An approach to specification-based attack detection for in-vehicle networks. In Proceedings of the IEEE Intelligent Vehicles Symposium, Eindhoven, The Netherlands, 4–6 June 2008; pp. 220–225.
19. Wang, C.; Zhao, Z.; Gong, L.; Zhu, l.; Cheng, X. A distributed anomaly detection system for in-vehicle network using HTM. *IEEE Access* **2018**, *6*, 9091–9098. [CrossRef]
20. Hu, W.; Liao, Y.; Vemuri, V.R. Robust anomaly detection using support vector machines. In Proceedings of the International Conference on Machine Learning, Washington, DC, USA, 21–24 August 2003; pp. 282–289.
21. Li, H.; Wang, Y.; Qin, H.; Xinkai, W. Investigating the effects of attack detection for in-vehicle networks based on clock drift of ECUs. *IEEE Access* **2018**, *6*, 49375–49384.
22. Xiao, J.; Wu, H.; Li, X. Robust and Self-Evolving IDS for In-Vehicle Network by Enabling Spatiotemporal Information. In Proceedings of the IEEE 21st International Conference on High Performance Computing and Communications, Zhangjiajie, China, 10–12 August 2019.
23. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to sequence learning with neural networks. *arXiv* **2014**, arXiv:1409.3215.
24. Cho, K.; van Merrienboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv* **2014**, arXiv:1406.1078.
25. Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Gated feedback recurrent neural networks. *arXiv* **2015**, arXiv:1502.02367.
26. Jozefowicz, R.; Zaremba, W.; Sutskever, I. An empirical exploration of recurrent network architectures. In Proceedings of the International Conference on Machine Learning, Lille, France, 6–11 July 2015.
27. Greff, K.; Srivastava, R.K.; Koutník, J.; Steunebrink, B.R.; Schmidhuber, J. LSTM: A search space odyssey. *IEEE Trans. Neural Netw. Learning Syst.* **2017**, *28*, 2222–2232. [CrossRef]
28. Parikh, A.P.; Täckström, O.; Das, D.; Uszkoreit, J. A decomposable attention model for natural language inference. *arXiv* **2016**, arXiv:1606.01933.
29. Pappas, N.; Popescu-Belis, A. Multilingual hierarchical attention networks for document classification. *arXiv* **2017**, arXiv:1707.00896.
30. Sukhbaatar, S.; Szlam, A.; Weston, J.; Fergus, R. End-to-end memory networks. *arXiv* **2015**, arXiv:1503.08895.
31. Lee, H.; Jeong, S.H.; Kim, H.K. OTIDS: A novel intrusion detection system for in-vehicle network by using remote frame. In Proceedings of the 15th Annual Conference on Privacy, Security and Trust, PST, Calgary, AB, Canada, 27–29 August 2017.
32. Cho, K.-T.; Shin, K.G. Fingerprinting electronic control units for vehicle intrusion detection. In Proceedings of the USENIX Security Symposium, Austin, TX, USA, 10–12 August 2016.
33. Breiman, L. Random Forests. *Mach. Learning* **2001**, *45*, 5–32. [CrossRef]
34. Zhang, H.; Yang, S.; Guo, L.; Zhao, Y.; Shao, F.; Chen, F. Comparisons of isomir patterns and classification performance using the rank-based manova and 10-fold cross-validation. *Gene* **2017**, *569*, 21–26. [CrossRef] [PubMed]
35. Meijer, R.J.; Goeman, J.J. Efficient approximate k-fold and leave-one-out cross-validation for ridge regression. *Biom. J.* **2013**, *55*, 141–155. [CrossRef] [PubMed]