Threat Analysis for Automotive CAN Networks: A GAN Model-Based Intrusion Detection Technique

Guoqi Xie, *Senior Member, IEEE*, Laurence T. Yang, *Fellow, IEEE*, Yuanda Yang, Haibo Luo, Renfa Li, *Senior Member, IEEE*, and Mamoun Alazab, *Senior Member, IEEE*

***Abstract*— With the rapid development of Internet of vehicles, connected vehicles, autonomous vehicles, and autonomous driving technologies, automotive Controller Area Networks (CAN) have suffered from numerous security threats. Deep learning models are the current mainstream intrusion detection techniques for threat analysis, and the state-of-the-art intrusion detection technique introduces the Generative Adversarial Networks (GAN) model to generate usable attacked samples to supplement the training samples, but it exists the limitations of rough CAN message block construction and fails to detect the data tampering threat. Based on the CAN communication matrix defined by the automotive Original Equipment Manufacturer (OEM) for a vehicle model, we propose an enhanced deep learning GAN model with elaborate CAN message blocks and the enhanced GAN discriminator. The elaborate CAN message blocks in the training samples can precisely reflect the real generated CAN message blocks in the detection phase. The GAN discriminator can detect whether each message has suffered from the data tampering threat. Experimental results illustrate that the enhanced deep learning GAN model has higher detection accuracy, recall, and F1 scores than the state-of-the-art deep learning GAN model under various attacks and threats.**

***Index Terms*— Automotive networks, deep learning, intrusion detection, threat analysis.**

I. Introduction

A. Background

CONTROLLER Area Network (CAN) [1] is the most widely used bus in today's automotive networks due to its simplicity, low cost, high stability, and long-term applicability for more than 30 years. Unfortunately, the lacks of any security protection mechanism make CAN extremely vulnerable to security threats [2]. With the rapid development of Internet of vehicles, connected vehicles, autonomous vehicles, and autonomous driving technologies [3], automotive CAN networks have suffered from numerous security threats [4], [5]. Increasing security threats affect the normal message transmission in automotive CAN networks.

Distributed automotive embedded systems built on automotive CAN networks control the normal operation of automotive applications. Automotive embedded systems are typical safety-critical systems, and any abnormal message transmission due to security threats may cause malfunctioning behavior for safety-critical applications (e.g., brake-by-wire, adaptive cruise control [6], [7]), resulting in serious hazardous events and even fatal casualties [8]. Therefore, it is quite necessary to timely detect abnormal message transmission caused by security threats in automotive CAN networks [9]. Through intrusion detection techniques, automotive embedded systems are prompted to isolate the attacked network or enter the safe mode, thereby reducing the security threat of the vehicle when it is driving.

B. Motivation

Deep learning models are the current mainstream intrusion detection techniques for threat analysis [10]. The state-of-the-art intrusion detection technique [11] introduces the deep learning model called Generative Adversarial Networks (GAN) [12] to generate usable attacked samples to supplement the training samples, thereby improving the intrusion detection accuracy. GAN is one of the most promising unsupervised learning methods on complex distributions in recent years. The essence of GAN is below [12]: through the dynamic minimum and maximum game training between the GAN generator and the GAN discriminator, the GAN generator can generate the most realistic adversarial samples, while the GAN discriminator is not enough to distinguish between the real and the adversarial samples; finally, the Nash balance between the GAN generator and the GAN discriminator is achieved. Although [11] solves the problem of insufficient attacked samples, it still has the following limitations.

(1) In the preprocessing phase, 64 consecutive CAN message jobs are constructed to one CAN message block [11]; however, the constructed CAN message block in the preprocessing phase cannot accurately reflect the actual CAN message block in the detection phase, such that the CAN message block is rough (see more details in Section III.C).

Manuscript received July 28, 2020; revised November 22, 2020; accepted January 19, 2021. Date of publication February 10, 2021; date of current version July 12, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 61932010, Grant 61972139, Grant 61672217, and Grant 61702172; in part by the Open Research Project of the Electronic Information and Control of Fujian University Engineering Research Center, Minjiang University, China, under Grant MJXY-KF-EIC1902; and in part by the Fundamental Research Funds for the Central Universities, Hunan University, China. The Associate Editor for this article was A. Jolfaei. *(Corresponding author: Mamoun Alazab.)*

Guoqi Xie, Yuanda Yang, and Renfa Li are with the Key Laboratory for Embedded and Network Computing of Hunan Province, College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China (e-mail: xgqman@hnu.edu.cn; yyd@hnu.edu.cn; lirenfa@hnu.edu.cn).

Laurence T. Yang is with the Department of Computer Science, St. Francis Xavier University, Antigonish, NS B2G 2W5, Canada (e-mail: ltyang@ieee.org).

Haibo Luo is with the College of Computer and Control Engineering, Minjiang University, Fuzhou 350108, China (e-mail: robhappy@mju.edu.cn).

Mamoun Alazab is with the College of Engineering, IT and Environment, Charles Darwin University, Casuarina, NT 0810, Australia (e-mail: alazab.m@ieee.org).

Digital Object Identifier 10.1109/TITS.2021.3055351

(2) The GAN discriminator in [11] fails to detect the data tampering threat; however, attackers prefer to tamper with the messages to achieve serious destructiveness (see more details in Section IV.B).

C. Contributions

To solve the above limitations, this paper introduces the CAN communication matrix [13] defined by the automotive Original Equipment Manufacturer (OEM) for a vehicle model. The CAN communication matrix contains a wealth of important information, including message identifier, message classification, message period, etc. On the basis of the CAN communication matrix, we propose an enhanced deep learning GAN model with the elaborate CAN message block and the enhanced GAN discriminator. Compared with [11], this paper makes the following innovative contributions.

(1) Based on the message sending type provided in the CAN communication matrix, we let all CAN message jobs in a CAN message block be from the same CAN message (i.e., all CAN message jobs have the same message identifier). Through this treatment, the elaborate CAN message blocks in the training samples can precisely reflect the real generated CAN message blocks during the detection phase.

(2) As the CAN communication matrix provides the required range of the minimum value and the maximum value for each signal in a message (a message contains multiple signals), we add the CAN communication matrix into the GAN discriminator to train an enhanced GAN discriminator. Through the enhanced GAN discriminator, we can discriminate whether each message has suffered from the data tampering threat, thereby increasing the intrusion detection precision.

(3) Experimental results illustrate that the enhanced deep learning GAN model has higher detection precision and recall than the state-of-the-art deep learning GAN model under various threats and attacks.

II. Related Work

Wu *et al*. [2] provided a systematic survey on intrusion detection techniques in automotive CAN networks published from 2012 - 2018. Some intrusion detection techniques, such as fingerprint [14], [15], parameters monitoring [16], and information theory [17], [18], have been proposed for automotive CAN networks. The aforementioned techniques have individual limitations. For instance, the fingerprint technique usually targets for the masquerade attack or the bus off attack [14], [15]; the parameters monitoring technique usually targets for the injection attack [16]; and the information theory technique usually targets for the injection attack or the Denial Of Service (DoS) attack [17], [18].

