ASTES

Advances in Science, Technology and Engineering Systems Journal
Vol. 11, No. 1, 25-32 (2026)
www.astesj.com
Special Issue on Innovation in Computing, Engineering Science & Technology

ASTESJ
ISSN: 2415-6698

Detection Method and Mitigation of Server-Spoofing Attacks on SOME/IP at the Service Discovery Phase

Kazuki Iehira[*1], Hiroyuki Inoue[2]

[1]*Division of Frontier Informatics, Kyoto Sangyo University, Kyoto, 603-8555, Japan*

[2]*Faculty of Information Science and Engineering, Kyoto Sangyo University, Kyoto, 603-8555, Japan*

ARTICLE INFO

Article history:
Received: 14 November, 2025
Revised: 18 December, 2025
Accepted: 20 December, 2025
Online: 11 January, 2026

Keywords:
Control system security
Ethernet
IDS
SOME/IP
SOME/IP-IDS
Spoofing attack

ABSTRACT

Service-oriented architecture has attracted attention in automotive development. The Automotive Open System Architecture (AUTOSAR) specifies Scalable Service-Oriented Middleware over IP (SOME/IP) as a key middleware for service-oriented communication in-vehicles. However, SOME/IP-based networks are vulnerable to server spoofing during the service discovery phase, enabling attackers to cause man-in-the-middle attacks by impersonating legitimate services. This paper proposes a method for detecting attacks in which an attacker spoofs a server during service discovery using a packet switch in a network employing SOME/IP. In addition, a mitigation method is proposed to discard malicious messages. The proposed method addresses the limitations of conventional encryption-based approaches by reducing both processing and communication loads. It also improves anomaly detection rate and detection timing, which have been identified as shortcomings in intrusion detection systems proposed in previous studies. In this study, we evaluated the method using a dataset consisting of 36 attack patterns that combined attack types, attack initiation timing, and message categories. Real-world evaluations demonstrated that the proposed method detected all server spoofing attempts. Further analysis of processing time and memory resource usage confirmed its feasibility for in-vehicle system.

1. Introduction

Recently, there has been an increasing demand for software and hardware updates via over-the-air (OTA) and plug-and-play (PnP) after delivery for consumers to add new features and update existing ones. Advanced driver assistance systems are becoming popular, and automotive software is becoming increasingly large and complex. Service-oriented architecture has attracted attention as a solution to these issues. Automotive open system architecture (AUTOSAR), a global development partnership in the automotive industry, provides specifications for Scalable Service-Oriented Middleware over IP (SOME/IP) [1] and SOME/IP Service Discovery (SOME/IP-SD) [2], which are service-oriented communication middleware.

Communication in-vehicle systems has strict performance requirements for communication establishment and delays [3], [4]. To meet these performance requirements, SOME/IP suppresses the amount of bandwidth used across the entire network using multicast for certain communications. SOME/IP is designed to communicate over the TCP/IP stack and does not have its own encryption and authentication mechanisms. In [5], the authors reported that analysis of the SOME/IP specification revealed the possibility of man-in-the-middle attacks, in which an attacker spoofs as a server providing a service or as a client using the service.

In [6], the use of IPSec and TLS/DTLS in SOME/IP was discussed; however, it was demonstrated that IPSec lack multicasting support, while TLS/DTLS incurred high communication overhead owing to its complex communication establishment process. In SOME/IP, both the server and the client perform event-triggered communications. The server periodically sends messages to registered clients using a unique publish/subscribe (Pub/Sub). Pub/Sub communication is used for communications where real-time performance, such as sensor information, is important. Pub/Sub employs multicast communication to reduce communication load because it sends the same information to multiple devices, which poses a challenge to the introduction of encryption and authentication. Therefore, it is important to develop an anomaly detection and mitigation method

[*]Corresponding Author Kazuki Iehira, Email: i2386157@cc.kyoto-su.ac.jp

that minimizes communication load and delay while effectively addressing server spoofing attacks.

In [7], an Intrusion Detection System (IDS) that detects the spoofing attack using SOME/IP is proposed. The timing of anomaly detection in this method is during communication for service use; however, there is a possibility of misdetecting normal communication as an anomaly, and there is a concern that communication related to safety, such as vehicle operation, may be mistakenly obstructed. Therefore, it is difficult to perform actions, such as communication termination, when an anomaly is detected. Additionally, anomaly detection methods based on monitoring communication values struggle to detect man-in-the-middle attacks that only involve spoofed message reception. Therefore, an anomaly detection and mitigation method that can detect spoofing attacks, including man-in-the-middle attacks, during service discovery, that is, before service utilization begins, is necessary.

