Journal of Automotive Software Engineering
Vol. **1(1)**, 2020, *pp.* 51–77
DOI: https://doi.org/10.2991/jase.d.200826.001; ISSN: 2589-2258
https://www.atlantis-press.com/journals/jase/

Security Patterns for Connected and Automated Automotive Systems†

Betty H. C. Cheng[1,*], Bradley Doherty[2,‡], Nicholas Polanco[1], Matthew Pasco[3,‡]

[1] *Michigan State University East Lansing, Michigan 48824, USA*
[2] *Technology Services Corporation Chicago, Illinois 60602, USA*
[3] *Microsoft Corporation Boulder, Colorado 80301, USA*

ARTICLE INFO

Article History

Received 04 Jan 2020
Accepted 06 Aug 2020

Keywords

Security patterns
Automotive
Cybersecurity

ABSTRACT

As automotive systems become increasingly sophisticated with numerous onboard features that support extensive inward and outward-facing communication, cybersecurity vulnerabilities are exposed. The relatively recent acknowledgement of automotive cybersecurity challenges has prompted numerous research efforts into developing techniques to handle individual threat vectors, the pathways and threat surfaces by which an attack can be realized. Security design patterns have been developed for many application domains (e.g., enterprise systems, networking systems, and distributed systems), but not much has been explored for automotive systems. This paper introduces a collection of security design patterns targeted for automotive cybersecurity needs. We leverage and extend the *de facto* standard template used to describe design patterns to include fields specific to the automotive domain and SAE J3061 cybersecurity development guidelines.

1. INTRODUCTION

As autonomous systems and features become further integrated into vehicles, software and electronics will account for over 50% of the design overhead [1]. However, the current software and its respective sub-systems have been demonstrated to be vulnerable to cyber threats in controlled testing environments [24]. These types of attacks demonstrate a clear increase in threat vectors and attack surfaces for autonomous features and systems, while illustrating the potential for impacting human safety [5]. In order to support systematic and reusable development practices focused on automotive cybersecurity needs, this paper introduces automotive-focused security design patterns to address cybersecurity vulnerabilities associated with inward facing (i.e., intra-vehicle) and outward-facing (i.e., inter-vehicle) communication.

The connectivity of modern vehicles to each other, third-party systems, and consumer devices increases the volume and pathways for possible attacks. The increasing automotive security vulnerabilities motivate the development of prevention, detection, and mitigation techniques that take into consideration automotive-specific constraints. Formalizing and abstracting common problem and solution strategies (i.e., designs) into design patterns facilitates rigorous development practices and promotes design reuse [6]. Common software security patterns enable developers to rigorously harden a system against security vulnerabilities [7]. While these security patterns, along with security solutions such as encryption and secure network protocols, have helped to harden many software systems, the unique architecture of automotive systems and constraints on system performance necessitate application-specific design strategies [8].

In order to leverage successful security patterns, while also addressing the automotive constraints, we have specialized existing security patterns to account for automotive systems' unique architecture and constraints. This framework for hardening security design and assessing security features can be applied to current automotive systems comprising numerous automated driver-assistance systems (ADASs) elements and extended to the emerging fully-autonomous vehicles. Automotive-specific security patterns enable developers to explicitly and rigorously address security as part of the development process for automotive systems in a proactive fashion. We introduce a template to describe the patterns, the template extends a previously developed security patterns template [9,10] to include fields, classification schemes, and information specific to the automotive domain [11]. In addition to the textual descriptions, we also include unified modeling language (UML) [12] class and sequence diagrams to respectively describe the structure and the behavior of a given solution strategy.

The automotive cybersecurity patterns leverage and extend previously developed security patterns to address vulnerabilities specific to automotive systems, with solution strategies that adhere to constraints and contexts relevant to automotive systems and relevant operating environments. As a preliminary step, we performed an

[*] *Corresponding author. Email: chengb@msu.edu*
[†] A short, preliminary version of this work was previously published in the MODELS Workshop for Modeling for Automotive Systems (MASE2019).
[‡] This author worked on this project while he was a student at MSU.

extensive review of the research literature (i.e., state of the art), technical reports from regulation and standards bodies (e.g., [11,13]), and obtained input from our industrial collaborators (i.e., state of the practice). From these sources, we identified threat vectors and attack surfaces, as well as techniques and/or solution strategies to address the respective vulnerabilities. We distinguish vulnerabilities and solutions that apply to onboard (inward) facing communication (e.g., wheel speed sensor and throttle position sensor signals sent to power train control module) versus outward-facing communication sub-systems (e.g., telematics units and navigation systems that communicate with cell towers and global positioning systems, respectively). Based on the state of the art and state of the practice reviews, as well as existing security patterns [7,9,10], we collated the information to define a set of automotive cybersecurity patterns. In order to facilitate their use, we provide a number of descriptors for each pattern that are commonly used in security development, including threat types using Microsoft's STRIDE framework [14], Viega and McGraw security principles [15] promoted by a given pattern, and automotive cybersecurity development guidelines as described in SAE J3061 [11].

The paper presents an ongoing effort with international industrial collaborators working in automotive cybersecurity, including both Tier 1 suppliers and original equipment manufacturers (OEMs) to develop a repository of security patterns.[1] These patterns are being incorporated into development practices that focus on security and its impact on safety for autonomous vehicles, involving different levels of autonomy. The remainder of this paper is organized as follows: Section 2 discusses related work in the area of security and design patterns. Section 3 overviews the vehicle's cyber-threat surfaces. Section 4 discusses related state of the art security solutions in automotive systems. Section 5 discusses the use of security patterns in automotive systems. Section 6 provides the collection of security patterns developed to date, organized according to those that address inward facing and outward-facing communication, respectively. Finally, Section 7 summarizes the work and outlines future work.

2. DESIGN PATTERNS

Software design patterns make it easier to reuse successful systems and architectures [16]. By using a known solution to a common problem, developers are able to benefit from successful designs and lessons learned from other developers. The original design patterns by Gamma *et al* [16] describe a design pattern with four essential parts: the name of the pattern, the problem addressed by the pattern, the solution strategy, and the consequences of the pattern. Following this approach, several hundred abstract patterns have been implemented for general use in software systems (e.g., [10,17,18]).

Security Patterns

While the design patterns developed by Gamma *et al.* are intentionally abstract with the goal of being applicable to a wide range of problems, domain-specific constraints need to be considered for the design. In the case of security patterns, threats to a system need to be monitored using specific security mechanisms for the specific context [7]. Following a similar format for describing a design pattern, security patterns are also given a descriptive name, the security problem addressed (e.g., a class of specific threats), a solution strategy to prevent, detect, and/or mitigate the security vulnerability, and a set of consequences [7].

Research in the field of security patterns has been active for almost two decades [17,19,20]. The focus has been on capturing higher level security mechanisms and organizing them into abstract patterns, which has guided cross-application security patterns [21,22]. The aforementioned work underscores the wide use of security patterns throughout different software systems, including distributed systems [23], enterprise systems [7,18], and, more recently, cloud computing systems [7,10], Security patterns apply to all phases of the software development process, such as requirements gathering, design, and implementation [20]. Within a collection of security patterns, several abstract patterns exist, indicating more specializations of patterns (e.g., [7,18,24]). A study by Ito *et al.* [17] surveys trends in security pattern research over the past decade.

Similar to general design patterns [16], security patterns have been described using specific fields to capture use-cases, consequences of design, etc. We leverage and extend a specific template developed by Wassermann and Cheng [9] and then refined by Konrad *et al.* [25] that captures important areas of a security pattern, namely: **Name**, **Alias**, **Intent**, **Applicability**, **Motivation**, **Structure**, **Consequences**, **Known Uses**, and **Related Patterns**.

3. AUTOMOTIVE CYBERSECURITY VULNERABILITIES

In this section, we overview *attack vectors* of automotive systems (i.e., interfaces or paths that an attacker uses to exploit a vulnerability) [13], such as an open cellular or Bluetooth network. Of particular interest, given its long-standing and prevalent use, is the internal communication system, the Controller Area Network (CAN) Bus [26] that acts as a medium for a variety of ECUs on a vehicle. We also overview other attack vectors according to the type of access needed to perpetrate an attack (e.g., physical, remote).

Automotive Communication

The CAN Bus is the major networking communication infrastructure used within an automotive system [27]. It relies on a broadcasting protocol that allows for any ECU attached to the CAN Bus to both send and receive messages as predefined ECU behavior, independent of the CAN Bus [28]. The CAN Bus is designed to be lightweight and fast, and consequently lacks common communication protocol features such as authentication or encryption. While data on a CAN Bus may carry identifying information to be interpreted by the intended receiving ECU, the CAN itself does not identify particular data frames, instead relying on a prioritization scheme of message arbitration to control access to the CAN Bus [29].

Potential security drawbacks to such a communication model include vulnerabilities posed by the lack of message authentication, such as spoofing and injection attacks. An injection attack makes it

[1] Our project sponsors involve international researchers and developers, as well as customers, who have provided feedback regarding the patterns.

possible for an adversary to manipulate the functionality of an ECU [30]. Because the CAN Bus has no default protocol for encrypting data frames on the network, the risk of compromising sensitive data poses another security concern [27,28,31,32]. Finally, the CAN message arbitration system can make it vulnerable to denial of service (DoS) attacks through an attacked ECU, such as the tire pressure monitoring system (TPMS) [3,33].

In addition to the CAN Bus, multiple technologies are increasingly being used to handle the growing demands of automotive communication. For example, automotive wired networking technologies include Local Interconnect Network (LIN), Media-Oriented System Transport (MOST), FlexRay, and automotive Ethernet; several recent surveys have overviewed the respective vulnerabilities and research challenges [28,34,35]. Rouf *et al.* also reviewed security vulnerabilities for in-vehicle wireless networking [33].

Vehicle Ad-hoc Networks (VANETs) make use of a number of wireless inter-vehicle networking, such as dedicated short range communications (DSRCs), long-term evolutionary vehicle (LTE-V), and Worldwide Interoperability for Microwave Access (WiMAX) [34]. With the increasing movement toward vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I), and vehicle-to-third-party devices (V2X) communication [36,37], security and privacy challenges correspondingly increase [34,38].

Degree of Access

While numerous automotive threat surfaces have been identified, we broadly categorize them by the type of access an attacker uses: direct (i.e., physical), indirect, and remote [39].

Direct access targets include the CAN Bus, the OBD-2 port, and the media/auxiliary port. The OBD-2 port provides a mechanism for sub-system diagnostics and requires an attacker to be physically plugged in as shown in experimental attacks [3,4,40]. The media player and auxiliary ports in a vehicle are often connected to the CAN Bus; and by physically inserting compromised files into the new media device, an attacker can subsequently inject (malware) code onto the CAN Bus [3].