Deep learning models are the current mainstream intrusion detection techniques. For example, some advanced deep learning models, such as Long Short-Term Memory (LSTM), Recurrent Neural network (RNN) [19], Deep Neural Network (DNN) [20], and Deep Convolutional Neural Network (DCNN) [21] have been used for intrusion detection techniques in automotive CAN networks. Although deep learning models can effectively detect malicious software, malicious code, malicious behavior, and other security threats, there is a common challenge: the number of the attacked samples during the training phase is insufficient and far fewer than the number of the normal samples. The imbalance between the attacked and the normal samples prevents these models from properly implement the precise intrusion detection technique.

In 2018, Seo *et al*. [22] for the first time introduced the GAN model for the intrusion detection technique in automotive CAN networks; this model is a hybrid model that includes the DNN model and the GAN model. Unfortunately, the GAN model has a small contribution to the detection precision of the hybrid model, and the main contribution is still the DNN model. To solve the problem of insufficient attacked samples, the state-of-the-art intrusion detection technique in [11] adopts a pure GAN model in both the training and detection phases; Experiments revealed that the pure GAN model in [11] has higher detection precision than the hybrid DNN and GAN model in [22]. However, the constructed CAN message block is rough and the GAN model fails to detect the data tampering threat in [11] as pointed earlier.

III. Preprocessing

Fig. 1 shows the preprocessing phase for the intrusion detection technique in this study.

A. CAN Communication Matrix and Message Classification

For a vehicle model, the CAN Communication matrix is stored in the DBC (Data Base CAN [23]) file, which is provided by the automotive OEM. Fig. 2 shows a simple example of the CAN communication matrix, which contains much important information, such as message identifier (i.e., *Msg_ID*), message classification (i.e., *Msg_Send_Type*), message period (i.e., M*sg_Cycle_Time*), etc. Note that different OEMs may have different notations, and this study uses the notations illustrated in Fig. 2. Each Electronic Controller Unit (ECU) node in automotive CAN networks must follow the CAN communication matrix to complete the interaction and information sharing. Once the CAN communication matrix is available, developers must follow the CAN Communication matrix to develop automotive CAN networks and ECUs.

In general, there are several message types (i.e., *Msg_Type* in the CAN communication matrix) in automotive CAN networks, including normal communication messages (*Normal* for short in the CAN communication matrix), network management messages (*NM* for short in the CAN communication matrix), and diagnostic messages (*Diag* for short in the CAN communication matrix).

There are generally multiple message classifications in automotive CAN networks according to the message sending type (i.e., *Msg_Send_Type*). In general, these message classifications are as follows: 1) periodic; 2) on event; 3) if active; 4) periodic and on event; and 5) periodic and if active. To facilitate the development and management of engineers, the message classification can be derived through the message's ID value as the CAN communication matrix defines the concrete ID range for each message classification.

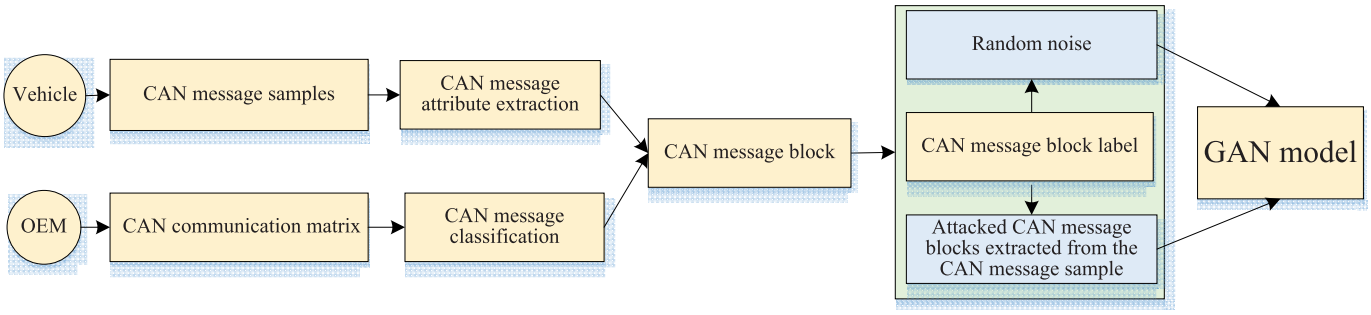


Fig. 1. Preprocessing phase in this paper.

Msg_ Name	Msg_T ype	Msg_I D	Msg_Sen d_Type	Msg_Cyc le_Time	Msg_Cycle_ Time_Fast	Msg_Nr_Of _Reption	Msg_Del ay_Time	Msg_L ength	Signal_N ame	Byte_Or der	Start _Byte	Start _Bit	Signal_S end_Type	Bit_L ength	Signal_M in_Value	Signal_M ax_Value
D_2F3	Normal	0x2F3	Event	0	0	0	0	4								
D_400	Normal	0x400	Cycle	100	0	0	0	8								
									D_S_Name1	Intel	0	0	Cycle	1	0x0	0x1
									D_S_Name2	Motorola	0	1	Cycle	2	0x1	0x2
									D_S_Name3	Motorola	0	3	Event	5	0x2	0x3
									D_S_Name4	Motorola	1	8	Cycle	8	0x3	0x4
									D_S_Name5	Motorola	3	24	Event	16	0x4	0x5
									D_S_Name6	Motorola	7	56	Cycle	32	0x5	0x6

Fig. 2. Simple example of the CAN communication matrix.

TABLE I
EXAMPLE OF ID RANGES OF FIVE MESSAGE CLASSIFICATIONS IN A VEHICLE MODEL

Message classification	ID (min)	D (max)
On event	0x000	0x0FF
Periodic and on event	0x100	0x1FF
If active	0x200	0x2FF
Periodic and if active	0x300	0x3FF
Periodic	0x400	0x4FF

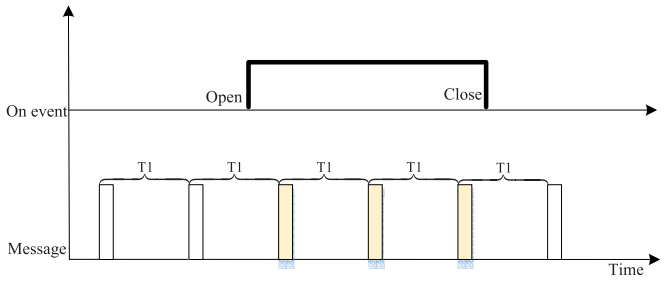


Fig. 3. *Cycle* message is released repeatedly according to a fixed period when the system starts.

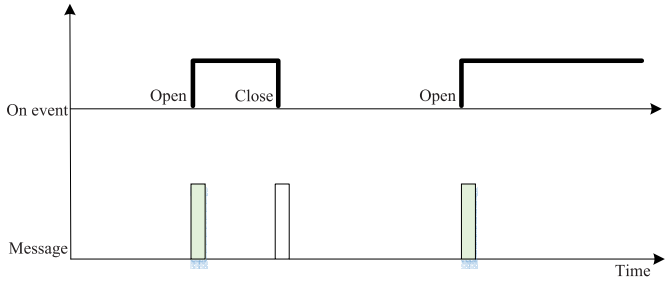


Fig. 4. *Event* message is released once when it is triggered by an event.

For instance, Table I shows an example of the ID ranges of five message classifications in a vehicle model.