In this study, we propose a detection and mitigation method that detects and mitigates the possibility of spoofing as a server attack during service discovery in networks using SOME/IP. In the proposed method, anomaly detection is performed before the service starts. Therefore, even if normal communication is mistakenly detected as an anomaly, there are no safety concerns because the service has not started operating. The contributions of this study are as follows:

- We propose a method for detecting all spoofing attacks on SOME/IP using a simple algorithm requiring fewer than 100 operations per message, considering implementation in embedded devices.
- We classified attack patterns into 36 types based on the classification of the attackers' messages sent and attack timing, including man-in-the-middle attacks. Through a real-world evaluation, we demonstrated that the proposed method could detect spoofing attacks in all patterns. In particular, for the 24 attack patterns, normal communication and attacks were correctly classified, enabling continued service use.
- For the remaining 12 attack patterns, normal communications and attacks cannot be correctly classified. These cases occur only before service use begins . Thus, by blocking the target communications from the first use of the service, the system can be maintained in a safe state.

The remainder of this paper is organized as follows. Section 2 describes the requirements for in-vehicle networks and the characteristics of the SOME/IP and SOME/IP-SD protocols used in in-vehicle networks, introduces solutions for attacks on in-vehicle networks and their protection, and explains these issues. Section 3 describes the anomaly detection mechanism and mitigation mechanism in the proposed method. Section 4 presents the results of an evaluation using vsomeip[8], a prototype implementation of SOME/IP, to demonstrate the effectiveness of the proposed method. Section 5 discusses the evaluation results. Finally, Section 6 presents the conclusions.

2. Related Research

2.1. In-Vehicle Network Requirements

Reference [3] classified communications in-vehicle networks based on their strict real-time requirements as static, real-time, IP-based, or web services. For various communications, such as sensor information, real-time performance is guaranteed by setting a communication cycle, and the maximum allowable communication delay is less than 10% of the communication cycle. In particular, for Static Real-Time Services used in Safety Electronics and Engine/Powertrain applications, communication is performed at a 1[ms] cycle; thus, the maximum allowable communication delay is 0.1[ms].

2.2. SOME/IP

SOME/IP is a service-oriented automotive middleware specification used to control communication messages [1]. SOME/IP is designed to communicate over TCP/IP or UDP/IP, and does not provide encryption or authentication mechanisms. It supports the following functions:

- Serialization:
  Conversion of transmitted and received data into communication formats.
- Remote Procedure Call (RPC) and message:
  Remote calls to functions and other messages.
- Event Notifications:
  Notification from server to client.
- Segmentation of UDP messages:
  Transfer SOME/IP messages that do not fit into a single UDP packet via UDP without fragmentation.

SOME/IP communicates using a TCP/IP protocol stack. It supports IPv4/IPv6 as the Internet layer and TCP and UDP as the transport layer. Table 1 lists the messages defined by SOME/IP. Each SOME/IP message contains a message ID composed of three components: a service ID, which specifies the type of service; a method ID, which specifies the type of method; and a session ID, which identifies communications between the same device pair. Attackers' ultimate goal is to tamper with these messages.

2.3. SOME/IP-SD

SOME/IP-SD is a middleware specification [2] that enables the discovery of available services on the system and the management of event message transmission on SOME/IP and supports the following functions.

- Service Discovery:
  Dynamic service discovery, and access control.
- Publish/Subscribe (Pub/Sub):
  Dynamic configuration of communication between Server and Client.

Table 2 shows a list of messages specified in SOME/IP-SD. SOME/IP-SD messages include a Service ID that indicates the type of service, and an Instance ID that identifies the server. In particular, in the case of SOME/IP-SD messages, the Message ID included in the lower-level protocol SOME/IP message is fixed at 0xFFFF8100; thus, the target service is identified by the Service ID included in the SOME/IP-SD message. SOME/IP supports multiple servers providing services with the same Service ID, in which case, these servers will have different Instance IDs.

Table 1: SOME/IP Message Type List

Kind (Value)	Description
REQUEST(0x00)	Sent from Client (Unicast) . Requests that require a response.
REQUEST_NORETURN (0x01)	Sent from Client (Unicast). Requests that do not require a response.
NOTIFICATION(0x02)	Sent from Server (Unicast or Multicast). One-way notification.
RESPONSE(0x80)	Sent from Server (Unicast). Response to REQUEST.
ERROR(0x81)	Sent from Server (Unicast). Error response.

Table 2: SOME/IP-SD Message Type List

Kind (Value)	Description
FIND SERVICE(0x00)	Send from Client (Multicast). Notification of service provider discovery.
(STOP) OFFER SERVICE (0x01)	Sent from Server (Unicast or Multicast). Notification of service provision status.
(STOP)SUBSCRIBE EVENTGROU (0x02)	Sent from Client (Unicast). Request to add/delete recipients of NOTIFICATION.
SUBSCRIBE EVENTGROUP ACK / NACK (0x03)	Sent from Server (Unicast). Success or failure of adding/deleting recipients of NOTIFICATION.