Indirect access to a vehicle refers to compromising a device that can be used as a medium to attack the vehicle's sub-systems. Bluetooth networks are commonplace in modern vehicles, and accessing the vehicle's internal Bluetooth network is feasible through a connected, compromised device [3,4]. Similarly, if a diagnostic device has been compromised and then attached to the OBD-2 port of a vehicle, then the vehicle becomes vulnerable to a malicious attack. [3,4]

Finally, *remote attacks* have been demonstrated in several scenarios, where the attacker can be physically distant from the target. Through a vehicle's telematics system, such as General Motors' OnStar® or Hyundai's Blue Link®, "white hat hackers"[2] have shown the ability to attack using only an Internet connection [41]. With more outward-facing communication (e.g., V2V, V2I, and V2X), there is increasing risk for malicious actors to inject malware through VANETs [37,38,42,43].

4. SECURITY SOLUTIONS FOR AUTOMOTIVE SYSTEMS

Automotive security needs differ from security solutions used for traditional computing-based systems due to the specific performance constraints and requirements imposed by a vehicle's architecture and communication infrastructure [35,44]. Specifically, automotive systems face challenges with available communication protocols and the use of automotive-specific communication architecture (e.g., CAN Bus, FlexRay, LIN, MOST), limited communication resources, and stringent performance requirements for safety-critical functionality [8,45]. Despite these limitations, researchers have adapted existing security approaches to comply with automotive system's constraints, with a focus on authentication and encryption [46,47,48,49,50].

As mentioned previously, a common security issue for automotive systems is the inability to authenticate communications. Without the ability to determine what traffic is valid can expose a network to spoofing, DoS, and other types of attacks. This vulnerability can be addressed by using authentication protocols or intrusion detection systems (IDSs). However, the traditional approaches for these solutions are prohibitively expensive for an automotive network (e.g., with respect to performance constraints). A method for developing a more lightweight IDS is to base identification of connected ECUs on clock-based finger-printing with predictive algorithms to determine expected clock-skews of ECUs and subsequently detect spoofed messages [46]. Similar to the clock-based IDS, is the concept of a module-specific firewall. Rizvi *et al.* [51] propose a firewall-like program installed on individual ECUs to examine incoming packets in order to determine if the message should be released to the ECU for processing based on a set of rules and accepted subjects.

The VeCure framework [52] proposes another solution to address the lack of authentication in the CAN Bus protocol [52]. VeCure addresses message authentication by having ECUs transmit an authentication frame with the message being broadcast. At the same time, ECUs are segmented into trust groups where outward-facing ECUs are given low trust. The high trust group ECUs are given a key to verify message senders as a protection mechanism from potentially spoofed outward-facing nodes.

Vai *et al.* [48] examine a framework for both data encryption and message authentication in similarly constrained cyber physical systems. Using a co-processor that performs cryptographic primitives in hardware as an independent cryptographic module, using the least squared method (LSM), the researchers specify a communication bus security protocol where the cryptographic module behaves like an ordinary ECU attached to the bus. Each ECU on the communications channel has a digitally-signed configuration file containing rules for communication with other ECUs, as well as at startup. The LSM validates the configuration files and establishes secure communication channels for the ECUs using the session keys.

A similar use of cryptography can be applied to VANETs as a method of authentication and privacy preservation [49]. The LESPP system uses lightweight symmetric encryption and message authentication code (MAC) generation for message signing. It also contains a MAC re-generation for verification in order to authenticate senders. Moreover, privacy is protected as only the system's key management center can expose a vehicle's real identity.

[2] Hackers who look for security vulnerabilities for the sole purpose of hardening and improving the systems. https://us.norton.com

In order to prevent physical tampering to systems, the use of tamper-resistant designs can prevent malicious actors from making unsafe changes to a system[50]. The researchers propose the use of system-on-chip design and nondetachable connections to prevent ECU hardware alteration, or at least detect and provide evidence of tampering.

In addition to security solutions for specific attack vectors, several recent efforts have contributed surveys overviewing the range of automotive cyber security vulnerabilities [3,4,28,34,44,53,54], as well as general techniques to model the automotive security attacks and their mitigation.

Several of the general approaches involve techniques for security threat assessments and workflows to support security-aware development [55,56]. For example, SAHARA [57] combines hazard and risk analysis with security threat modeling for automotive systems. SAHARA has defined *SecL* to specify levels of security based on resources needed perpetrate an attack, degree of knowledge needed, and the threat of criticality (i.e., impact on safety). Security Abstraction Model (SAM) [58] is a security modeling language and methodology, customized for the automotive sector. Using the SAHARA SecL levels to classify security threats, SAM can be used to model security architectures and attack vectors [5], as well as provide a model-based approach to evaluate security requirements using the Common Vulnerability Scoring System (CVSS) criteria [59].

Ray *et al.* [44] recently overviewed the state of the practice of automotive security architectures and identified challenges to developing extensible automotive security architectures to balance security mitigation, real-time performance constraints, and runtime vulnerabilities.

5. SECURITY PATTERNS FOR AUTOMOTIVE SYSTEMS

This section overviews the template used to describe the automotive security patterns, as well as the security principles and automotive security development guidelines used to characterize the patterns. In addition to the template description, we overview the Viega and McGraw security principles [15], the STRIDE threat categories [14], and the SAE J3061 development guidelines for automotive security [11], all of which is incorporated into the pattern descriptions.

Automotive Security Pattern Template

The automotive security patterns are described in a template similar to that used by Konrad *et al.* [25].

- **Pattern Name** - The name and classification of a pattern as either behavioral or structural.
- **Intent** - How the pattern can be used and what security problem it addresses.
- **Motivation** - Background on the security problem, and a mechanism to solve, including examples and security principles that are related.
- **Properties** - STRIDE properties [14] that are addressed through this pattern.
- **Applicability** - Indicates the type of solution proposed: prevention, detection, and/or mitigation of a given security vulnerability.
- **Structure** - Includes the UML class diagram for the pattern and the related descriptions of participants and collaborations.
- **Behavior** - Includes a UML sequence diagram and descriptions of an illustrative scenario with the pattern.
- **Constraints** - Denotes constraints placed on the pattern implementation stemming from the limited resources and real-time constraints of the automotive system.
- **Consequences** - Discusses the effects of the pattern on the areas of Accountability, Confidentiality, Integrity, Availability, Performance, Cost, Manageability, and Usability following the template from Wasserman and Cheng [9].
- **Known Uses** - Discusses examples of this pattern's use in an automotive setting.
- **Related Patterns** - Any design patterns or security patterns that relate to the given pattern.

5.1. Security Properties Using STRIDE Threat Framework

STRIDE is a component of Microsoft's Security Development Lifecycle [14] framework used to categorize security threats to a system. Along with threat categorization, STRIDE incorporates a set of information technology properties that correspond to the given threats. Table 1 outlines the STRIDE threats, the corresponding information technology properties, and example security questions related to the properties. While STRIDE is applicable to general cybersecurity vulnerabilities, other efforts have also explored how

Table 1 STRIDE threats and properties [14].

Threat	Property	Example Security Question(s) Addressed
Spoofing	Authentication	Does the system use multi-factor authentication? Does the system enforce secure credential creation, use and maintenance principles?
Tampering	Integrity	Can the system detect and prevent parameter manipulation? Does the system protect against tampering and reverse engineering? Were secure software design principles followed during development, including third-party software?
Repudiation	Nonrepudiation	Does the system verify and log all user actions with attribution?
Information Disclosure	Confidentiality	Does the system follow standard encryption practices to secure connections?
Denial of Service	Availability	Was the system built and tested for high availability (e.g., fuzz testing and load testing)?
Elevation of Privilege	Authorization	Does the system support the management of all users and privileges?

STRIDE can be applied to the automotive domain, such as the SAHARA project that extends automotive hazard and risk analysis with STRIDE-based analysis [57].

Using the STRIDE framework in the pattern template assists developers in connecting the threats and their corresponding properties to patterns. Our industrial collaborators suggested the use of the STRIDE system in the pattern descriptions given their common use in industrial contexts. The STRIDE descriptors further facilitate the identification of appropriate patterns to use for a given problem/solution context.

5.2. Guiding Principles

Following a template similar to that used by Konrad *et al.* [9,25], several security principles are used to help motivate the problems that the security patterns are intending to solve. The following security principles from Viega and McGraw [15] are incorporated into the template.

- **V1 - Secure the Weakest Link** - Securing the most vulnerable piece of an application (e.g. network access points or interfaces).
- **V2 - Practice Defense in Depth** - Promote several layers of security throughout an application, which will ensure redundant security if one layer fails.
- **V3 - Fail Securely** - In the case of a system failure, ensure that unauthorized access to the system or information is not possible.
- **V4 - Follow the Principle of Least Privilege** - Give actors in a system the lowest degree of access/clearance necessary to carry out a given task.
- **V5 - Compartmentalize** - Keep sub-systems separate from each other so that a security failure in one will not compromise the others.
- **V6 - Keep it Simple** - Using unnecessarily complicated procedures or systems may lead to unforeseen security vulnerabilities.
- **V7 - Promote Privacy** - Ensure personal information and sensitive data is only accessible by those with ownership or given consent.
- **V8 - Remember that Hiding Secrets is Hard** - Security critical information such as passwords are not easily kept safe and secret. Therefore, designers must be wary of compromised accounts/sub-systems affecting the whole system.
- **V9 - Be Reluctant to Trust** - Sub-systems should not rely on other sub-systems being secure and must enforce security against all actors that interface with them.
- **V10 - Use Community Resources** - Community resources are often more secure than individual solutions due to the large number of developers.

We also used guiding principles for developing secure automotive systems defined in SAE Standard J3061 [11] to explicitly relate the security patterns to the industry standards for automotive systems. Our industrial collaborators also indicated that applying J3061 would potentially facilitate more industrial collaborations between development organizations. While the principles share common principles/objectives with those from Viega and McGraw [15], J3061 captures the principles in the specific context of the automotive system. They are as follows [11]:

- **J1 - Protect Personally Identifiable Information and Sensitive Data** - Ensure personal information and sensitive data is only accessible by those with ownership or given consent.
- **J2 - Use Principle of Least Privilege** - Give actors in a system the lowest degree of access/clearance unless necessary.
- **J3 - Apply Defense in Depth** - Promote several layers of security throughout an application. This will ensure redundant security if one layer fails.
- **J4 - Prohibit Software Changes that have not been Thoroughly Analyzed and Tested** - Preventing software from being changed without being thoroughly tested prevents unforeseen security flaws in a system.
- **J5 - Prevent Vehicle Owners from Making Unauthorized Changes** - Along with J4, changes to a vehicle that are not planned thoroughly may miss critical security flaws leading to system vulnerabilities.

5.3. Overview of Automotive Security Patterns Repository

The current repository of automotive security patterns has been drawn largely from automotive security solutions described in recent research literature, where we leverage the intent of previously developed security patterns [7,23]. These patterns have been developed with guidance from our industrial collaborators, who have developed additional patterns for internal use, as well as augmented their development process to include the use of our security patterns. Table 2 categorizes the automotive security patterns to date in the context of their applicability (i.e., prevention (Prev), detection (Det), and/or mitigation (Mit) of security vulnerabilities) and related security principles/guidelines as defined by Viega and McGraw (prefixed with "V") and by the SAE J3061 automotive cybersecurity guidelines (prefixed by "J"). Next, we include a brief description for each of the patterns in the repository, including citations to known realizations in the automotive domain.