In the following, we explain the release mode of the following message classifications: 1) periodic; 2) on event; 3) if active; 4) periodic and on event; and 5) periodic and if active.

*1) Periodic (*Cycle *in the CAN Communication Matrix):* The *Cycle* message is released repeatedly according to a fixed period (i.e., *Msg_Cycle_Time*) when the system starts. Fig. 3 shows that when the event is opened, it triggers the change of the *Cycle* message's status signal, such that the content of the *Cycle* message changes (rectangle marked with orange color); when the event is closed, the content of the *Cycle* message recovers. A typical *Cycle* message is the *ECU status message*, which reports the status of an ECU.

*2) On Event Message (*Event *in the CAN Communication Matrix):* The message is released once when it is triggered by an event. Fig. 4 shows that when the event is opened, it triggers the *Event* message release with valid content (rectangle marked with green color); when the event is closed, the content of the *Event* message is blank and the message release stops.

*3) If Active Message (*IfActive *in the CAN Communication Matrix):* The *IfActive* message is released repeatedly according to a fixed period (i.e., *Msg_Cycle_Time_Fast* in the CAN communication matrix) only in its active time interval. Fig. 5 shows that when the active switch is opened, it triggers the *IfActive* message to release repeatedly with valid content (rectangle marked with orange color); when the active switch is closed, the *IfActive* message release stops.

*4) Periodic and on Event Message (*CE *in the CAN Communication Matrix):* Fig. 6 shows that the *CE* message is the

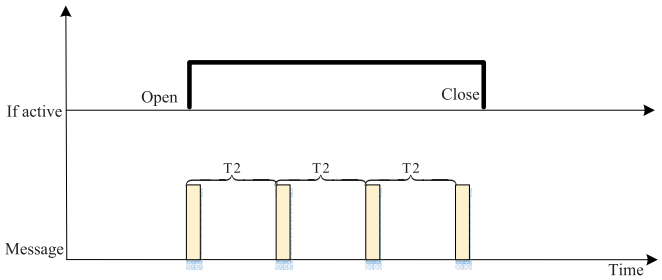


Fig. 5. *If Active* message is released repeatedly according to a fixed period only in its active time interval.

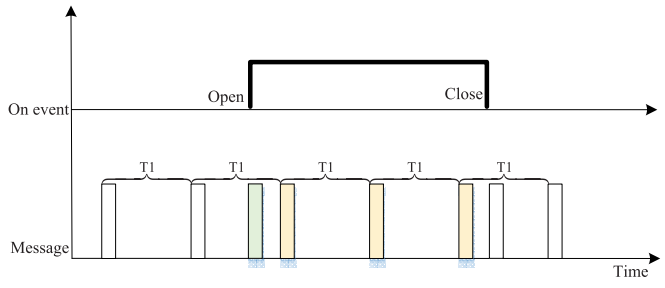


Fig. 6. *CE* message is the combination of periodic message and on event message.

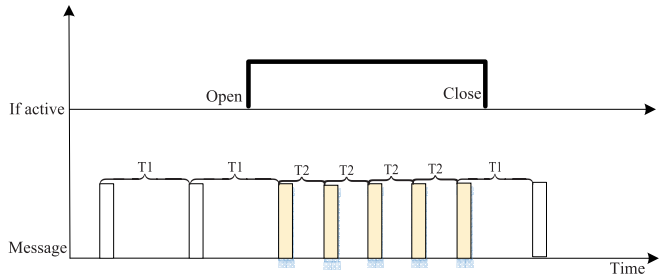


Fig. 7. *CA* message is the combination of periodic message and if active message.

combination of *Cycle* message (Fig. 3) and *Event* message (Fig. 4).

*5) Periodic and If Active Message (*CA *in the CAN Communication Matrix):* Fig. 7 shows that the *CA* message is the combination of the *CE* message (Fig. 3) and the *IfActive* message (Fig. 5). Note that the period of the *CA* message in the active interval time is a fast period (T2), which is less than the original period (T1).

The state-of-the-art work [11] does not classify messages during the preprocessing phase. In this case, a big misjudgment is brought to the GAN discriminator. Let us take the *CA* message in Fig. 7 as an example to explain the misjudgment. For the released multiple jobs of the same message Fig. 7, the GAN discriminator may have two distinct discrimination results: 1) if this message is a *CA* message, it may be discriminated as a normal message; 2) if this message is a *CE* message, it may be discriminated as an attacked message; The reason is that a *CA* message (Fig. 7) has more jobs than a *CE* message (Fig. 6) in the equal interval under the same period. From the perspective of a *CE* message, Fig. 7 was injected by an attacker with new attacked messages; that is, the message jobs shown in Fig. 7 have suffered from the injection attack. Therefore, to reduce the misjudgment of the GAN discriminator, this paper classifies messages based on the values enumerated by *Msg_Send_Type* in the CAN communication matrix.

Timestamp	ID	DLC	Content
147819837.0001,	0440,	8,	ff,00,00,00,ff,d0,08,00,
147819838.0012,	0441,	8,	ff,0B,10,22,ff,d2,3C,2A,
.	.	.	.
.	.	.	.
.	.	.	.
147819841.0012,	0439,	8,	ff,22,3C,4D,2f,32,D4,16,
.	.	.	.
.	.	.	.
.	.	.	.

Fig. 8. Simple message attribute extraction.

B. CAN Message Samples and Attribute Extraction

The CAN message samples are collected from the automotive CAN network of a real vehicle after various attacks. We extract message attributes from the CAN message samples. When a CAN message job is transmitted on the CAN bus, three attributes are directly exposed: 1) ID (identifier) (i.e., *Msg_ID*); 2) Data Length Code (DLC) (i.e., *Msg_Length* in the CAN communication matrix); and 3) content. In automotive CAN networks, timestamp is the bit time when a CAN controller receives a message and is relative to the start time of the CAN channel. Although the timestamp is not displayed in a message, a CAN controller can read a 16-byte counter value (i.e., timestamp) from its *CAN_RDTxR* register. Fig. 8 shows a simple message attribute extraction, where the values of four attributes (i.e., timestamp, ID, DLC, and content) are displayed according to each CAN message job.

C. CAN Message Block

CAN message block is a group of multiple CAN message jobs into a block. In [11], 64 consecutive CAN message jobs are constructed into one block according to the ascending order of the timestamp values of message jobs in the CAN message samples; In general, the CAN message block in [11] comes from different messages (i.e., these 64 consecutive CAN message jobs can have different message IDs). However, the CAN message block consisting of 64 consecutive CAN message jobs has a serious deviation in detection precision. The explanations are below.