The Service Discovery function uses the OFFER SERVICE, STOP OFFER SERVICE, and FIND SERVICE to share information about available services between the server and client. After starting, the server waited for a certain period before sending the OFFER SERVICE. The client detects that the target service is available by receiving the OFFER SERVICE and begins using the service. If the client cannot receive an OFFER SERVICE for a certain period after startup, it sends a FIND SERVICE to search for a server that provides the target service. When the server receives a FIND SERVICE, it sends an OFFER SERVICE to the client. When the server stops providing the service, it sends a STOP OFFER SERVICE to notify the client. Attackers may send these messages during an attack. The proposed method monitors these messages to detect attacks.

2.4. Attacks on Vehicle-Mounted Networks

Controller area networks (CAN), which are widely used in automotive networks, are vulnerable to spoofing attacks because they cannot identify source devices. Reports indicate that spoofing attacks can be conducted by injecting CAN messages into the CAN bus or OBD-II diagnostic port, causing unauthorized operation of automotive devices [9], [10]. In SOME/IP, although the source device can be identified, the absence of a standard authentication mechanism allows attackers to forge device information and conduct spoofing attacks. In such cases, legitimate communication from authorized devices continues alongside malicious activity, resulting in a combination of normal and spoofed messages. This study refers to such scenarios as "Normal and Anomaly (Attack)." The communication sequence of this attack pattern is shown in Figure 1. Furthermore, this attack has the lowest difficulty level.

It has been reported that a DoS attack called a bus-off attack [11] that disables the sending device is possible in a CAN. A spoofing attack [12] using a bus-off attack has been reported, and this attack has the characteristic that only spoofing attacks are performed because the communication of the legitimate device is disabled. The feasibility of a DoS attack with the same characteristics as a bus-off attack against SOME/IP has been examined in previous studies [13]. In this study, a comprehensive analysis of the state transition specifications of SOME/IP-SD concluded that while it is possible to disable legitimate device communications, an Offer Service must be sent at least once from the legitimate server. In this study, such an attack pattern is referred to as "Anomaly (Attack) Only." The communication sequence of this attack pattern is shown in Figure 2. Furthermore, this attack has the highest difficulty level.

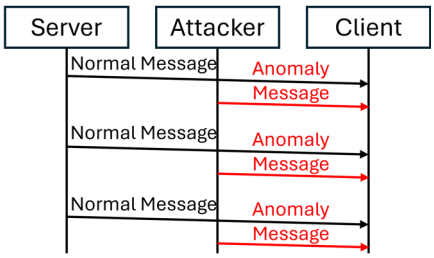

Figure 1: Communication Sequence of Normal and Anomaly (Attack)

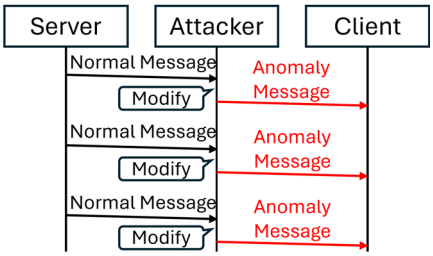

Figure 2: Communication Sequence of Man-in-the-middle attack

Because CAN always performs broadcast communication in a bus-type network topology, it is impossible to conduct a man-in-the-middle attack without physically changing the network connection. Man-in-the-middle attacks can be performed using the service discovery mechanism of SOME/IP [5]. Man-in-the-middle attacks against SOME/IP are conducted by attackers spoofing legitimate servers and sending an OFFER SERVICE, thereby inducing clients to use the services provided by the attackers. Furthermore, the difficulty level of this attack is medium.

2.5. Mitigation Measures Against Spoofing Attacks in Automotive Networks

For CAN, such as authentication and IDS, as countermeasures against spoofing attacks, have been proposed. AUTOSAR specifies a method for assigning MAC values to CAN communication messages between devices [14]. An IDS [15] that detects attacks based on the transmission cycle of information communicated in an in-vehicle network and the content of communications, such as sensor information, has been proposed. As a response to anomaly detection, [16] proposes a method that

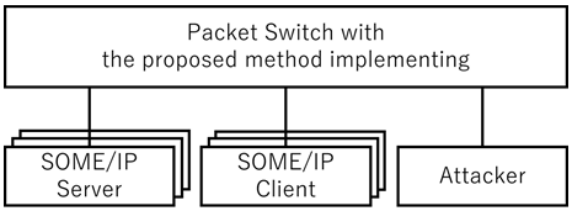

Figure 3: The proposed method assumes a network configuration where multiple servers, clients, and attackers are connected to different ports on a packet switch

discards CAN communication messages with invalid MAC values by monitoring all communications on the CAN bus.

In SOME/IP, encryption using IPSec and TLS/DTLS is being discussed; however, it has been demonstrated that high communication overhead is an issue owing to the lack of multicast support and the complex communication establishment process [6]. In [6], a mitigation method for SOME/IP using whitelists and authentication was proposed. However, there are concerns that flexibility may be reduced owing to whitelist management. An IDS [7] that monitors communication intervals, payload changes, and session counts as features has been proposed. However, because this method monitors communication related to service usage, there is a concern that false positives can affect safety-related communication, such as vehicle operation, making it difficult to take action, such as communication termination, when anomalies are detected.