- Authorization provides a structure that facilitates access control to resources [48].
- A Blacklist pattern intends to keep track of the traffic of potentially malicious addresses in a network. Nodes in the network use the Blacklist to block traffic originating from the malicious nodes [53,60].
- The DDoS Redundancy pattern is intended to make a resource or network more resilient to a distributed denial of service (DDoS) attack by providing redundant resources in case a resource becomes inundated with service requests [43,61].
- Pattern Firewall designs a structure that allows for network traffic to be filtered by a set of predefined rules to prevent malicious intrusion (e.g., securing ECUs individually [51]).

Table 2 | Applicability and Viega McGraw [48]/J3061 principles of select patterns [11].

Pattern Name	Applicability	V1	V2, J3	V3	V4, J2	V5	V6	V7, J1	V8	V9	V10	J4	J5
Authorization	Prev				X	X		X					
Blacklist	Mit/Prev		X			X				X			
DDoS Redundancy	Prev/Mit		X	X		X							
Firewall	Prev/Det	X			X					X			
Multi-Factor Authentication	Prev		X			X				X			
Multilevel Security	Prev/Mit				X	X		X	X	X			
Signature IDS	Prev/Mit/Det									X			
Symmetric Encryption	Prev							X		X			
Tamper Resistance	Prev/Det/Mit			X	X							X	X
Third-Party Validation	Det/Mit							X		X			

DDos, Distributed Denial of Service; IDS, Intrusion Detection Systems.

- The Multi-Factor Authentication pattern is used to provide a redundant security measure in authenticating an actor or a message. By enforcing an additional level of authentication, malicious actors can be prevented from attacking if a node or single credential is compromised [62].
- Multilevel Security is intended to provide a mechanism for handling access in a system with various security classification levels [52].
- Signature IDS provides a mechanism for detecting anomalies in network traffic by using a baseline characteristic of the (communication) traffic [46,63].
- Symmetric Encryption is used to encrypt messages (between vehicles and with infrastructure) so that only a sender and receiver can read the contents [49].
- A Tamper Resistance-based module deters unauthorized changing of a system by preventing alterations, or preserving evidence of alteration [50].
- The Third-Party Validation pattern is intended to provide validation of messages broadcasted in a given network. If a spoofed message is sent to a node and the receiving node cannot validate the message with a trusted node somewhere else in the network, then the receiving node can disregard the message [43].

6. REPOSITORY FOR AUTOMOTIVE SECURITY PATTERNS

This section presents the collection of the automotive security patterns, described according to the template from Section 5. Based on the development process and the industrial feedback, the patterns are organized according to those that address security vulnerabilities due to inward facing communication and those that are outward facing, respectively. Underlined Helvetica refers to pattern names, in-line italics refers to UML diagram elements (participants), and courier font refers to security principles/guidelines.

6.1. Security Patterns for Inward Facing Communication

The following patterns are more applicable to inward facing communication: *Authorization, Firewall, Multilevel Security*, and *Tamper-Resistant Module.*

6.1.1. Authorization

The Authorization pattern is a Structural Security Pattern.

Intent

Authorization provides a structure that facilitates access control to resources.

Motivation

Many systems need to restrict access to their resources according to certain criteria (e.g., a security policy) [7]. In automotive systems, inadequate authentication and authorization protocols have exposed the system to a variety of security exploits such as attacks on inter-vehicle networks [28], spoofing ECUs [30], and various other exploits [2,3]. In the case of Cranchelli *et al.* [48], the motivation for developing an authentication protocol for a communications bus, similar to a CAN bus, comes from the potential for ECUs and ECU messages to be spoofed. According to SAE standard J3061, **preventing unauthorized access to data** as well as controlling component privileges are key principles in developing automotive cybersecurity for a system[11]. According to Viega McGraw [15], the principle of being **reluctant to trust** improves the systems abilities to mitigate any security exploits a sub-system may have. This is particularly applicable to an automotive system, as many components come from different principle, and increasingly, vehicles have more communication with external entities.

Properties

The Authorization can be used to satisfy the Authentication property, and the Authorization property.

Applicability

Authorization is applicable to attack Prevention.

Structure

The Authorization structure of a system can be captured in terms of their relationships (Figure 1). Active entities are represented by instances of a *Subject* class and passive resources by instances of a *Protection Object* class. Between those main participants exists a relation that portrays which *Subject* is entitled to access certain objects. The properties of this relationship are organized in the association class *Rights*. The objects of this class define the type of access, transfer conditions, and constraints that restrict the use of *Rights*.

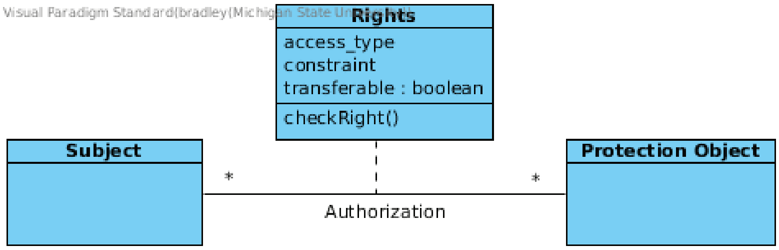

Figure 1 | Class diagram for Authentication Pattern.

Participants

The following section describes the participating classes in the pattern structure:

- *Protection Object*
  - Represents passive resources of the system that are accessed by *Subjects*.
- *Rights*
  - Defines the properties of the authorization rule between *Subject* and *Protection Object*.
  - The access type property describes which kind of access of the current *Rights* object grants. Commonly, this property holds values in the style of read, write, execute, . . .
  - The constraint property is a predicate that describes under which circumstances the current *Rights* object is valid and may grant a certain privilege.
  - The transferable property determines whether a right is transitive and its connected *Subject* may grant the right to other active entities.
- *Subject*
  - Represents active entities that need to access *Protected Objects* in accordance to their *Rights*.

Collaborations

Different *Subjects* in the system want to access *Protection Objects*. In order to make use of a certain resource they need to request access to it from the responsible controlling instance. This instance will check if an association class between the *Subject* and the *Protected Object* exists that justifies the required access request. Depending on this examination, access is granted or not.

Behavior

A *Subject* wishing to access a *Protected Object* will request access at a *Checkpoint Object* (Figure 2). The *Checkpoint* will request the rights pertaining to the *Subject*, and if approved, will forward the request to the *Protected Object*.

Constraints

In an automotive system, example constraints on an Authorization pattern include the performance of the authorization protocol and the amount of resources it may require. Because a vehicle is limited in the resources it has available (e.g., CPU and memory), there is a need to perform authorization efficiently. In addition, the need for real-time execution; where multiple *Subjects* may be sharing access to a *Protected Object*, delay is often unacceptable.

Consequences

Table 3 describes consequences in implementing the pattern.

Known Uses

As mentioned in the motivation section, Cranchelli *et al.* [48] has developed an authorization module for a shared communication bus to authenticate and manage authorized communication between ECUs on the bus [48]. It is able to authenticate ECUs on the bus at system start-up, and by using digital signatures, authorize ECUs' interactions with one another.

Related Security Patterns

The Check Point pattern can be used to examine requests by using the structure of the Authorization pattern. The RBAC or Roles pattern is an extended version of this pattern.

Supported Principles

The Authorization pattern should be used in combination with Principle 4 (**principle of least privilege**) to assign the user's rights. Furthermore, Authorization can be used to **compartmentalize** (Principle 5) the system and reduce the impact of a security breach. For example, an attacker that illegally manages to authenticate as user A, has only the rights user A would have. Given a restrictive security policy, the enforcement of access rights **promotes privacy** (Principle 7).

6.1.2. Firewall

6.1.3. Pattern name and classification

Firewall is a structural security pattern.

Intent

A firewall allows for network traffic to be filtered by a set of predefined rules to prevent malicious intrusion.

Motivation

Application firewalls allow for control protocols to be placed on traffic accessing specific services in a sub-system. This may be appropriate when system level protocols are not able to capture the requirements of application level control [7]. In an automotive network, unfiltered traffic can result in attacks on system components

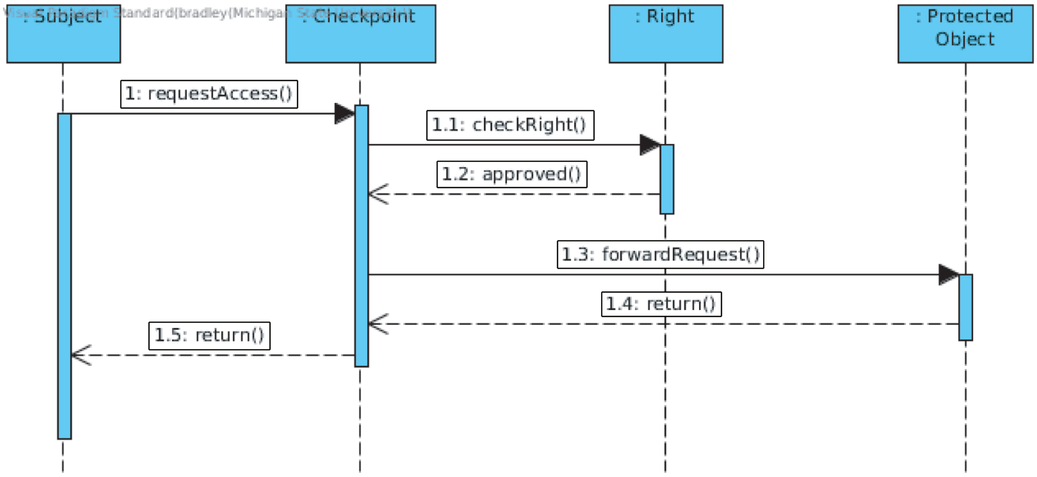

Figure 2 Sequence diagram for Authentication Pattern.

Table 3 Consequences for Authorization Pattern.

Accountability:	Not Affected
Confidentiality:	Can be improved by specification and rigorous enforcement of rights.
Integrity:	Can be improved by specification and rigorous enforcement of rights.
Availability:	Can be improved by specification and rigorous enforcement of rights.
Performance:	The system performance might be derogated by extensive right checks and the evaluation of the rights constraint predicates. This issue is of particular concern in a real-time environment.
Cost:	Cost is not significantly affected by the application of this pattern. May require additional hardware to perform authentication.
Manageability:	Modify to account for automotive systems, e.g., ECUs, communication privileges, etc., instead of traditional computing terminology.
Usability:	If the access rights are checked extensively the user might recognize a loss of performance. Authorization may limit utilization of shared resources, impede safety-critical sub-systems.

from malicious traffic, as well as from compromised system components. On a CAN Bus, e.g., externally facing ECUs such as a media player can be compromised and as a result present a threat to other ECUs on the CAN Bus. Rizvi *et al.* [51] specifically addressed this scenario in their work. As it pertains to SAE J3061, the use of Firewall promotes the principle of **practicing defense in depth**, by securing ECUs on an individual basis [11]. This pattern also promotes the Viega and Mcgraw principles of **Reluctance to Trust** and **Compartmentalization** [15].