(1) Assuming that the CAN message block consisting of 64 consecutive CAN message jobs is an attacked CAN message block and named *B1*, then an attacked CAN message block (named *B2*) generated by the GAN generator will be similar to *B1*. However, it is almost impossible to generate CAN message blocks that are similar to *B1* and *B2* later in actual automotive CAN networks. The specific reason is that *B1* is only a CAN message block collected from

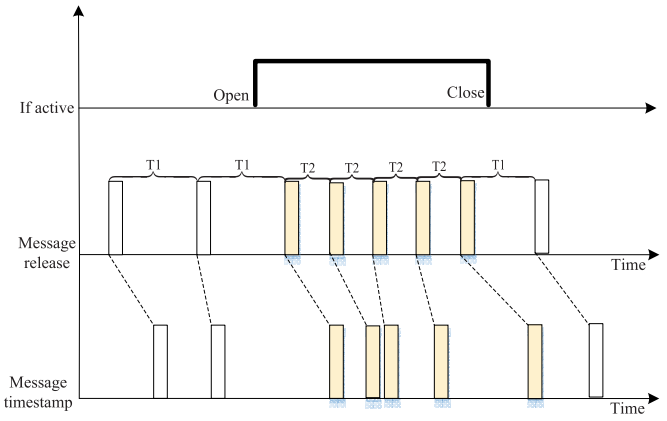


Fig. 9. Relationship between them released time and timestamp for a *CA* message.

automotive CAN networks in the preprocessing phase, and it cannot elaborately reflect the subsequent CAN message blocks generated in automotive CAN networks; in other words, *B1* is a rough CAN message block.

(2) For the CAN message blocks currently generated in automotive CAN networks, the GAN discriminator discriminates that these blocks similar to *B1* are attacked CAN message blocks, and the remaining CAN message blocks are normal blocks; therefore, most of the attacked CAN message blocks are misjudged as normal CAN message blocks.

To overcome the problem of the rough CAN message block in [11], we have the following improvement: all CAN message jobs in a CAN message block are from the same CAN message (i.e., all CAN message jobs have the same message ID). This method actually refers to the message classification mentioned in Section III.A. The details are explained below.

We know that the timestamp and released time of a message job are two different concepts. However, the timestamp values and released time values in a message basically present a similar pattern. Let us take the *CA* message in Fig. 7 as an example to explain the relationship between the released time values and timestamps.

From Fig. 9, we can see that the timestamp intervals between two adjacent message jobs are not fixed, but are sometimes long or sometimes short. The reason for the above phenomenon is that the CAN bus adopts priority-based non-preemptive scheduling; when multiple CAN messages compete for the same CAN bus, the high-priority CAN messages will interfere with the current message, thereby leading to the delay of the current message. Even so, the current message job has completed its transmission before the next message job is released; due to the fact that the developed real-time automotive CAN networks based on the OEM's CAN communication matrix require no message loss. Therefore, the timestamp of the current job is less than the released time of the next message job.

We assume that Fig. 9 is a normal CAN message block with 8 jobs in the given interval; once this CAN message block is attacked, then it may have the following exceptions.

(1) The number of jobs has become bigger (i.e., 9 jobs) or smaller (i.e., 7 jobs). The possible reason for the increase in the number of jobs is that the CAN message block has suffered from the injection attack; the possible reason for the decrease in the number of jobs is that the block has suffered from the masquerade attack.

(2) The transmission of 8 jobs is too concentrated or too scattered. The possible reason for this phenomenon is that the CAN message block has suffered from the DoS attack.

TABLE II
MESSAGE TYPES AND CORRESPONDING LABELS IN THIS PAPER

Message type	Label
attacked&Cycle	0
attacked&Event	1
attacked&IfActive	2
attacked&CE	3
attacked&CA	4
normal&Cycle	5
normal&Event	6
normal&IfActive	7
normal&CE	8
normal&CA	9

Timestamp	ID	DLC	Label
147819837.0001,	0440,	8,	1
147819837.0012,	0440,	8,	1
.	.	.	.
.	.	.	.
.	.	.	.
147819838.0012,	0440,	8,	1
.	.	.	.
.	.	.	.
.	.	.	.

Fig. 10. Simple CAN message block (ID = 0440) with the *attacked&Event* label.

D. Message Label

We establish 10 types of labels based on whether the message is attacked or not combining message classification, as shown in Table II. For instance, if a *Cycle* CAN message block has been attacked, then its label name is "*attacked&Cycle*" and its label code is "0".

On the basis of the CAN message block of Section III.C and the message labels in Table 10, we provide a simple CAN message block (ID = 0440) with the *attacked&Event* label, as shown in Fig. 10. The difference between Figs. 8 and 10 is below: Fig. 8 is a message attribute extraction for all CAN messages (i.e., including all message IDs), whereas Fig. 10 is a simple CAN message block whose ID is 0440 with the attacked&Event label (i.e., including only one message with ID = 0440 and Label = 1).

IV. DEEP LEARNING GAN MODEL

A. GAN Model Organization

The basic principle of the GAN model is shown in Fig. 11, where *G* (Generator) and *D* (Discriminator) are two core component networks.

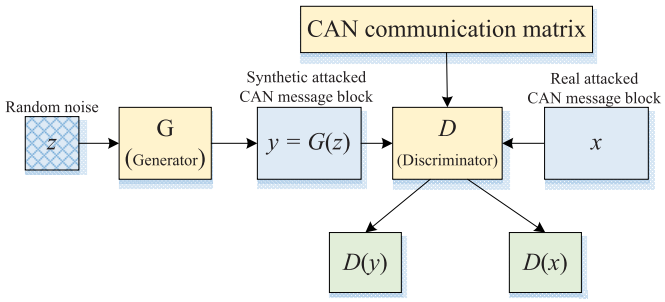


Fig. 11. Basic principle of GAN Model.

(1) G is a generator that generates attacked CAN message blocks; it receives a random noise z as input, and aims to output a synthetic attacked CAN message block, which is denoted as $y = G(z)$.

(2) D is a discriminator, which discriminates whether a CAN message block is "attacked" or not.

- One input parameter of D is x, which represents a real attacked CAN message block; the output *D(X)* represents the probability that x is an attacked CAN message block. The output probability belongs to the range of [0,1]; if *D(X)* is 1, it means that x is a 100% real attacked CAN message block; if *D(x)* is 0, it means that x cannot be an attacked CAN message block. Note that x is the attacked CAN message block extracted from the CAN message samples in the preprocessing phase. Therefore, to illustrate the discrimination ability of D, we hope that $D(x)$ is as close to 1 as possible.
- Another input parameter of D is y, which represents a synthetic attacked CAN message block outputted by G; the output *D(y)* represents the probability that y is an attacked CAN message block. The output probability belongs to the range of [0,1]; if *D(y)* is 1, it means that y is a 100% attacked CAN message block; if *D(y)* is 0, it means that y cannot be an attacked CAN message block. G hopes that the CAN message block generates "the closer to the real, the better". In other words, G hopes that $D(y)$ is as close to 1 as possible. However, the stronger the discriminative ability of D, the smaller the $D(y)$ value as possible.

B. Enhanced GAN Discriminator

Considering that the GAN model in [11] fails to detect the data tampering threat, this paper adds the CAN communication matrix into the GAN discriminator. Let us explain the reason below.

In [11] and this paper, only three attributes (i.e., timestamp, ID, and DLC) are included into the CAN message block (see Fig. 10), whereas the "content" attribute is discarded. The reason is that the contents of many messages are variable in practice. For example, the ECU status message transmits the status of the ECU at different times; thus, the content is variable (see Fig. 3). This is especially true for connected vehicles and autonomous vehicles, because they need to collect enough data collected from the external environment to assist them in making driving decisions; the external environment is complex and changeable, such that the content of messages changes frequently. However, attackers prefer to tamper with the message data to achieve serious destructiveness. If the content of a message is maliciously tampered with by an attacker, then this message will affect the correct decision of the autonomous vehicles, thereby leading to serious catastrophic consequences.