3. Proposed Method

This paper proposes a detection and mitigation method to detect the possibility of spoofing as a server attack during service discovery in networks using SOME/IP.

3.1. Assumed Attacker

The network configuration targeted by the proposed method is shown in Figure 3, which is implemented using a packet switch and a server/client/attacker device connected in a one-to-many configuration. This study assumes that an attacker has the following characteristics:

- Connected to the same network as the attack server.
- Can send messages that do not follow the SOME/IP (-SD) specifications.
- No access to the internal information of devices other than the attacker.
- The physical connections of the network, including those of the attacker, cannot be changed.

Attacker communication can be classified into three types: Normal and Anomaly (attack), anomaly (attack) Only, and Man-in-the-Middle, as described in Section 2.4. Table 3 summarizes the characteristics of the communication patterns in these attacks from the perspective of communication between the server and attacker. In normal and anomalous cases, the attacker does not send a (STOP)OFFER SERVICE, a communication management service, but only sends messages for other actual data communications. Specifically, under Normal and Anomaly (attack) conditions, after the server starts providing services, the attacker spoofs as a server and sends a NOTIFICATION and RESPONSE. In a man-in-the-middle attack, all messages, including (STOP)OFFER SERVICE, are sent. In Anomaly Only, the server's communication is blocked by the attacker, and as shown in previous research [13], the server only sends an OFFER SERVICE once at startup and does not subsequently send any messages.

Table 3: Feature of Spoofing Attacks

Attack Pattern	Server's communication	Attacker's communication
Normal and Anomaly (Attack)	Send all messages.	Send all messages except (STOP) OFFER SERVICE.
Anomaly (Attack) Only	Do not send.	Send all messages.
Man-In-The-Middle	Send all messages.	Send all messages.

Man-in-the-middle and anomaly only attacks share the common characteristic that the attacker and client communicate one-to-one and that both the server and attacker send OFFER SERVICE messages. The only difference between them is whether there is communication between the server and attacker. Because this study focuses on whether OFFER SERVICE messages are sent during service discovery, these two attack patterns have the same characteristics. Therefore, we treat them as man-in-the-middle attacks without distinguishing between them.

3.2. Concept of The Proposal Method

This paper proposes a mitigation method that satisfies the following requirements: The proposed method is intended to be implemented on the switch.

Req.A) Prevent the Client from using services provided by the attacker in a spoofing attack with a pattern other than an anomaly (attack).

Req.B) The service can continue even if an anomaly is detected after the client starts using it.

Req.C) The increased communication latency for anomaly detection and mitigation method is within acceptable limits for communication in an in-vehicle network.

Req.D) The amount of read-only memory (ROM) and random-access memory (RAM) required to implement the proposed method was within the capacity of the microcontroller used to implement the in-vehicle system.

By imposing restrictions on the design of the systems that comprise the network, we propose a method for detecting the presence of attackers during service discovery at the packet switch and for protecting communications from malicious activities by attackers. In the proposed method, the following restrictions were imposed on the system design:

Cond.A) Have only one instance per Service ID (only one device to send an OFFER SERVICE per Service ID).

Cond.B) The Client starts after the time elapses until the server sends the OFFER SERVICE for the first time (starts using the service).

The proposed method does not impose restrictions on the number of devices that can be connected to a packet switch. However, for

Table 4: Internal State of Proposed Method

Internal state	Server's communication	Client's communication
Not offered (Initial state)	Allow communication on all devices.	Allow communication on all devices.
Offered	Allow communication only for devices included in the list.	Allow communication on all devices.
Anomaly	Allow communication only for devices included in the list.	Allow communication only for devices included in the list.

the convenience of explanation and evaluation in this study, we assumed that a maximum of 32 devices can be connected to a single switch.

In the proposed method, the internal states Not-offered (0x00), Offered (0x01), and Anomaly (0x02) shown in Table 4, the list of servers and clients that communicated in the past (communication-permitted device list) are managed by the packet switch, and the transfer of messages is determined based on the internal states and communication-permitted device list. The procedure for updating the internal state and communication-permitted device list is called the "anomaly detection algorithm," and the procedure for determining whether or not to forward messages according to the internal state and communication-permitted device list is called the "mitigation algorithm." In the proposed method, an anomaly detection algorithm and a mitigation algorithm are executed to determine whether to forward the received message. The communication-permitted device list can be set as communication-not-permitted (0x00) or communication-permitted (0x01), with the initial value being communication-not-permitted (0x00). The internal state, communication-permitted device list for the server, and communication-permitted devices for the client were managed for each service ID.