Properties

The Firewall can be used to satisfy the Authentication property, the Authorization property, the Integrity property, and the Nonrepudiation property.

Applicability

The pattern is applicable to attack Prevention and attack Detection.

Structure

A *Client* accesses a *Service* through a *Firewall* (Figure 3). The *Firewall* references the rules which are represented by a *Policy Object* and aggregated in the *Policy Base Object*. The *Firewall* consists of the *Policy Authorization Point*, where the rules and identities for the *Firewall* are centrally stored. The *Firewall* also consists of several *Policy Enforcement Points* which check accesses to a particular module. The data flowing through the *Firewall* is checked by the *Content Inspector*.

Participants

- *Client*
  - Actor requesting access through the *Firewall*.
- *Application*
  - Module protected by *Firewall*, composed of *Services*.
- *Firewall*
  - Enforcement and access point through which the *Client* accesses an *Application*'s *Services*.
- *Policy Authorization Point*
  - Central collection of access policies and identities used by the *Firewall*.
- *Identity Base*
  - Collection of authorized *Identities*.
- *Policy Base*
  - Collection of authorized *Policies*.
- *Policy Enforcement Point*
  - Checks access to *Applications*
- *Content Inspector*
  - Checks data flowing through *Firewall*

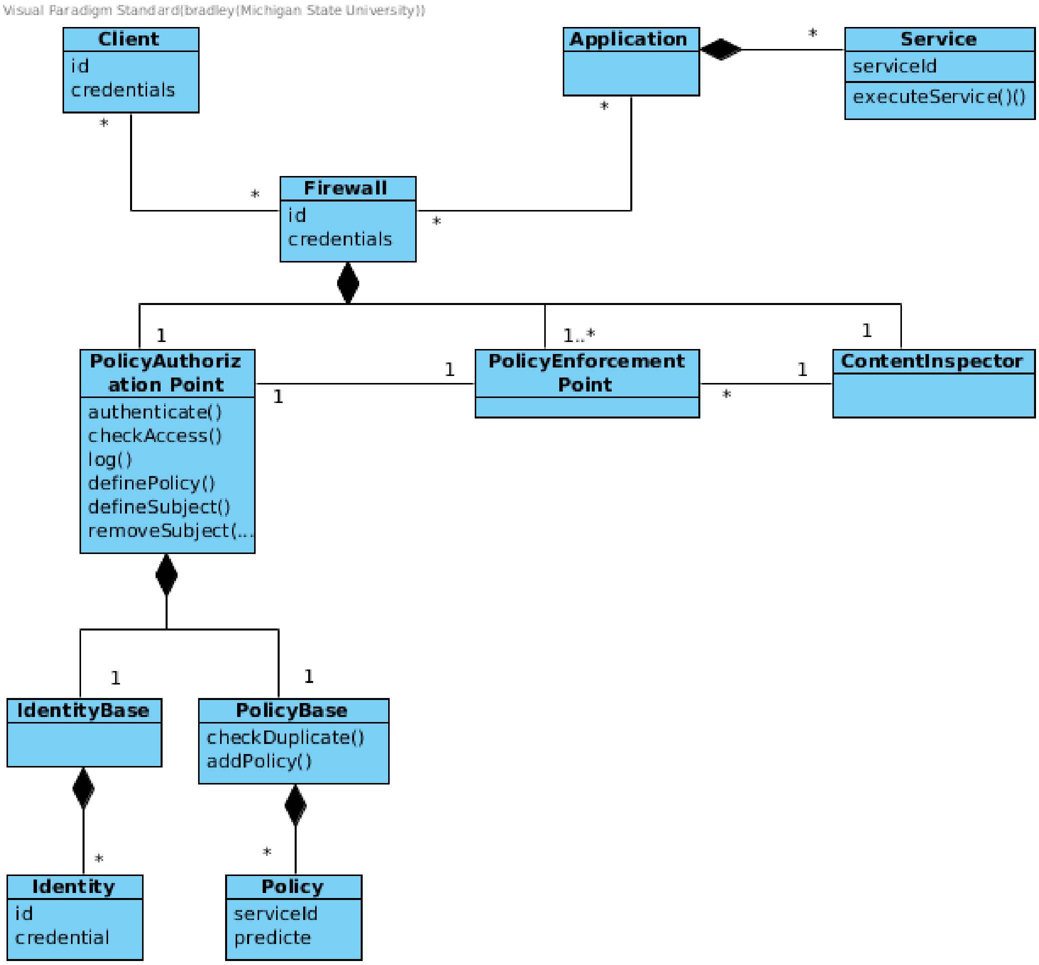

Figure 3 Class diagram for Firewall Pattern.

Collaborations

A *Firewall* can act as a mediation point for many *Applications* and many *Clients*. An *Application* is composed of *Services*. A *Firewall* is composed of a *Policy Authorization Point*, possibly several *Policy Enforcement Points*, and a *Content Inspector*. The *Policy Authorization Point* is composed of an *Identity Base* and a *Policy Base* which are respectively composed of *Identities* and *Policies*.

Behavior

A *Client* begins by requesting a *Service* from an *Application* through the *Firewall* (Figure 4). The *Firewall* forwards the request to the *Policy Enforcement Point* where a *Policy Authorization Point* is asked to check the *Client*'s access privileges by looking up the *Identity* and *Policies* in the *Identity Base* and *Policy Base,* respectively. If access is granted, then content of the request is checked by the *Content Inspector* and finally sent to the *Application* to service the request.

Constraints

In an automotive system, limited and shared resources limit the extent of Firewall usage. Given the constraints of a real-time environment, overhead incurred by processing messages may affect critical response time. Finally, given a variety of different ECU vendors, there can be difficulty in developing a uniform Firewall system and enforcing the practice.

Consequences

See Table 4.

Known Uses

As described previously, a solution to systemic risk of attack on the CAN network given the compromising of an ECU was proposed by Rizvi *et al.* Here, firewall-like programs installed on individual ECUs examine incoming packets to determine if the message should pass to the ECU for processing [51].

Supported Principles

Firewall pattern promotes Principle 1 (**Secure the Weakest Link**), Principle 4 (**Least Privilege**), and Principle 9 (**Be Reluctant to Trust**).

Related Security Patterns

The pattern is related Authorization and Role-Based Access [7].

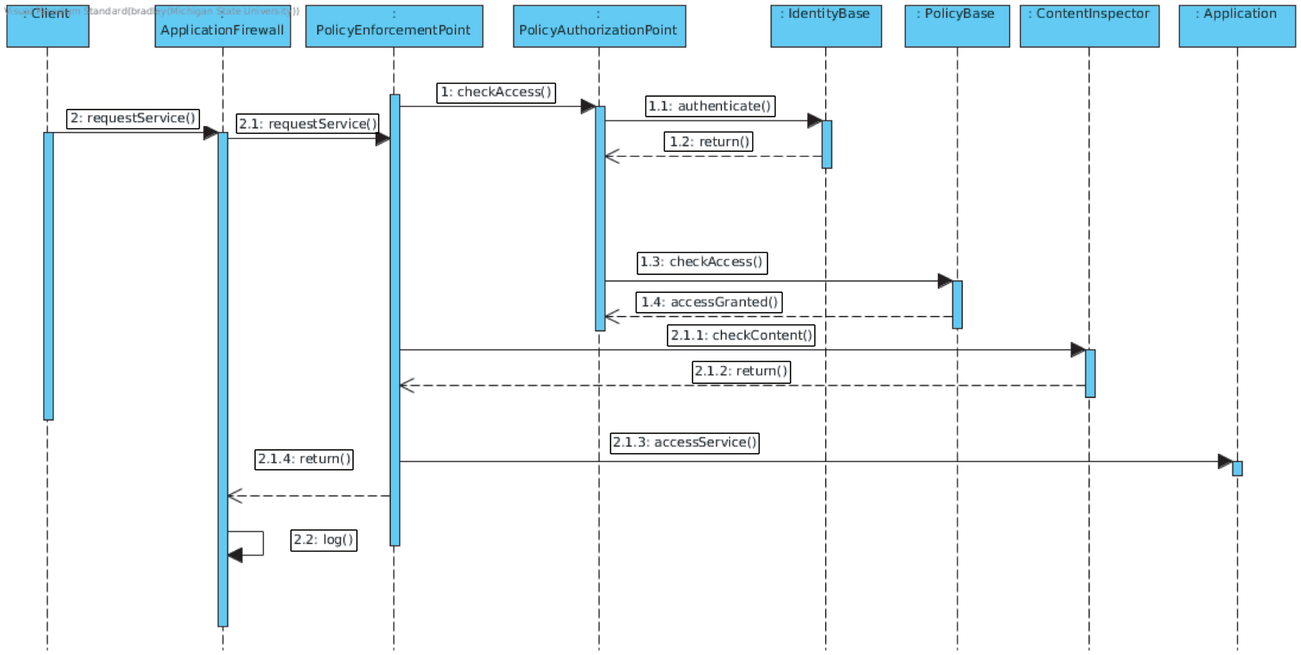

Figure 4 Sequence diagram for Firewall Pattern.

Table 4 Consequences of firewall pattern.

Accountability:	Accountability Is Not Affected
Confidentiality:	Confidentiality can be better protected by controlling access to a particular resource.
Integrity:	Integrity of the system is better maintained by enforcing defense on an individual-module basis.
Performance:	Performance of the system may be affected by overhead of validating senders through the Firewall.
Cost:	May require additional hardware.
Manageability:	May require additional management overhead for managing individual ECU compliance and uniformity of a protocol across vendors.
Usability:	Usability of system resources may be affected by overhead of the Firewall.

6.1.4. Multilevel security

Multilevel Security pattern is a structural pattern.

Intent

This pattern is intended to provide a mechanism for handling access in a system with various security classification levels.

Motivation

In many systems integrity and confidentiality of data need to be guaranteed [7]. In an automotive environment, certain resources are critical to system and user safety (e.g., ABS braking and power steering) and must be given a higher degree of access control. Moreover, on an automotive communication network, e.g., some ECUs might be externally facing (i.e., communicating with external entities) and are consequently more susceptible to compromises. By segmenting access by degrees of trust in a multilevel security hierarchy, system critical resources can be given this higher degree of access control, and resources more susceptible to compromise can be given limited trust, thus ensuring a higher degree of overall security in the system. In the work by Wang and Sawhney [52], the researchers address the problem of different levels of access control, differentiating between externally facing ECUs and those that manage more system critical components [52]. According to SAE J3061 Standards [11], key cybersecurity principles include protecting sensitive data, using the principle of least privilege, and applying defense in depth. A Multilevel Security pattern achieves the aforementioned objectives by securing sensitive resources from less trusted entities, giving individual system components privileges to use other system resources only if they are trusted, and finally ensuring that interactions between different components is authenticated on a case-to-case basis.

Properties

The Multilevel Security can be used to satisfy the Authorization property, and the Confidentiality property.