In summary, we cannot add the "content" attribute to the CAN message block because the change in content does not imply the presence of a data tampering threat. Meanwhile, we should also note that the change in content may be a data tampering threat.

To detect the data tampering threat, an enhanced GAN discriminator is built through adding the OEM's CAN communication matrix into it, thereby increasing its ability to discriminate against the data tampering threat. The details are blow.

We know that a CAN message is encapsulated by one or more signals (i.e., packing signals to frames, signal packing, frame packing). The CAN communication matrix lists the number of signals for a message, as well as the minimum and maximum values of each signal. For instance, Fig. 2 shows that the message *D_400* has 6 signals. The minimum value (i.e., *Signal_Min_Value* in Fig. 2) and the maximum value (i.e., *Signal_Max_Value* in Fig. 2) of each signal are given. For the message *D_400*, we can divide this message content into 6 signal values according to the *Byte_Order*, *Start_Byte*, *Start_Bit*, and *Bit_Length* values of each signal in the CAN communication matrix. Then, we can discriminate whether each signal value is in its individual range between its minimum and maximum values. As long as a signal value is not in the corresponding range, the content of the message can be suffered from the data tampering threat. In addition to detecting the data tampering threat, the GAN discriminator can obtain the release modes of CAN message blocks through the CAN communication matrix, thereby improving the detection precision of other types of attacks.

In summary, we make the following improvement to the GAN discriminator: as the CAN communication matrix contains a wealth of information, we develop a function based on the CAN communication matrix to enhance the detection of the data tampering threat, thereby improving the discriminate ability of the GAN discriminator. Furthermore, the function enhances the learning ability of the GAN discriminator through referring to the details in the CAN communication matrix.

C. Generator Training and Loss Function Design

As mentioned earlier, although instruction attacks on automotive CAN networks are very threatening, the actual attacked messages collected by automotive engineers are insufficient and far less than normal messages. The imbalanced amounts between the attacked and normal message samples lead to insufficient training of the GAN discriminator; therefore, it is hard to achieve high-precise intrusion detection technique.

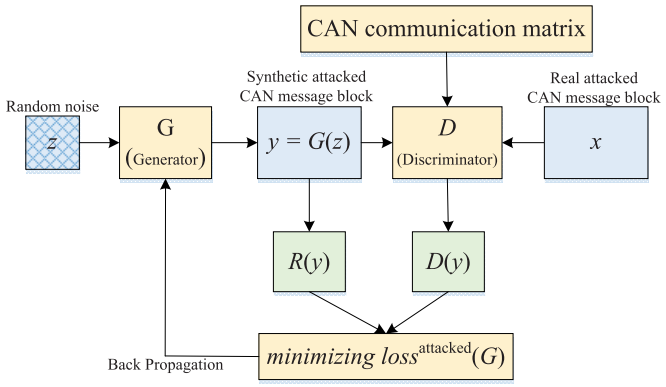


Fig. 12. GAN generator training for generating synthetic attacked CAN message blocks.

By fixing the GAN discriminator, we can train the GAN generator to generate synthetic attacked CAN message blocks, thereby providing enough attacked CAN message blocks for the GAN discriminator training. In other words, the aim of the GAN generator training is to supplement enough attacked CAN message blocks. Fig. 12 shows the principle of the GAN generator training for generating synthetic attacked CAN message blocks.

(1) Random noise z is used as the input of the GAN generator, and a synthetic attacked CAN message block denoted by $y = G(z)$ is generated through forward propagation to the GAN generator.

(2) Let $R(y)$ represent the actual probability set of the synthetic attacked CAN message block y under 10 different labels; $O(y)$ is denoted by

$$R(\mathrm{y}) = (p(y_0), p(y_1), \ldots, p(y_9)).$$

$p(y_j)$ $(n \in [1, 9])$ represents the actual probability of y with label j.

In Table II, we have defined 10 labels, including 5 attacked labels and 5 normal labels. Therefore, $R(\mathrm{y})$ contains 10 values, which are the actual probability values of the corresponding CAN message blocks with different labels. For instance, assume that we have

$$R(\mathrm{y}) = (0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0),$$

The actual probability values of y under 10 different types of CAN message blocks are shown in Table III. Considering that $attacked\&CA$ has the maximum probability value of 1.0, y is an attacked *CA* CAN message block.

(3) The synthetic attacked CAN message block y is input to the GAN discriminator for discrimination, and the discrimination result denoted by $D(y)$ is obtained:

$$D(\mathrm{y}) = (q(y_0), q(y_1), \ldots, q(y_9)).$$

$q(y_j)$ $(n \in [1, 9])$ represents the D-discriminated probability value of y with label j.

The 10 results contained in $D(\mathrm{y})$ are D-discriminated probability values of y under 10 different labels. For instance, assume that we have

$$D(\mathrm{y}) = (0.3, 0.1, 0.2, 0.5, 0.7, 0.6, 0.1, 0.1, 0.0, 0.1).$$

TABLE III
MESSAGE TYPES AND THE CORRESPONDING LABELS IN THIS PAPER

Message type	Label Code	Real probability $R(y)$	D-discriminated probability $D(y)$
attacked&Cycle	0	0	0.3
attacked&Event	1	0	0.1
attacked&IfActive	2	0	0.2
attacked&CE	3	0	0.5
attacked&CA	4	**1**	0.7
normal&Cycle	5	0	0.6
normal&Event	6	0	0.1
normal&IfActive	7	0	0.1
normal&CE	8	0	0.0
normal&CA	9	0	0.1

Then, the D-discriminated probability values of y under 10 different labels are shown in Table III. Considering that $attacked\&CA$ has the maximum probability value, y is the most likely to be an attacked *CA* CAN message block.

The GAN discriminator is regarded as a binary classifier (or 0-1 classifier) to distinguish the authenticity of the synthetic attacked CAN message blocks generated by the GAN generator; therefore, we use the cross entropy to distinguish the similarity of two probability distributions. Since $R(y)$ and $D(y)$ have different probability sets, we define the cross entropy when the GAN generator generates the attacked CAN message block:

$$CE^{\text{attacked}}(G) = -\sum_{j=0}^{9} p(y_j) \times \log q(y_j).$$

The cross entropy loss function with m CAN message blocks is calculated by

$$loss^{\text{attacked}}(G) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=0}^{9} p(y_j) \times \log q(y_j).$$

$loss^{\text{attacked}}(G)$ is back propagated to the GAN generator. To make the G-generated synthetic attacked CAN message blocks get closer and closer to the real attacked CAN message blocks, the loss function should be minimized; through the back propagation, the loss function prompts the GAN generator to optimize its parameters.

D. Discriminator Training and Loss Function Design

By fixing the GAN generator, we can train the GAN discriminator to discriminate synthetic attacked CAN message blocks, thereby enhancing the discrimination ability of the GAN discriminator to the attacked CAN message blocks. In other words, the aim of the GAN discriminator training is to improve the precision of the intrusion detection technique. Fig. 13 shows the principle of the GAN discriminator training for improving the precision of the intrusion detection technique.

A real attacked CAN message block x is used as the input of the GAN discriminator, and a discrimination result called $D(x)$ is obtained through back propagation to the GAN discriminator.