In the anomaly detection algorithm, communication-permitted (0x01) is set for devices that communicate while Condition A is satisfied (i.e., Not-offered (0x00) or Offered (0x01) state), and the communication-permitted device list is not updated after detecting communication that does not satisfy Condition A (i.e., Anomaly (0x02) state). Communication is permitted between a server and client with a communication-permitted (0x01) set, regardless of the internal state, thereby achieving Requirement B. As clients cannot begin using the service when only attackers are providing it under Condition B, attackers and clients are never simultaneously granted communication. Thus, the proposed method satisfies Requirement A.

3.3. Anomaly Detection Algorithm

In the anomaly detection algorithm, as shown in the state transition diagram in Figure 4, when the switch receives an OFFER SERVICE for the first time, it transitions from the Not-offered (0x00) state to the Offered (0x01) state and sets the communication-permitted (0x01) for the sending device (port). When the switch receives an OFFER SERVICE from a device that does not have communication-permitted (0x01) set in the Offered (0x01) state, it transitions to the Anomaly (0x02) state. The communication-permitted device list for the servers is updated only when the switch transitions to the Offered (0x01) state. The

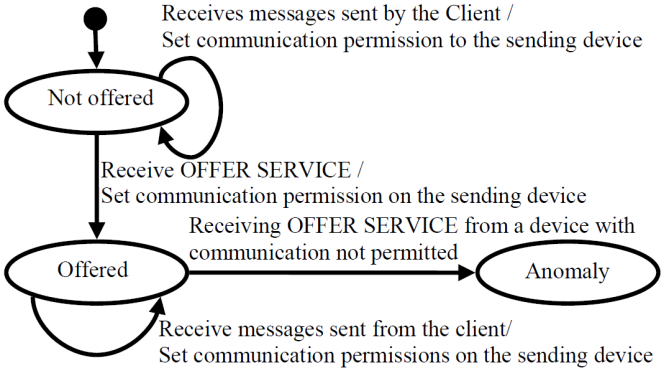

Figure 4: State machine in a proposal method where state transitions are triggered by communication from servers and clients

communication-permitted device list for clients is updated when the switch receives a message from a client in a state other than Anomaly (0x02).

The pseudocodes for the function that implements the anomaly detection algorithm is shown in algorithm 1. The process for determining whether the sender is a server or a client is shown in the pseudocode in algorithm 2 as a separate function. The pseudocode for the function that obtains the service ID targeted by the message from the SOME/IP and SOME/IP-SD headers is shown in algorithm 1. In these functions, the arguments msg_id and msg_type indicate the Message ID and Type contained in the SOME/IP message header, respectively; the arguments sd_service_id and sd_type indicate the Service ID and Type contained in the SOME/IP-SD message header, respectively; and the argument src indicates the port number to which the sending device is connected.

3.4. Mitigation Algorithm

The mitigation algorithm determines whether to forward a received message based on the internal state and the communication-permitted device list. Specifically, it identifies the internal status of the service ID included in the received message, and the values in the communication-permitted device list corresponding to the ports of the source and destination devices. If the internal status is not an Anomaly (0x02) or if communication-permitted (0x01) was set for both the source and destination devices in the communication-permitted device list, the message is judged to be transferable; otherwise, it is judged to be nontransferable. The pseudocode for the function that implements the mitigation algorithm is shown in algorithm 4. In this function, argument dst indicates the port number to which the destination device is connected. Other arguments were presented in Section 3.3.

4. Evaluation

In this section, we demonstrate that the proposed method can fulfill Requirement A-D described in Section 3.2. Requirement A-B, namely anomaly detection accuracy, will be evaluated on actual hardware. Requirement C-D, namely processing capacity and resource usage, will be evaluated on study.

4.1. Actual Machine Evaluation Environment

The evaluation uses a network configuration environment in which three devices—a legitimate server, legitimate client, and

Algorithm 1: Function to detect spoofing attacks

```
srv_lst, clt_lst: 32-bit integer array listing servers and clients
state: 8-bit integer array indicating the communication
permission status assigned to each port
service_id: 16-bit integer indicating the service ID
function ids(msg_id,msg_type,sd_service_id,sd_type,src)
    service_id = get_service_id(msg_id, sd_service_id);
    if sent_by_server(msg_id, msg_type, sd_type) then
        if state[service_id] == 0x00 then
            state[service_id] = 0x01
            srv_lst[service_id] = srv_lst[service_id] | (1<<src)
        else if state[service_id] == 0x01 then
            if ((srv_lst[server_id] >> src) & 0x01) == 0 then
                state[service_id] = 0x02
            end
        end
    else
        if state[service_id] != 0x02 then
            clt_lst[service_id] = clt_lst[service_id] | (1 << src)
        end
    end
end
```

Algorithm 2: Function to determine the sender

```
function sent_by_server (msg_id,msg_type,sd_type)
    if msg_id != 0xFFFF8100 then
        return  (msg_type & 0x82) != 0x00
    else
        return (sd_type & 0x01) != 0x00
end
```

attacker are connected to an L2 switch that implements the proposed method. In this study, we implemented the proposed method assuming that only three ports were used for the L2 switch. These devices were personal computers running Ubuntu 22.04 and version 3.3.8, of vsomeip [8], a prototype SOME/IP protocol stack.