Applicability

The pattern is applicable to attack Prevention, and attack Mitigation.

Structure

To represent the Multilevel Security pattern, for each *Subject*, there exists an instance of the *Subject Classification* class and for each *Object* an instance of the *Object Classification* class (Figure 5). These instances are used to aggregate a *Subject*'s and *Object*'s respective security levels and categories.

Participants

- *Object Category*
  - Defines a category to which an *Object* belongs.
- *Object Classification*
  - Determines the security classification of an *Object* (passive resource that is accessed by *Subjects*).
  - The *Object*'s security classification is represented by a set of *Object Categories* and *Object Levels*.

- *Object Level*
  - Defines the security level of an *Object*.
- *Subject Category*
  - Defines a category to which a *Subject* has access.
- *Subject Classification*
  - Determines the security classification of a *Subject* (active entity that accesses *Objects*).
  - The Subject's security classification is represented by a set of *Subject Categories* and *Subject Levels*.
- *Subject Level*
  - Defines the security level of the *Subject*.

Collaborations

The classes *Subject Classification* and *Object Classification* contain a set of *Category* and *Level* classes. This set determines the security classification of the respective *Object*. Access will be granted if the *Classification* of the requesting *Subject* dominates the Classification of the *Object*. The Behavior section explains in detail how to determine whether one classification dominates a second one.

Behavior

In the Multilevel Security pattern, the *Subject* requesting access to a resource will request access at the *Checkpoint* (Figure 6). The *Checkpoint* then obtains the *Classification* of both the *Object* and the requesting *Subject*. If the requesting *Subject*'s classification dominates the *Object*, i.e., has an equal or greater security level, then the request is forwarded.

Constraints

In a real-time environment, such as that of an automotive system, verification of clearance must be a quick and efficient process in order to minimize delays in access to resources.

Consequences

Table 5 describes the consequences of using the Multilevel Security pattern.

Known Uses

As mentioned previously, the VeCure system seeks to segregate access to certain ECUs by differential degrees of trust. Using multi-tier trust circles, the system is implemented to keep core systems safe from external facing ECUs on a CAN bus network [52].

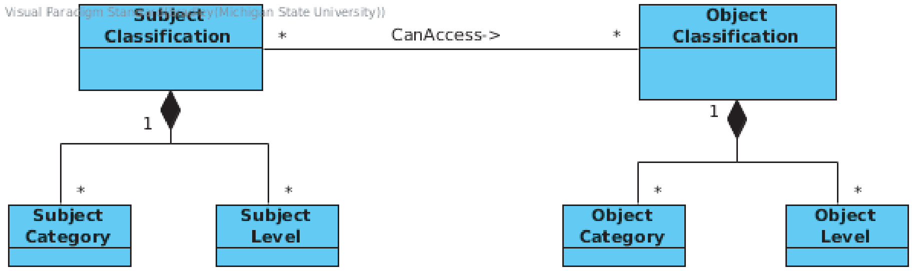

Figure 5 Class diagram for Multilevel Security Pattern.

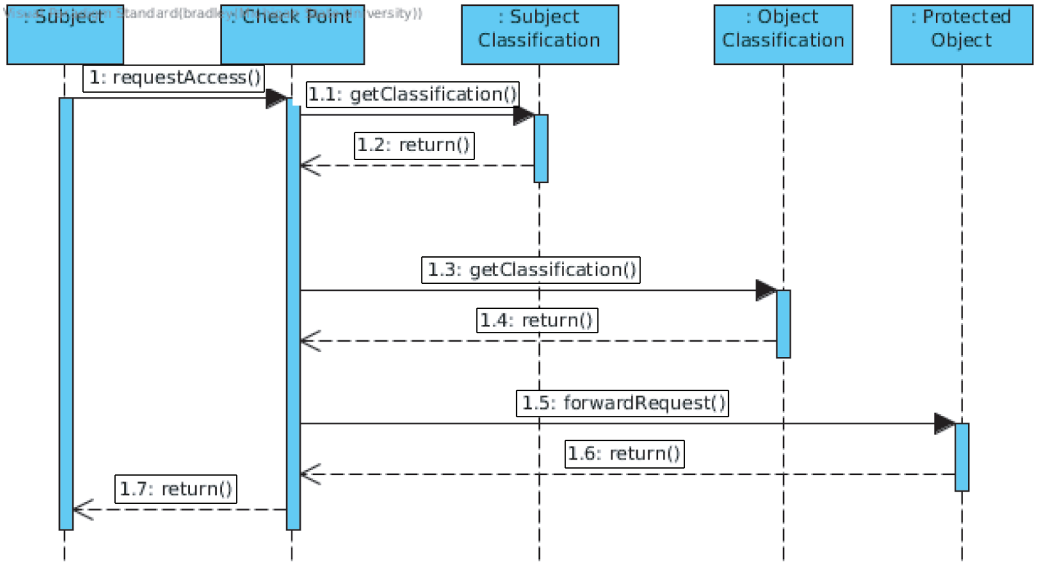

Figure 6 Sequence diagram for Multilevel Security Pattern.

Table 5 Consequences of multi-level security pattern.

Accountability:	Not Affected
Confidentiality:	User access is controlled according to the rules that are stated in the Bell-LaPadula [4] model in order to guarantee confidentiality of data.
Integrity:	Biba's model [28] and its rules establish integrity. Idea of circles of trust in data received.
Availability:	Not affected.
Performance:	The evaluation of access rights can derogate performance.
Cost:	Establishing a multilevel security system can be costly. All subjects and objects need to be classified in certain sensitivity levels. May require additional hardware support.
Manageability:	Not affected.
Usability:	Subjects may be limited in what sub-systems they communicate with.

For example, messages from the externally facing OBD-II port and Telematics systems are considered to be in a lower trust group than the nonexternally facing ECUs of the higher trust group, such as the ABS braking system [52].

Related Security Patterns

The Check Point pattern may be used to enforce the Multilevel Security structure, which can be provided in Session object.

Supported Principles

The Multilevel Security pattern and its mechanism of granting access to subjects implements the **principle of least privilege** (Principle 4). Furthermore, it facilitates **compartmentalization** (Principle 5) and reduces the impact of a security breach. **Promoting privacy** (Principle 7), **Hiding secrets is hard** (Principle 8), and **Reluctance to trust** (Principle 9) are key ideas of the Multilevel Security pattern.

6.1.5. Signature-based IDS

Signature-based IDS is a structural-based security pattern.

Intent

The pattern provides a mechanism for detecting anomalies in network traffic by using a baseline characteristic of the traffic.

Motivation

On an open network, there is a need for detecting malicious traffic as soon as it occurs to prevent damage to a system [7]. In an automotive network such as that used with the CAN, there is no mechanism for detecting malicious traffic in the protocol, such as a spoofed ECU. By detecting deviations from a baseline behavior, such malicious activity can be detected. SAE J3061 emphasizes the need for comprehensive responses to cybersecurity incidents [11]. A component of this response is the ability to detect intrusions and respond quickly. The pattern supports the principle of Reluctance to Trust [15] as authenticity is always verified before a message is allowed to pass.

Properties

The Signature-based IDS can be used to satisfy the Authorization property, and the Integrity property.

Applicability

The Signature-based IDS is applicable to attack Detection, attack Prevention, and attack Mitigation.

Structure

The Signature-based IDS intercepts the access request for a service (Figure 7). The *Event Processor* processes the request to parse the relevant *Signature Information*. The *Attack Detector* then tries to match the *Signature Information* against the known signature information to determine if the *Signature* is known. If the *Signature* is unknown, then the appropriate *Response* is issued [7].

Participants

- *IDS:* Intercepts messages and performs proper action given a *Response*
- *Event Processor:* Obtains *Signature* information from message
- *Attack Detector:* Checks *Signature* against known signatures
- *Signature Information:* Has ID of known actor and the corresponding *Signature*
- *Signature:* Identifying characteristic of actor
- *Response:* Communicates signature results back to *IDS* for message handling

Collaborations

The *IDS* has a one-to-one relationship with the *Event Processor* and forwards requests to be processed. The *Event Processor* collaborates in a one-to-one relationship with the *Attack Detector* that obtains processed data from the *Event Processor*. The *Attack Detector* includes the *Signature Information* of known entities. The *Attack Detector* has a one-to-one relationship with the *Response* object that formulates the response given the results of the *Attack Detector*. Finally the *IDS* has a one-to-one relationship with the *Response* object that forwards the correct response to the IDS.

Behavior

A message is sent from a source to an intended destination node (Figure 8). The Signature-based IDS intercepts the message and determines if the *Signature* is known to the system. If the sender is unknown, then an appropriate *Response* is raised [7].

Constraints

As an IDS analyzes traffic before the messages are sent to the intended receiver, there is overhead incurred when using the pattern. The solution provided by Cho and Shin [46] utilizes a predictive algorithm implemented in hardware to decrease the processing time in order to minimize the overhead.

Consequences

Table 6 describes the consequences of using the Signature-based IDS pattern.

Known Uses

An example of a lightweight IDS for CAN bus was proposed by Cho and Shin [46]. Here, the system used clock-based finger-printing and predictive algorithms to determine expected clock-skews of ECUs, thus enabling the detection of spoofed messages and authentic ones.

Related Security Patterns

The pattern is a specialization of the Abstract IDS pattern and is related to the Firewall pattern, another access control pattern for networks [7].

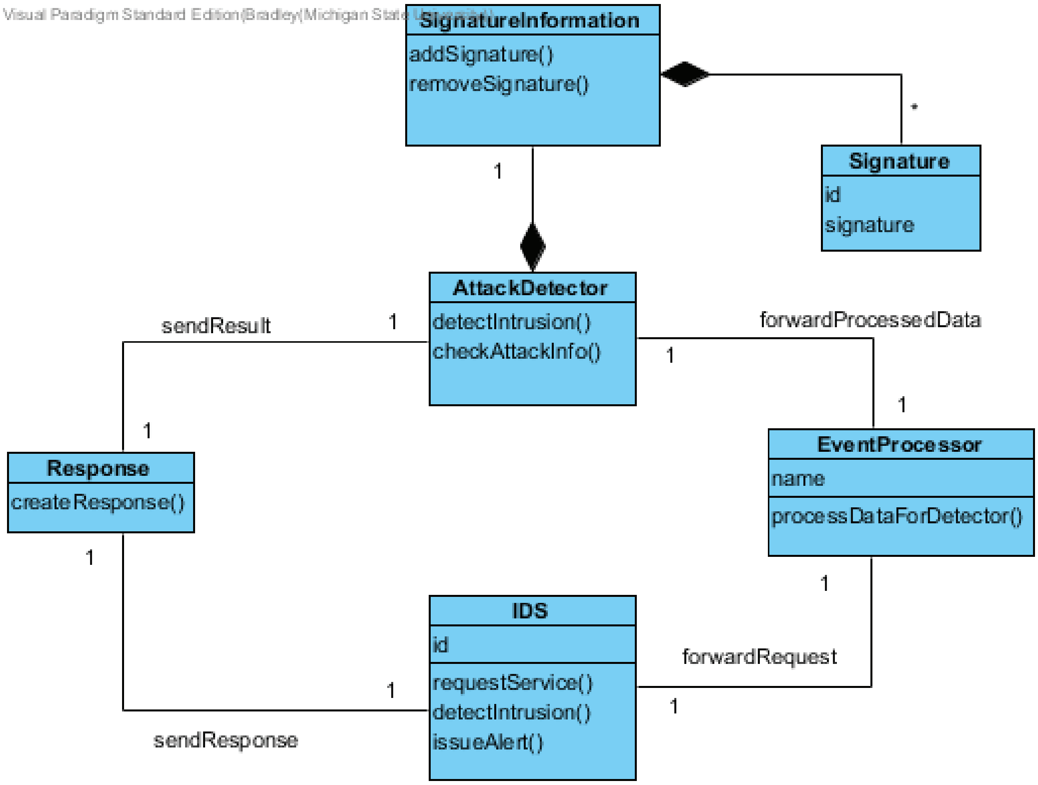

Figure 7 Class diagram for Signature Intrusion Detection Systems (IDS) Pattern.