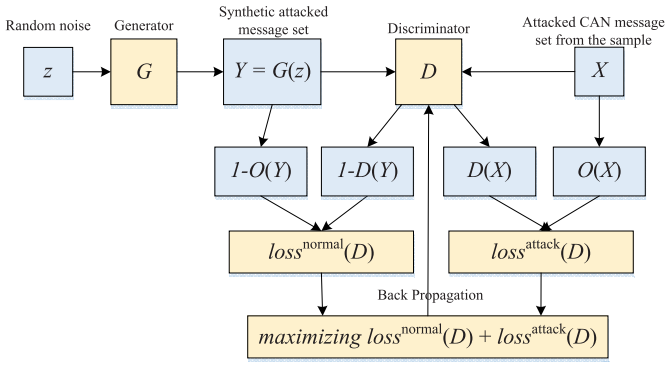


Fig. 13. GAN discriminator training for intrusion detection.

Let $R(x)$ represent the actual probability set of the real attacked CAN message block x under 10 different labels; $R(x)$ is denoted by

$$R(\mathrm{x}) = (p(x_0), p(x_1), \ldots, p(x_9)).$$

Let $D(x)$ represent the D-discriminated actual probability set of the real attacked CAN message block x under 10 different labels; $D(x)$ is denoted by

$$D(\mathrm{x}) = (q(x_0), q(x_1), \ldots, p(x_9)).$$

Since $R(x)$ and $D(x)$ may have different probability values, we define the cross entropy when the GAN discriminator discriminates the real attacked CAN message blocks:

$$CE^{\text{attacked}}(D) = -\sum_{j=0}^{9} p(x_j) \times \log q(x_j).$$

The corresponding cross entropy loss function with m CAN message blocks is calculated by

$$loss^{\text{attacked}}(D) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=0}^{9} p(x_j) \times \log q(x_j).$$

When this loss is propagated back to the GAN discriminator, it tells D that x is an actual attacked CAN message block.

Besides $loss^{\text{attacked}}(D)$ for the GAN discriminator, another cross entropy loss function is defined. We first provide the cross entropy when the GAN discriminator discriminates the normal attacked CAN message blocks:

$$CE^{\text{normal}}(D) = -\sum_{j=0}^{9} \left(1 - p(y_j)\right) \times \left(1 - \log q(y_j)\right).$$

$1 - p(y_j)$ $(j \in [1, 9])$ represents the real probability that y does not has the label j; $1 - q(y_j)$ $(n \in [1, 9])$ represents the D-discriminated probability that y does not has the label j. The corresponding cross entropy loss function with m CAN message blocks is calculated by

$$loss^{\text{normal}}(D) = -\frac{1}{m}\sum_{j=0}^{9} \left(1 - p(y_j)\right) \times \left(1 - \log q(y_j)\right).$$

When this loss is propagated back to the GAN discriminator, it tells the GAN discriminator that y is a normal attacked CAN message block.

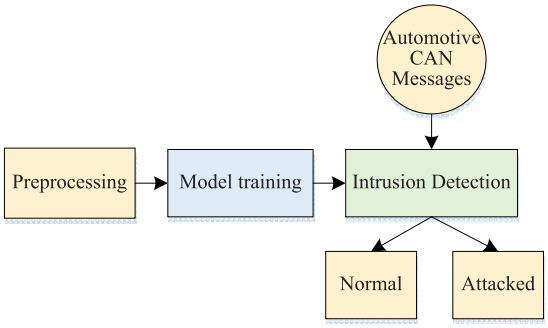


Fig. 14. Flow of the deep learning GAN model for the intrusion detection technique in automotive CAN networks.

Finally, the total cross entropy loss function for the GAN discriminator with m CAN message blocks is calculated by

$$loss(D) = loss^{\text{normal}}(D) + loss^{\text{attacked}}(D).$$

$loss(G)$ is back propagated to the GAN discriminator. In order to make the discrimination of the GAN discriminator stronger, the $loss(D)$ value should be larger; essentially, $D(x)$ should be larger, and $D(y)$ should be smaller (i.e., 1- $D(y)$ should be larger) in essence. The back propagation prompts the GAN discriminator to optimize its parameters.

E. Intrusion Detection Using the GAN Discriminator

In the GAN model training phase, the objective of the GAN generator is to generate real CAN message blocks as much as possible to deceive the GAN discriminator, whereas the objective of the GAN discriminator is to try to distinguish the CAN message block generated by the GAN generator from the real attacked CAN message blocks collected from the CAN message sample. In this way, the GAN generator and the GAN discriminator constitute a dynamic "game process." In the end, a good output is generated through the mutual game learning of the GAN generator and the GAN discriminator.

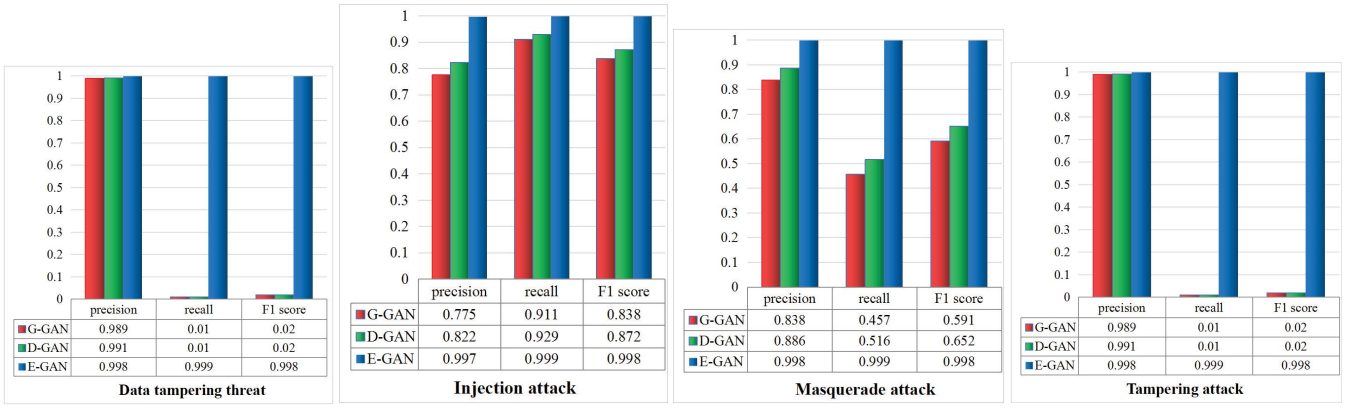
After the training is completed, the GAN generator is used to conduct the intrusion detection. We use the GAN generator to identify whether the automotive CAN network has suffered from the intrusion attacks. Fig. 14 shows the flow of the deep learning GAN model for the intrusion detection technique in automotive CAN networks.

V. Experiments

A. Experimental Process and Metrics

The experimental process contains three parts: 1) offline GAN model training on a desktop computer; 2) online GAN model test in a CAN network prototype; and 3) online detection evaluation for a real vehicle.

1) Offline GAN Model Training on a Desktop Computer: We use the deep learning framework PyTorch 1.1.0 as training environments; meanwhile, we use the CUDA 0.1.168 acceleration module to provide GPU computing acceleration for the GAN model programmed in Python. The experiment is deployed on a workbench with 16GB DRAM, NVIDIA GP102 (TITAN Xp) GPU, Intel(R) Xeon(R)CPU E5-2673 v3 2.40GHz processor, which runs the Ubuntu 18.04.1 operating system.