4.2. Anomaly Detection Rate

In this section, we evaluate the anomaly detection rate of the proposed method and demonstrate that it satisfies Requirement A and Requirement B shown in Section 3.2. The dataset used for evaluation is created by combining three factors: attack patterns, attack start timing, and SOME/IP(-SD) message types. The attack patterns are the two described in Section 3.1: normal and anomaly/man-in-the-middle patterns. Because the order of service provision by the server and service use by the client is determined by Condition B, shown in Section 3.2, only the timing of the attack by the attacker is varied.

The evaluation was performed using the following three patterns, in which communication started in the following order:

A) Start attack→Start of service provision→Use of service

B) Start of service provision→Start attack→Use of service

C) Start of service provision→Use of service→Start attack

The attacker sends six types of messages that can be sent by servers other than those listed in Tables 1 and 2. In a man-in-the-middle attack, an attacker sends an OFFER SERVICE message

Algorithm 3: Function to get the service ID

```
service_id: 16-bit integer indicating the service ID
function get_service_id (msg_id, sd_service_id)
    if msg_id != 0xFFFF8100 then
        service_id = (msg_id >> 16) & 0xFFFF
    else
        service_id = sd_service_id
    end
    return service_id
end
```

Algorithm 4:
Function to determine whether transfer is possible

```
srv_lst, clt_lst: 32-bit integer array listing servers and clients
state: 8-bit integer array indicating the communication
permission status assigned to each port
service_id: 16-bit integer indicating the service ID
function jdg_send (msg_id, sd_service_id)
    res: Boolean data indicating True (Permitted)/False (Not
    Permitted)
    res = True
    if state[service_id] == 0x02 then
        if sent_by_server(msg_id, msg_type, sd_type) then
            res = (((srv_lst[service_id] >> src) & 0x01) != 0)
                  and (((clt_lst[service_id] >> dst) & 0x01) !=0)
        else
            res = (((srv_lst[service_id] >> dst) & 0x01) != 0)
                  and (((clt_lst[service_id] >> src) & 0x01) !=0)
        end
    end
    return res
end
```

before sending the attack message. The anomaly detection rate of the proposed method was evaluated using 36 sets of data combinations.

Table 5 presents the experimental results using this dataset. Under normal and anomalous conditions, the attacker does not send an OFFER SERVICE. Therefore, the proposed method continues to determine that the attacker is not allowed to communicate. In man-in-the-middle attacks, communication patterns A) and B) produce the same result, but pattern C) produces a different result. In A) and B), the server and attacker send an OFFER SERVICE before the client starts using the service and the proposed method detects an anomaly. If the client begins using the service when an anomaly is detected, the proposed method sets communication-permitted to deny the client, preventing the client from using the service. In C), the server sends the OFFER SERVICE first. Thus, the proposed method sets the communication-permitted for the server. Because the client begins using the service before an anomaly is detected, the proposed method sets the communication-permitted for the client. Subsequently, when the attack begins, the proposed method

Table 5: Result of Anomaly Detection Rate

Attack Pattern	True	False Negative	False Positive
Normal and Anomaly	18 cases	0 case	0 case
Man-In-The-Middle	6 cases	0 case	12 cases

detects the anomaly and sets the communication permission to deny the attacker.

Based on the evaluation results in Table 5, there were no false-negative cases; therefore, the proposed method satisfied Requirement A. The only false-positive cases were communication patterns A) and B) in the man-in-the-middle attacks. These cases occur only before service use begins; thus, the proposed method satisfies Requirement B.

4.3. Processing Time and Resource Usage

In this section, we demonstrate that the proposed method satisfies Requirement C and Requirement D, as shown in Section 3.2, by evaluating the processing time and ROM/RAM usage based on the pseudocode of the proposed method shown in Section 3 and the performance of the processor used in the in-vehicle system. To demonstrate that the proposed method is feasible as an in-vehicle system, we established evaluation criteria based on the specifications of STMicro Stellar P series microcontrollers [17], which are being developed with the implementation of devices that connect multiple networks, such as Gateways and Zone Controllers, in an in-vehicle network and perform packet transfer. The Stellar P series microcontroller supports a 32-bit CPU operating at up to 1.6 GHz (Cortex-R52+ [18]), 2.3 to 8.2 MB of RAM, and 15.5 MB of ROM. While the CPU operating clock and memory size vary by product, this study employs a CPU operating clock of 1.6 GHz, RAM size of 2.3 MB, and ROM size of 15.5 MB as the baseline to demonstrate the feasibility of implementing the proposed method on microcontrollers used in automotive systems.