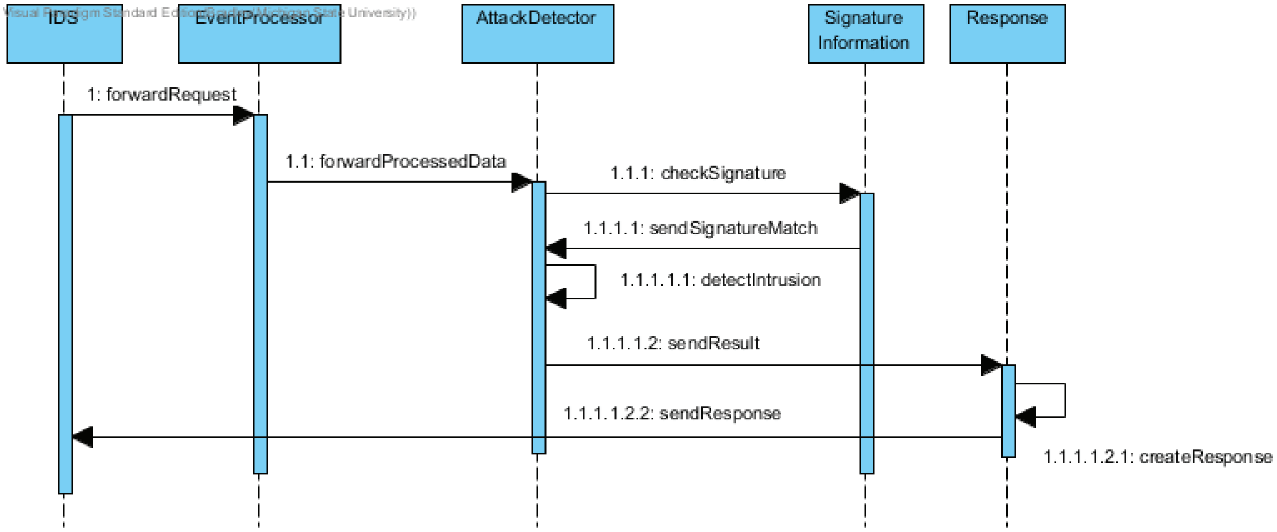
Visual Paradigm Standard Edition(Bradley(Michigan State University))
IDS
EventProcessor
AttackDetector
Signature Information
Response
1: forwardRequest
1.1: forwardProcessedData
1.1.1: checkSignature
1.1.1.1: sendSignatureMatch
1.1.1.1.1: detectIntrusion
1.1.1.1.2: sendResult
1.1.1.1.2.2: sendResponse
1.1.1.1.2.1: createResponse

Figure 8 Sequence diagram for Signature Intrusion Detection Systems (IDS) Pattern.

Supported Principles

Principals supported by the Signature-based IDS include Principle 9 (Be Reluctant to Trust), as the IDS intercepts messages to verify the identity of the sender before allowing it to access protected resources, and Principle 3 (Fail Securely).

6.1.6. Tamper resistance

Tamper Resistance is a structural based security pattern.

Intent

A Tamper Resistance-based module deters unauthorized changing of a system by preventing alterations, or preserving evidence of alteration.

Motivation

In a given system, altering certain modules may lead to unpredictable and potentially dangerous system behavior. In an automotive system, this can result in dangerous vehicle functioning and increased susceptibility to malicious attacks. Work by Wolf and Gendrullis notes the potentially dangerous consequences of alteration to a standard Hardware Security Module [50]. Unchecked alteration of the module can lead to security vulnerabilities for any other vehicle sub-system that relies on the module for cryptography services. As it pertains to SAE J3061, the use of tamper-resistant design can promote **the protection of sensitive data**, and **prevention of unauthorized changes to a vehicle** [11]. The pattern also promotes the principles of **failing securely** [15].

Table 6 Consequences of signature-based IDS pattern.

Accountability:	The Signature-Based IDS Allows a System to Be Accountable for the Authenticity of the Traffic Sent on Its Network
Confidentiality:	Not addressed.
Integrity:	By preventing unrecognized agents from communicating on a network, the Signature-based IDS protects the network from misuse, and the agents attached to the network from attack
Availability:	The Signature-based IDS may prevent availability on a network as overhead on authentication may lead to under use.
Performance:	The system may take a performance hit in validating the message signatures as it may require significant overhead.
Cost:	Additional hardware to process signatures may incur a cost.
Manageability:	Allows a system to more easily manage the actors that may utilize a network.
Usability:	The usability may decrease as only a small subset of actors may be known to a network.

IDS, Intrusion Detection Systems.

Properties

The Tamper-Resistant Modules can be used to satisfy the Integrity Property, the Nonrepudiation property, and the Authorization property.

Applicability

The pattern is applicable to attack Prevention, attack Mitigation, and attack Detection.

Structure

The Tamper-Resistant Modules structure of a system can be captured in terms of their relationships (Figure 9). The pattern is understood as the *Interface*, between the *Subject* and the *Tamper-Resistant Object*. The *Interface* has a *Working State* that abstractly describes the untampered status of the *Tamper-Resistant Object*. If the *Tamper-Resistant Object* is altered, the *Interface* will fail to interact with the *Tamper-Resistant Object* in the way the *Working State* predicts, causing the *Interface* to initiate a tamper response, whether it be disabling itself or changing to an immutable tamper state.

Participants

The following section describes the participating classes in the pattern structure.

- *Subject*
  - An actor requesting to access a *Tamper-Resistant Object*.
- *Interface*
  - Medium for communication between the *Tamper-Resistant Object* and a *Subject*.
- *Working State*
  - Abstractly, the *Interface's* working model of using the *Tamper-Resistant Object* as it was designed.

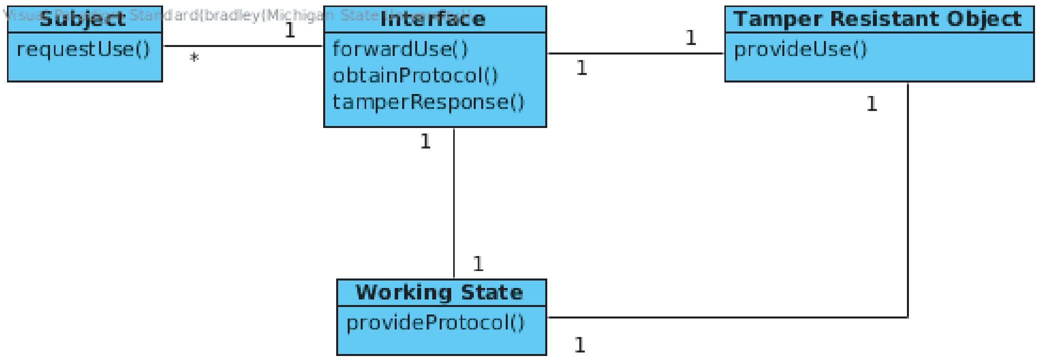

Figure 9 Class diagram for Tamper-Resistant Modules Pattern.

Collaborations

A given *Subject* will attempt to communicate with a *Tamper-Resistant Object* through an *Interface*. Consequently, there is an association between both the *Subject* and the *Interface* and the *Tamper-Resistant Object* and the *Interface*. The *Interface* has an association with the *Working State*. The *Tamper-Resistant Object* has an associated *Working State*.

Behavior

Figure 10 depicts a scenario where a *Subject* interacts with the *Tamper-Resistant Object*. The *Subject* will send the request, prompting the *Interface* to forward the request to the *Tamper-Resistant Object* on behalf of the *Subject*. To do this, the *Interface* will consult the *Working State* object for the proper protocol of interaction. When the *Interface* fails to interact with the *Tamper-Resistant Object* as tampering has rendered the *Tamper-Resistant Object* different from its *Working State*, the *Interface* will execute its tamper response.

Constraints

In an automotive system, additional hardware, like that required to enforce Tamper-Resistant Modules, can be expensive and take valuable space in the system.

Consequences

Consequences for applying this pattern are given in Table 7.

Known Uses

As described by Wolf and Gendrullis [50], a tamper-resistant design achieved by a system-on-chip non-detachable connection with ECU hardware promotes tamper resistance or at least evidence of tampering.

Supported Principles

Tamper-Resistant Modules pattern promotes Principle 3 (**Fail Securely**), Principle 4 (**Least Privilege**), J3061 principle 4 (**Prohibit Untested Software Changes**), and J3061 principle 5 (**Prevent Vehicle Owners from Making Unauthorized Changes**).

Related Security Patterns

Tamper-Resistant Modules is related to the Secure Logger described by Fernandez [7].

6.2. Security Patterns for Outward-Facing Communication

In contrast, the following patterns are intended to secure outward-facing communication. *Blacklist, DDoS Redundancy, Multi-Factor Authentication, Symmetric Encryption*, and *Third-Party Validation*.

6.2.1. Blacklist

Blacklistis a structural-based security pattern.

Intent

A Blacklist intends to monitor the traffic of potentially malicious addresses in a network. Nodes in the network use the Blacklist to block traffic originating from the malicious nodes.

Motivation

Communication with external entities is becoming increasingly integrated into modern automotive systems. Today's cars communicate with smart infrastructure, receive over the air updates from manufacturers, and exchange information with other vehicles on

Table 7 Consequences of Tamper-Resistant Modules pattern.

Accountability:	Tamper-Resistant Modules Can Increase Accountability of a Sub-system
Confidentiality:	N/A
Integrity:	The integrity of a system is improved through the employment of tamper resistance.
Performance:	If implemented in the design of a given module, performance costs should not be a concern.
Cost:	May require additional hardware.
Manageability:	May be more difficult to update a module if additional steps are required to circumvent the tamper protection.
Usability:	Usability should not be affected.