(a) DoS attack. (b) Injection attack. (c) Masquerade attack. (d) Data tampering threat.

Fig. 15. Precision, recall, and F1 score values for three intrusion detection techniques under different attacks.

2) Online GAN Model Test in a CAN Network Prototype: Our CAN network prototype consists of three ECU nodes. Each node contains an Arduino UNO Rev3 development board and a CAN bus shield with a MCP2515 controller and MCP2551 transceivers provided by Seeed Studio, which is a leading open source hardware company. The CAN network prototype is configured with the bandwidth of 500kbps for using in real vehicles.

3) Online Detection Evaluation for a Real Vehicle: we implement the GAN model-based intrusion detection system in a Xilinx Spartan 6 FPGA. The intrusion detection system is connected to the automotive CAN networks through the OBD II port of the real vehicle. The CAN information samples are collected from the actual CAN messages of a vehicle model, and the corresponding DBC file is also provided to us by the OEM. Since the information in the DBC file is an important information asset and intellectual property rights of the OEM, it is not convenient to publish the DBC file and the CAN message sample.

We select three evaluation metrics (including precision, recall, and F1 score) from the Scikit-learn library [24] to evaluate the experimental results. The detailed explanations of the three evaluation metrics are below.

Table IV shows the confusion matrix used for metric evaluation; in this confusion matrix, "A" means "Attacked" and "N" means "Normal". In Table IV, we have: 1) AA' represents the number of CAN message blocks that are actually attacked and detected as attacked; 2) AN' represents the number of CAN message blocks that are actually attacked but detected as normal; 3) NA' represents the number of CAN message blocks that are actually normal but detected as attacked; and 4) NN' represents the number of CAN message blocks that are actually normal and detected as normal.

(1) Precision refers to the proportion of the number of CAN message blocks that are actually attacked and detected as attacked against the number of CAN message blocks that are detected as attacked, namely,

$$precision = \frac{AA'}{AA' + NA'}.$$

TABLE IV
CONFUSION MATRIX USED FOR METRIC EVALUATION IN THIS PAPER

Confusion matrix		Detection Value	
		A' (attacked)	N' (normal)
Actual value	A (attacked)	AA'	AN'
	N (normal)	NA'	NN'

The precision reflects the ability of the GAN discriminator not to misjudge a normal CAN message block as an attacked CAN message block. The best value for precision is 1, and its worst value is 0.

(2) Recall refers to the proportion of the number of CAN message blocks that are actually attacked and detected as attacked against the number of CAN message blocks that are actually attacked, namely,

$$recall = \frac{AA'}{AA' + AN'}.$$

The recall rate reflects the ability of the GAN discriminator to find all attacked CAN message blocks. The best value for recall is 1, and its worst value is 0.

(3) F1 score can be interpreted as the weighted average of precision and recall, and it is calculated by

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

The best value of F1 score is 1, and its worst value is 0.

B. Detection Results and Analysis

According to the training results for the GAN model in [11], the best model can be obtained by using a labeled CAN message block and a noise to participate in the CAN message block generation. We use the same parameters mentioned above for the GAN model training, and focus on the comparison of experimental results.

We collect four pieces of CAN message blocks with different types of security threats, including the DoS, injection, masquerade, and data tampering threats. Fig. 15 shows the

precision, recall, and F1 score values for three intrusion detection techniques under different attacks. G-GAN and D-GAN represent the intrusion detection techniques that train the GAN generator and the GAN discriminator, respectively [11]. E-GAN represents the intrusion detection technique that is developed in this paper.

(1) In the CAN message samples with the DoS attack, the three intrusion detection techniques all have high performance, as shown in Fig. 15(a). The main reason is that the DoS attack blocks the transmissions of low-priority messages through repeatedly sending high-priority attacked messages in automotive CAN networks. Such an attack is easy to be detected and misjudgment is relatively low. Our E-GAN model is superior over G-GAN and D-GAN in three indicators, because the CAN communication matrix helps to enhance the learning ability of E-GAN.

(2) E-GAN has higher recall than the G-GAN and D-GAN against the injection attack. Particularly, E-GAN exceeds G-GAN and D-GAN in precision by 22.27% and 17.55%, respectively (see Fig. 15(b)). The inherent reason is that G-GAN and D-GAN easily misjudge the normal *Event* and *CE* messages as attacked messages due to they lack explicit message classifications.

(3) Fig. 15(c) shows that the precision and recall values caused by the masquerade attack for G-GAN and D-GAN are both less than those of E-GAN. Particularly quite low recall values for G-GAN and D-GAN. Since the masquerade attack is mainly to intrude an ECU and pretend the ECU to send messages, it is extremely destructive, including speeding up or delaying the messages sending as well as sending mendacious messages. Considering that G-GAN and D-GAN lack message classifications and the constructed CAN message blocks are rough, their intrusion detection ability against the masquerade attack is not ideal.

(4) E-GAN shows that the recall as high as 99.9% against the data tampering threat; however, the recall values for G-GAN and D-GAN are extremely low to 1%. As explained earlier, previous intrusion detection techniques (including G-GAN and D-GAN) are hard to detect the data tampering threat because they do not make full use of the CAN communication matrix provided by the OEM. The good news is that E-GAN can perform effective intrusion detection against the data tampering threat.

In addition to performance indicators (i.e., precision, recall and F1 score), the detection time is also an important indicator. Since the attributes in a CAN message block for E-GAN are exactly the same as that for G-GAN and D-GAN, the detection time difference among the techniques is very small. The specific result is that the average detection time of one CAN message block for G-GAN, D-GAN and E-GAN is 0.16 ms, 0.11 ms and 0.09 ms, respectively.

VI. Conclusion

An increasing number of security threats affect the normal message transmission in automotive CAN networks. To analyze security threats in automotive CAN networks, this study develops an enhanced deep learning GAN model-based intrusion detection technique. Our deep learning GAN model adopts elaborate CAN message blocks through introducing the CAN communication matrix provided by automotive OEM. Our deep learning GAN model can detect the data tampering threat, whereas the previous deep learning models fail to do. Our deep learning GAN model simultaneously shows high precision, high recall, and high F1 score values against the DoS, injection, masquerade, and data tampering threats. The future work could consider compressing the GAN model to reduce system resource usage, thereby achieving lightweight deployment.

Acknowledgment

The authors would like to express their gratitude to an Associate Editor and three anonymous reviewers whose constructive comments have helped to improve the manuscript.

References

[1] J. Bräuninger, R. Emig, T. Küttner, and A. Loffler, "Controller area network for truck and bus applications," *SAE Trans.*, vol. 99, pp. 704–714, Jan. 1990.

[2] W. Wu *et al.*, "A survey of intrusion detection for in-vehicle networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 3, pp. 919–933, Mar. 2020.

[3] S. Ghane, A. Jolfaei, L. Kulik, K. Ramamohanarao, and D. Puthal, "Preserving privacy in the Internet of connected vehicles," *IEEE Trans. Intell. Transp. Syst.*, early access, Jan. 15, 2020, doi: 10.1109/TITS.2020.2964410.