Table 6 lists the number of instructions contained in the pseudocode for the method described in Section 3. However, function arguments and return values are assumed to be stored in general-purpose registers; therefore, they are excluded from the number of memory accesses. Assuming that bit operations, arithmetic operations, comparison operations, branch instructions, and memory accesses can all be executed in one clock cycle, the number of instructions in the proposed method is 76. Therefore, the processing is completed in 76 clock cycles, or 0.00005[ms] at 1.6 GHz. As explained in Section 2.1, the acceptable communication delay in an in-vehicle network is 0.1[ms]. Therefore, the increase in processing time owing to the proposed method is within the acceptable range, and the proposed method meets Requirement C.

As shown in the pseudocode of the proposed method in Section 3, dedicated memory areas are required as global variables for the communication-permitted device list and the internal state of the server/client. The communication-permitted device list is a 32-bit variable, and the internal state is an array of 65,536 8-bit variables (the maximum number of service IDs supported by SOME/IP). Therefore, the required RAM size for the proposed method was 576 Kbytes per packet switch. However, the stack memory was required regardless of whether the proposed method was implemented; therefore, it was excluded from the evaluation. The global variables used in the proposed method had no initial values (initial value = 0); thus, the ROM area was not used. Therefore, considering only instruction size, if 76 instructions (32-bit) are stored in the ROM, the required ROM size for the proposed method is 304 bytes per packet switch. The usage of RAM and ROM was 24.5% and 0.002%, respectively, which fits within the capacity of the microcontroller considered in this study. Therefore, the proposed method satisfies Requirement D.

Table 6: Number of Instructions for Proposed Method

Function name	Bitwise/ quadrature/ comparison operations	bifurcation process	memory access
Ids	10 times	4 times	22 times
sent_by_server	5 times	1 time	4 times
get_service_id	3 times	1 time	4 times
jdg_send	8 times	2 times	12 times

Table 7: The Position of Proposed Method

Method	Anomaly detection rate	Availability	Processing load
IPSec,TLS/DTLS [6]	Excellent	Excellent	Poor
IDS for SOME/IP [7]	Good	Good	Excellent
Proposed method	Excellent	Good	Excellent

5. Consideration

In this section, we discuss the following three points based on the evaluation results:

- Comparison between the proposed method and previous studies.
- False positives during man-in-the-middle attacks.
- Application to large-scale networks with multiple switches.

Table 7 presents the results of the comparison between the proposed method and those of previous studies. The anomaly detection rate indicates the detection rate of spoofing attacks, which determines whether normal communication can be continued when a spoofing attack, including false positives, occurs, and the processing load indicates the CPU load and network load for anomaly detection. The existing IDS [7] cannot detect all spoofing attacks; however, the proposed method can detect all spoofing attacks. These methods show differences in detection rates for Anomaly Only attacks and Man-in-the-middle attacks where payloads and communication timing are normal. Similar to the IDS, the proposed method generates false positives, and there are cases where normal communication and spoofing attacks cannot be correctly distinguished, thereby preventing the use of the target service. This is a disadvantage compared with methods that employ IPSec or TLS/DTLS. By contrast, the proposed method was implemented using a simple algorithm with fewer than 100 operations, thereby resolving communication delay issues that are problematic in IPSec or TLS/DTLS. The proposed method is considered superior to existing methods in terms of availability. Subsequently, the impact of availability constraints was examined.

As shown in Section 4.2, false positives occur in man-in-the-middle attacks; if an attacker launches an attack before the client starts using the service, the client would not use the service, resulting in reduced availability. However, if the client detects an anomaly after starting to use the service, it can continue using the service. In automobiles, passenger safety is the top priority; therefore, availability during operation (i.e., while using the service) should be guaranteed. If the possibility of reduced availability is detected before operation (i.e., before starting to use the service), passengers should be notified of the anomaly to ensure safety and prevent operation. In the proposed method,

safety is prioritized over availability by preventing clients from accessing services if a potential spoofing attack is detected before service initiation.

Next, we discuss the application of the proposed method to networks that contain multiple switches, similar to actual in-vehicle systems. In SOME/IP-SD, the OFFER SERVICE is multicast to all devices using SOME/IP. Client devices utilize the target service after receiving the OFFER SERVICE. Consequently, attackers must also send the OFFER SERVICE to client devices. Regardless of the network configuration, packet switches connected to client devices receive both the legitimate and malicious OFFER SERVICE messages. By deploying the proposed method on all packet switches within the network, spoofing attacks can be detected in the same manner as in a single-switch configuration. However, since all packet switches must maintain the same internal state and communication-permitted device list, resource utilization becomes inefficient, raising concerns that the implementation cost may exceed that of a system monitored by a device [7]. If necessary, This method removes condition A and allows multiple instances per service ID. The designer must configure the number of instances allowed, and an anomaly will only be flagged if that number is exceeded.

6. Conclusion

In this study, we propose a mitigation method that detects server spoofing attacks during service discovery in SOME/IP networks packet switching in a network using and discards malicious messages. For the attack evaluation, 36 attack patterns, including man-in-the-middle attacks, were classified, constructed, and verified using actual devices. The proposed method can detect spoofing attacks in all patterns, and in 24 patterns, it can distinguish between normal communication and attacks, allowing for continuous service accessibility. Although it cannot distinguish among the remaining 12 patterns, security can be maintained by blocking the initial communication. The proposed method can detect attacks during the service discovery stage, thereby enabling safe communication blocking. In addition, it successfully mitigates man-in-the-middle attacks while remaining significantly lighter than encryption-based approaches.