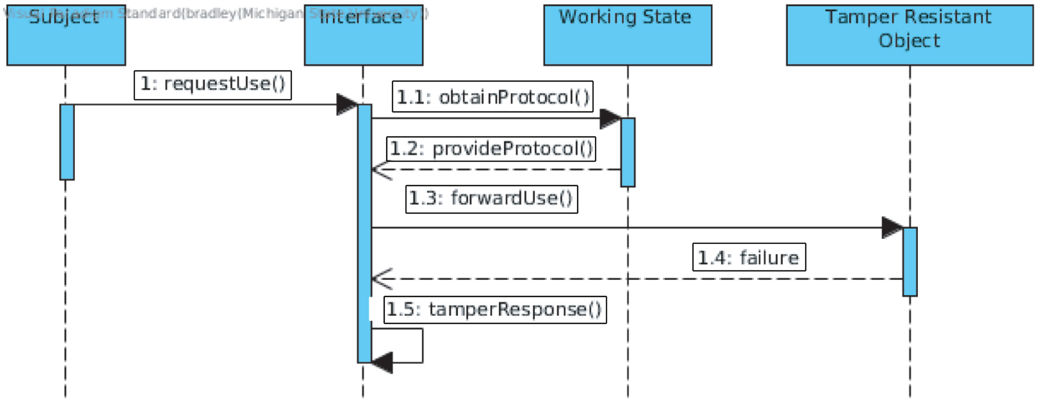

Figure 10 Sequence diagram for Tamper-Resistant Modules Pattern.

the road. While these technologies allow for improved performance and user experience, the increased reliance on networks leaves a vehicle susceptible to attack from a malicious user on the network. The inability to identify compromised nodes poses a security risk to an automotive system. VANETs allow vehicles and smart infrastructure to share information about current driving conditions within a specific broadcast range. Due to the routing protocols upon which VANETs rely, a malicious node can join a VANET and proceed to attack the network by intercepting the communication between vehicles, dropping packets, and changing or fabricating packets [64]. Applications such as the Post Crash Notification [65] can be deployed on VANETs to detect an attack such as injections of false messages that can lead to potentially dangerous responses by the vehicles on the network.

Properties

The Blacklist can be used to satisfy the Authentication property, the Authorization property, and the Nonrepudiation property.

Applicability

A Blacklist is applicable to the Prevention and Mitigation of an attack.

Structure

The Blacklist structure of a system can be captured in terms of their relationships (Figure 11). An entity sending a message is captured by a *Client* object. The entity secured with a Blacklist is represented as a *Service* object. The interface between the two objects is a *Checkpoint* object. And finally, the Blacklist captured with the *Blacklist* object.

Participants

The following section describes the participating classes in the pattern structure:

- *Client:* An actor requesting access to a node, or *Service* object.
- *Service:* The intended protected object. A *Service* has access to a system's message processing.
- *Blacklist:* The system's list object for tracking bad addresses.
- *Checkpoint:* The interface through which all communication with a *Service* is achieved, and where the *Blacklist* is checked.

Collaborations

A given *Client* will attempt to communicate with a *Service* through a *Checkpoint*. Consequently, there is an association between both the *Service* and the *Checkpoint* and the *Client* and the *Checkpoint* that acts as the interface. There is also an association between a *Checkpoint* and the associated *Blacklist*.

Behavior

A Blacklist (Figure 12) is a solution that can partially fulfill the relevant security requirements that are unmet in the problem statement. A Blacklist maintains a list of addresses within a network that have exhibited inappropriate behavior. When a packet from a

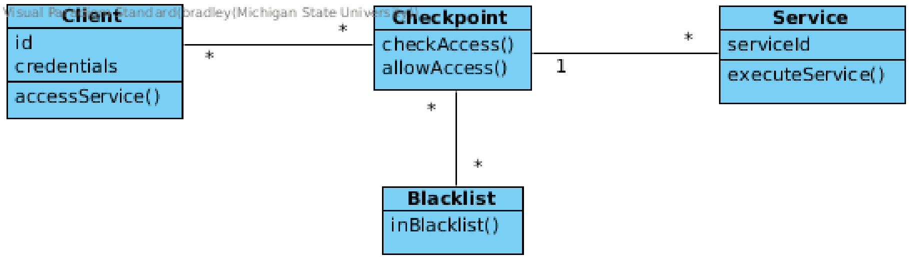

Figure 11 Class diagram for Blacklist Pattern.

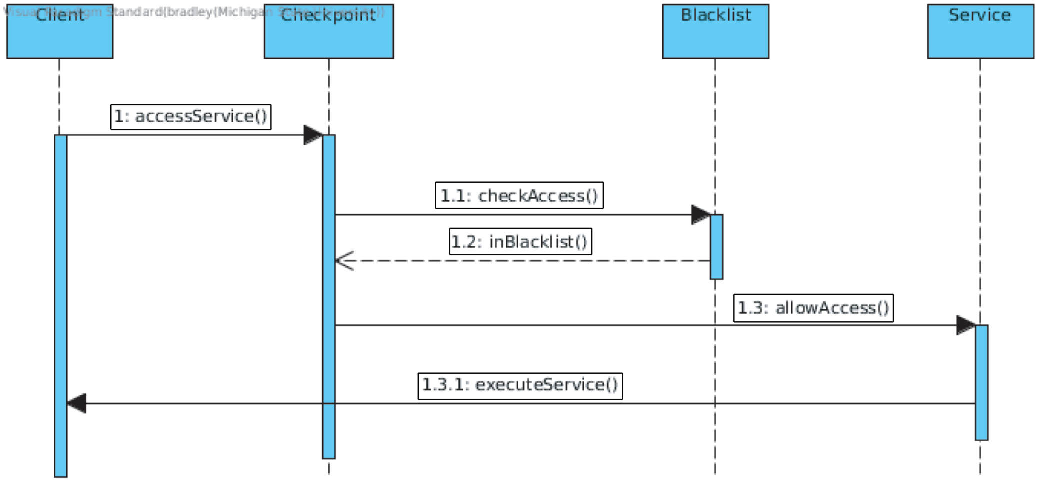

Figure 12 Sequence diagram for Blacklist Pattern.

blacklisted address arrives at a node, the node simply drops the packet. In the context of a VANET, nodes will not process or forward any messages that originate from a blacklisted address, and routing algorithms will not include the blacklisted nodes in calculating routes of the packets.

Constraints

Real-time constraints are important to consider in the context of the Blacklist pattern as message processing incurs overhead as well as consumes additional resources per message.

Consequences

See Table 8.

Known Uses

An example of a Blacklist being deployed in an automotive setting is given by Daeinabi and Rahbar [60]. Here the authors propose a system in which a select number of nodes in a VANET are tasked with monitoring behavior of the other nodes, where the overall network can dynamically change over time. If one of the monitoring nodes detects abnormal traffic from a given node, then it increases a distrust value for that node. Meanwhile, all nodes in the network maintain a Blacklist. If a node is given a distrust value beyond a certain threshold by the monitoring nodes, then that node is reported and blacklisted for the other nodes in the network.

Relevant Security Principles

The Blacklist leverages Practice Defense in Depth, Reluctance to Trust, and Compartmentalize.

Related Patterns

The Blacklist is related to traffic filtering patterns such as the Firewall pattern and the Signature-Based IDS.

Table 8 Consequences of Blacklist Pattern.

Accountability:	Preventing Misbehaving Nodes from Participating in a Network Improves the Ability of the Network to Hold Malicious Actors Accountable
Confidentiality:	Preventing nodes that are known to misbehave from accessing data sent between nodes in the network provides a higher degree of data confidentiality.
Integrity:	Preventing nodes that are known to misbehave from receiving and possibly modifying data sent between nodes in the network provides a higher degree of data integrity.
Availability:	Depending on the blacklisting protocol the Blacklist may prevent harmless nodes from participating in the network, thereby reducing availability.
Performance:	Performance may be affected by the overhead incurred by using the network Blacklist pattern. Performance improvements may occur however as the resources consumed by misuse are reduced .
Cost:	Not applicable.
Manageability:	By setting blacklist rules, manageability of a network can be improved.
Usability:	Depending on the blacklisting protocol, legitimate users may be prevented from accessing the services.

6.2.2. DDoS redundancy

DDoS Redundancy is a structural based security pattern.

Intent

The DDoS Redundancy pattern is intended to make a resource or network more resilient to a DDoS by providing redundant resources in case a resource becomes inundated with service requests.

Motivation

When a service is provided over a network, resources are typically allotted based on estimated average use. A DoS attack seeks to attack a service's availability to entities that may need to use it by flooding the service with requests. The service will have allocated all of its resources to the attacking node(s) leaving it unavailable to legitimate users. In VANETs, a DDoS attack may manifest as several nodes broadcasting on a frequency used by the VANET and consequently jamming the communication medium [61]. The result is the inability for legitimate applications and nodes to send messages on the VANET which may lead to potentially adverse affects.

Properties

The DDoS Redundancy can be used to satisfy the Availability property.

Applicability

The DDoS Redundancy is applicable to both prevention of an attack, and mitigation of an attack.

Structure

The DDoS Redundancy structure of a system can be captured in terms of their relationships (Figure 13). An entity sending a message is captured by a *Subject* object. The *Subjects* pass their request through a *Check Point* object which manages forwarding the request to the appropriate *Resource* as well as informing the *Resource Manager* about the request. The *Resource Manager* monitors loads of the system's redundant *Resources* and orchestrates with both the *Check Point* and *Resources* to balance the loads.

Participants

The following section describes the participating classes in the pattern structure.

- *Subject*
  - An actor requesting to access a *Resource*.
- *Check Point*
  - The interface through which all communication with a *Resource* is achieved, and where the *Resource Manager* is updated.
- *Resource*
  - The object requested by the *Subject*.
- *Resource Manager*
  - Tracks *Resource* usage and manages loads through orchestration with the *Check Point* and *Resource* objects.

Collaborations

A given *Subject* will attempt to communicate with a *Resource* through a *Check Point*. Consequently, there is an association between both the *Subject* and the *Check Point* and the *Resource* and the *Check Point*. The *Check Point* has a one to many relationship with *Resources* implying redundant resources. There is also an association between a *Check Point* and the *Resource Manager* which orchestrates forwarding by the *Check Point*. The *Resource Manager* also manages the many *Resources* by tracking loads and sending drop and accept messages based on *Resource* loads.

Behavior

The solution provided by the DDoS Redundancy pattern is related to a DDoS defense described by Mirkovic *et al.* [66] as Resource Multiplication. Many web based service providers deploy this method of redundant web-servers and advanced load-balancers to prevent their web-application from succumbing to a DDoS attack. In the given example of a VANET broadcast frequency being jammed by a DDoS attack, providing several frequencies allows redundant communication mediums for a VANET and consequently allows for a more available service. In Figure 14, a *Subject* requests a *Resource* at the *Check Point*. The *Check Point* forwards the request to the current *Resource* as well as an update to the *Resource Manager*. The *Resource* will not process the packet until it is signaled to do so by the *Resource Manager*. The *Resource Manager* flags the *Resource* as overloaded, and sends a drop request signal to the *Resource* and the resource overload signal to the *Check Point*. The *Check Point* then switches its forwarding protocol to a different *Resource*. The *Subject* will send a request again and this time it will be forwarded to the available *Resource*.