[4] S. Woo, H. J. Jo, and D. H. Lee, "A practical wireless attack on the connected car and security protocol for in-vehicle CAN," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 993–1006, Apr. 2014.

[5] G. Xie, L. T. Yang, W. Wu, K. Zeng, X. Xiao, and R. Li, "Security enhancement for real-time parallel in-vehicle applications by CAN FD message authentication," *IEEE Trans. Intell. Transp. Syst.*, early access, Jun. 23, 2020, doi: 10.1109/TITS.2020.3000783.

[6] F. Faezeh, M. S. Haghighi, S. Barchinezhad, and A. Jolfaei, "Detection and compensation of covert service-degrading intrusions in cyber physical systems through intelligent adaptive control," in *Proc. IEEE Int. Conf. Ind. Technol. (ICIT)*, Feb. 2019, pp. 1143–1148.

[7] M. Tang, M. Alazab, and Y. Luo, "Big data for cybersecurity: Vulnerability disclosure trends and dependencies," *IEEE Trans. Big Data*, vol. 5, no. 3, pp. 317–329, 2019, doi: 10.1109/TBDATA.2017.2723570.

[8] G. Xie, Y. Li, Y. Han, Y. Xie, G. Zeng, and R. Li, "Recent advances and future trends for automotive functional safety design methodologies," *IEEE Trans. Ind. Informat.*, vol. 16, no. 9, pp. 5629–5642, Sep. 2020.

[9] S. Wang *et al.*, "A fast CP-ABE system for cyber-physical security and privacy in mobile healthcare network," *IEEE Trans. Ind. Appl.*, vol. 56, no. 4, pp. 4467–4477, 2020, doi: 10.1109/TIA.2020.2969868.

[10] E. Hodo *et al.*, "Threat analysis of IoT networks using artificial neural network intrusion detection system," in *Proc. Int. Symp. Netw., Comput. Commun. (ISNCC)*, May 2016, pp. 1–6.

[11] Y. Yang, G. Xie, J. Wang, J. Zhou, Z. Xia, and R. Li, "Intrusion detection for in-vehicle network by using single GAN in connected vehicles," *J. Circuits, Syst. Comput.*, to be published, doi: 10.1142/S0218126621500079.

[12] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.

[13] (2020). *CAN-Bus Basics–Communication Matrix or CAN Matrix*. [Online]. Available: https://www.caneasy.de/caneasyhelp/k-matrix_oder_can-matrix.htm

[14] W. Choi, K. Joo, H. J. Jo, M. C. Park, and D. H. Lee, "VoltageIDS: Low-level communication characteristics for automotive intrusion detection system," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 8, pp. 2114–2129, Aug. 2018.

[15] J. Zhou, P. Joshi, H. Zeng, and R. Li, "BTmonitor: Bit-time-based intrusion detection and attacker identification in controller area network," *ACM Trans. Embedded Comput. Syst.*, vol. 18, no. 6, pp. 1–23, Nov. 2019.

[16] H. M. Song, H. R. Kim, and H. K. Kim, "Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2016, pp. 63–68.

[17] M. Muter and N. Asaj, "Entropy-based anomaly detection for in-vehicle networks," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2011, pp. 1110–1115.
[18] W. Wu *et al.*, "Sliding window optimized information entropy analysis method for intrusion detection on in-vehicle networks," *IEEE Access*, vol. 6, pp. 45233–45245, 2018.
[19] A. Taylor, S. Leblanc, and N. Japkowicz, "Anomaly detection in automobile control network data with long short-term memory networks," in *Proc. IEEE Int. Conf. Data Sci. Adv. Analytics (DSAA)*, Oct. 2016, pp. 130–139.
[20] M.-J. Kang and J.-W. Kang, "Intrusion detection system using deep neural network for in-vehicle network security," *PLoS ONE*, vol. 11, no. 6, Jun. 2016, Art. no. e0155781.
[21] H. M. Song, J. Woo, and H. K. Kim, "In-vehicle network intrusion detection using deep convolutional neural network," *Veh. Commun.*, vol. 21, Jan. 2020, Art. no. 100198.
[22] E. Seo, H. M. Song, and H. K. Kim, "GIDS: GAN based intrusion detection system for in-vehicle network," in *Proc. 16th Annu. Conf. Privacy, Secur. Trust (PST)*, Aug. 2018, pp. 1–6.
[23] (2020). *CAN DBC File–Convert Data Live (Wireshark, j1939)*. [Online]. Available: https://www.csselectronics.com/screen/page/dbc-database-can-bus-conversion-wireshark-j1939-example/language/en
[24] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.

Guoqi Xie (Senior Member, IEEE) received the Ph.D. degree in computer science and engineering from Hunan University in 2014. He was a Post-Doctoral Research Fellow with Nagoya University. He is currently a Professor with Hunan University. His current research interests include real-time systems, automotive embedded systems, and embedded safety and security. He received the 2018 IEEE TCSC Early Career Researcher Award. He is an ACM Senior Member. He is currently serves on the Editorial Board for *Journal of Systems Architecture*, *Microprocessors and Microsystems*, and *Journal of Circuits, Systems and Computers*.

Laurence T. Yang (Fellow, IEEE) received the B.E. degree in computer science and technology and the B.S. degree in applied physics from Tsinghua University, Beijing, China, in 1992, and the Ph.D. degree in computer science from the University of Victoria, Victoria, BC, Canada, in 2006. He is currently a Professor with St. Francis Xavier University, Antigonish, NS, Canada. His research has been supported by the National Sciences and Engineering Research Council and the Canada Foundation for Innovation. His research interests include parallel and distributed computing, embedded and ubiquitous/pervasive computing, and big data.

Yuanda Yang received the M.S. degree in communication engineering from Hunan University in 2020. He is currently an Engineer with Merchants Union Consumer Finance Company Ltd., Shenzhen, China. His research interests include intrusion detection and deep learning.

Haibo Luo received the Ph.D. degree from the College of Physics and Information Engineering, Fuzhou University, Fuzhou, China. He is currently an Associate Professor with Minjiang University, China. His research interests include cyber-physical systems, the Internet of Things, wireless networks, edge computing, and mobile computing.

Renfa Li (Senior Member, IEEE) is currently a Professor and the Chair of Embedded and Cyber-Physical Systems with Hunan University. He is the Chair of the Key Laboratory for Embedded and Cyber-Physical Systems. His major research interests include computer architectures, embedded computing systems, cyber-physical systems, and the Internet of Things. He is a member of the Council of CCF and a Senior Member of ACM.

Mamoun Alazab (Senior Member, IEEE) received the Ph.D. degree in computer science from the School of Science, Information Technology and Engineering, Federation University of Australia. He is currently an Associate Professor with the College of Engineering, IT and Environment, Charles Darwin University, Australia. He is also a Cyber Security Researcher and a Practitioner with industry and academic experience. He works closely with government and industry on many projects, including the Northern Territory (NT) Department of Information and Corporate Services, IBM, Trend Micro, and the Australian Federal Police (AFP). His research is multidisciplinary that focuses on cyber security and digital forensics of computer systems with a focus on cybercrime detection and prevention, including cyber terrorism and cyber warfare. He is the Founder and the Chair of the IEEE Northern Territory (NT) Subsection Detection and Prevention.