Future work will explore integration with cryptographic mechanisms and complementary intrusion detection methods, as well as investigate applicability to other service-oriented protocols such as DDS. This research demonstrates that packet switches can respond to vulnerabilities to man-in-the-middle attacks by supporting dynamic service discovery with SOME/IP, contributing to the introduction of service-oriented architectures in automobiles.

Conflict of Interest

The authors declare no conflict of interest.

Acknowledgment

Part of this study was funded by the JSPS KAKENHI Grant Number 25K15126, Japan.

References

[1] AUTOSAR, “Specification of SOME/IP Transformer,” Requirement Specification AUTOSAR FO R22-11, 2022.

[2] AUTOSAR, “SOME/IP Service Discovery Protocol Specification,” Requirement Specification AUTOSAR FO R22-11, 2022.

[3] M. Cakir, T. Hackel, S. Reider, P. Meyer, F. Korf, T. C. Schmidt, “A QoS Aware Approach to Service-Oriented Communication in Future Automotive Networks,” 2019 IEEE Vehicular Networking Conference (VNC), 1-8, 2019, doi:10.1109/VNC48660.2019.9062794.

[4] J. R. Seyler, T. Streichert, M. Glass, N. Navet, J. Teich, “Formal analysis of the startup delay of SOME/IP service discovery,” 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE), 49-54, 2015, doi:10.7873/DATE.2015.0469.

[5] D. Zelle, T. Lauser, T. Kern, C. Krauss, “Analyzing and Securing SOME/IP Automotive Services with Formal and Practical Methods,” 16th International Conference on Availability, Reliability and Security (ARES '21), Article 8, 1-20, 2021, doi:10.1145/3465481.3465748.

[6] M. Iorio, M. Reineri, F. Risso, R. Sisto, F. Valenza, “Securing SOME/IP for In-Vehicle Service Protection,” in IEEE Transactions on Vehicular Technology, **69**(11), 13450-13466, 2020, doi:10.1109/TVT.2020.3028880.

[7] T. Koyama, M. Tanaka, A. Miyajima, S. Ukai, T. Sugashima, M. Egawa, “SOME/IP Intrusion Detection System Using Real-Time and Retroactive Anomaly Detection,” 2022 IEEE 95th Vehicular Technology Conference, 1-7, 2022, doi:10.1109/VTC2022-Spring54318.2022.9860928.

[8] Bayerische Motoren Werke Aktiengesellschaft, “vsomeip,” GitHub, https://github.com/COVESA/vsomeip, 2023.

[9] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, “Experimental Security Analysis of a Modern Automobile,” 2010 IEEE Symposium on Security and Privacy, 447-462, May 2010, doi:10.1109/SP.2010.34.

[10] C. Valasek, C. Miller, “Advanced CAN Injection Techniques for Vehicle Networks,” Black Hat USA 2016, Aug. 2016.

[11] K. Cho, K. G. Shin, “Error Handling of In-vehicle Networks Makes Them Vulnerable,” 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS2016), 1044-1055, Oct. 2016, doi:10.1145/2976749.2978302.

[12] K. Iehira, H. Inoue, K. Ishida, “Spoofing attack using bus-off attacks against a specific ECU of the CAN bus,” 2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC), 1-4, Jan. 2018, doi:10.1109/CCNC.2018.8319180.

[13] K. Iehira, H. Inoue, K. Ishida, “Feasibility assessment of denial-of-service attacks by analyzing SOME/IP-SD state transition models,” 2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC), 1732-1738, Jul. 2024, doi:10.1109/COMPSAC61105.2024.00273.

[14] AUTOSAR, “Specification of Secure Onboard Communication Protocol,” Requirement Specification AUTOSAR FO R22-11, 2022.

[15] SF.Lokman, A.T. Othman, MH. Abu-Bakar, “Intrusion detection system for automotive Controller Area Network (CAN) bus system: a review,” EURASIP Journal on Wireless Communications and Networking 2019, 1-17, Jul. 2019, doi:10.1186/s13638-019-1484-3.

[16] R. Kurachi, Y. Matsubara, H. Takada, N. Adachi, Y. Miyashita, S. Horihata, “CaCAN-centralized authentication system in CAN (controller area network),”14th Int. Conf. on Embedded Security in Cars (ESCAR 2014), 1-10, Apr. 2014.

[17] STmicro, “Data brief of SR6P7C8, SR6P7C4,” Microcontroller Specification, 2022.

[18] Arm, “Cortex-R52+,” Arm, https://developer.arm.com/Processors/Cortex-R52%20Plus, Aug. 2025 (Date of Access).