Constraints

Automotive systems are constrained by resource limitations. In the context of DDoS Redundancy, this may mean that switching to another resource may not be possible. In a real-time environment, overhead required to monitor resources may impede the system response speed while also using more resources per request process.

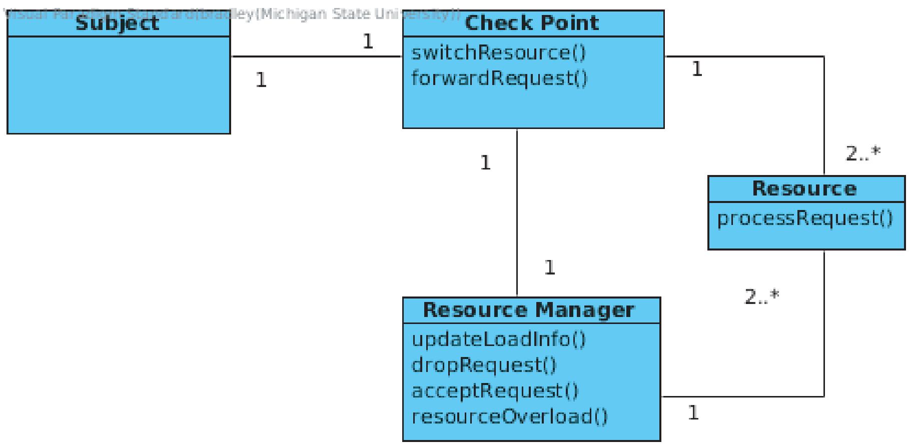

Figure 13 Class diagram for Distributed Denial of Service (DDoS) Redundancy Pattern.

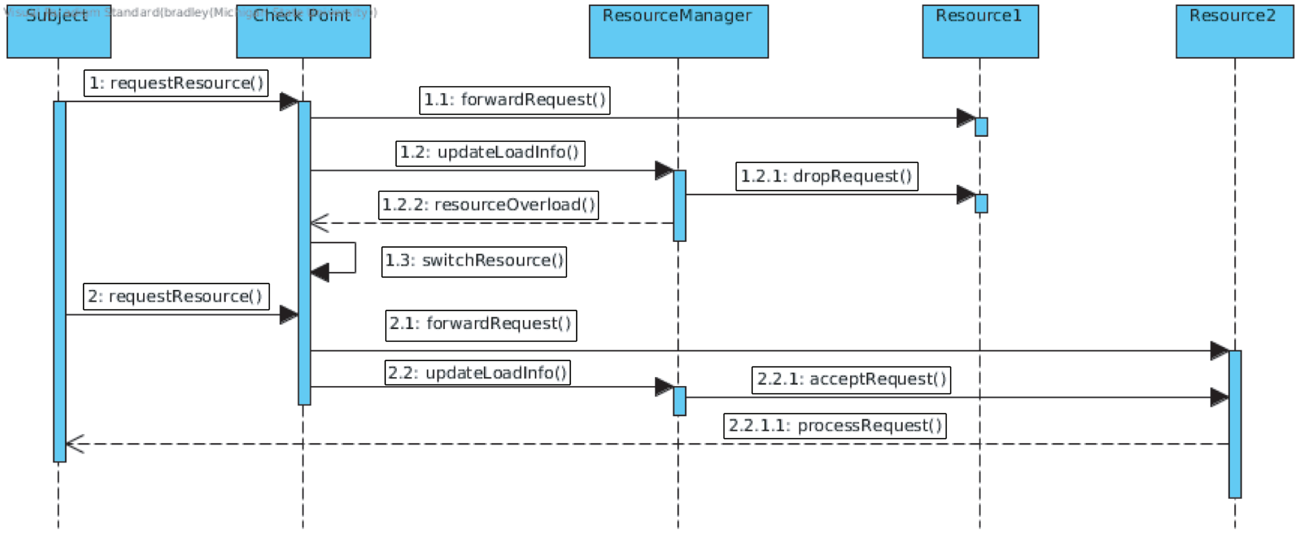

Figure 14 Sequence diagram for Distributed Denial of Service (DDoS) Redundancy Pattern.

Consequences

Table 9 describes how the DDoS Redundancy pattern affects the following system characteristics.

Known Uses

While DDoS Redundancy is employed widely by web applications today, proposed solutions for DDoS prevention in VANETs are described by Gillani *et al.* [37]. Researchers proposed switching between communication channels or technologies [67] when one is brought down, while other researchers [61] propose using multiple receivers operating in disjoint frequencies to provide a feasible solution.

Relevant Security Principles

Relevant security principles related to the DDoS Redundancy pattern include Practicing Defense in Depth and Compartmentalization [15]. The automotive security standard SAE J3061 [11] promotes defense in depth as a guiding security principle.

Relevant Security Principles

The DDoS Redundancy leverages Practice Defense in Depth, Fail Securely, and Compartmentalize.

Related Pattern

The DDoS Redundancy makes use of the Checkpoint pattern and the Secure Thread/Process patterns described by Fernandez [7].

6.2.3. Multi-factor authentication

Multi-factor Authentication is a structural based security pattern.

Intent

The Multi-factor Authentication pattern is used to provide a redundant security measure in authenticating an actor or a message. By enforcing an additional level of authentication, malicious actors can be prevented from attacking if a node or single credential is compromised.

Table 9 | Consequences of DDoS Redundancy Pattern.

Accountability:	Not Affected
Confidentiality:	Not Affected.
Integrity:	Not Affected.
Availability:	Improves availability by providing redundant resources when a service is inundated with requests.
Performance:	Not affected.
Cost:	The cost of additional resources, such as additional receivers in the previous example, would increase the cost of a system.
Manageability:	Not affected.
Usability:	The use of redundant resources may increase the usability of a given service by providing greater capacity.

DDos, Distributed Denial of Service.

Motivation

While it is common practice to authenticate messages and actors within a system using passwords or signatures, some highly critical systems that house personal data or handle safety-critical applications may be susceptible to attack if the passwords or signatures are compromised. If an actor's credentials are compromised, then it is possible to inject false information into a system, steal private data and modify data. In a VANET, if a given node or a vehicle within a smart infrastructure is compromised or an address/signature of the node is replicated (spoofed), the other vehicles in the VANET may be susceptible to a variety of attacks. As an example, in a Sybil attack where a single node sends a message from many fabricated or comprised addresses, applications such as those detecting traffic jams may incorrectly divert other vehicles away from a section of road even though the messages originate from one node [61]. The inability to authenticate these addresses as real vehicles or smart infrastructure could lead to critical safety applications from functioning correctly.

Properties

The Multi-factor Authentication can be used to satisfy the Authentication property, and the Confidentiality property.

Applicability

Multi-factor Authentication is applicable to the prevention of an attack.

Structure

The Multi-factor Authentication structure of a system can be captured in terms of their relationships (Figure 15). An entity requesting access is captured by a *Subject* object. The entity secured with a Multi-factor Authentication is represented as a *Protected Object*. The interface between the two objects is a *Checkpoint* object. The *Checkpoint* makes use of multiple *Authenticators* to obtain *Credentials* from a subject.

Participants

The following section describes the participating classes in the pattern structure.

- *Subject*
  - An actor requesting to access to a *Protected Object*.
- *Protected Object*
  - The intended protected object. The entity protected through an authentication *Checkpoint*.
- *Credential*
  - Some identifying secret required to access a *Protected Object*.
- *Checkpoint*
  - The interface through which all communication with a *Service* is achieved, and where the *Blacklist* is checked.
- *Authenticator*
  - Object tasked with retrieving a credential from a *Protected Object*.

Collaborations

A given *Subject* will attempt to communicate with a *Protected Object* through a *Checkpoint*. Consequently, there is an association between both the *Protected Object* and the *Checkpoint* and the *Subject* and the *Checkpoint* which acts as the interface. There is also an association between a *Checkpoint* and the multiple *Authenticators* associated with it. A given *Authenticator* is associated with a *Subject* which in turn is associated with several *Credentials*.

Behavior

By enforcing the Multi-factor Authentication pattern, a potential attacker is less likely to compromise an actor within a system as obtaining a single credential will not guarantee access to the actor (see Figure 16). Moreover, other nodes that may communicate with a compromised node are less likely to incorrectly validate spoofed messages given an extra level of authentication. Multi-factor Authentication is used in many web applications where a user is required to enter a password to log in as well as another authentication measure such as a security question the user would know, or a code sent to the user in a separate application such as SMS. In the example given previously, the use of another authenticating measure besides the address of a node would place a greater burden on the attacker using fabricated addresses. The Multi-factor Authentication strengthens an actor's resilience to compromise by strengthening authentication capabilities, and consequently the system's ability to promote integrity and privacy.

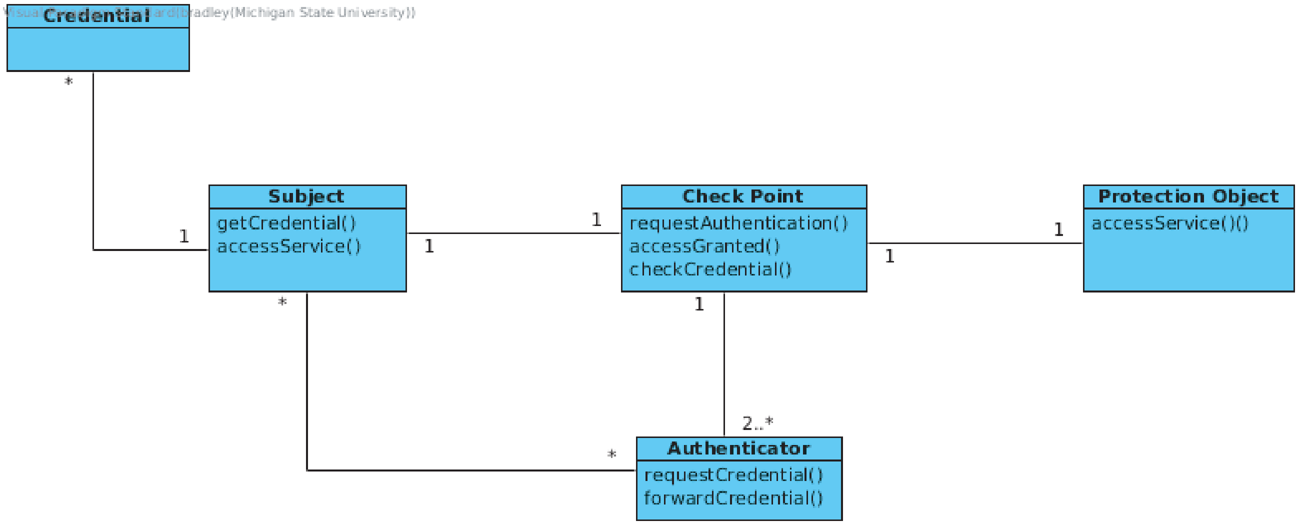

Figure 15 Class diagram for Multi-Factor Authentication Pattern.

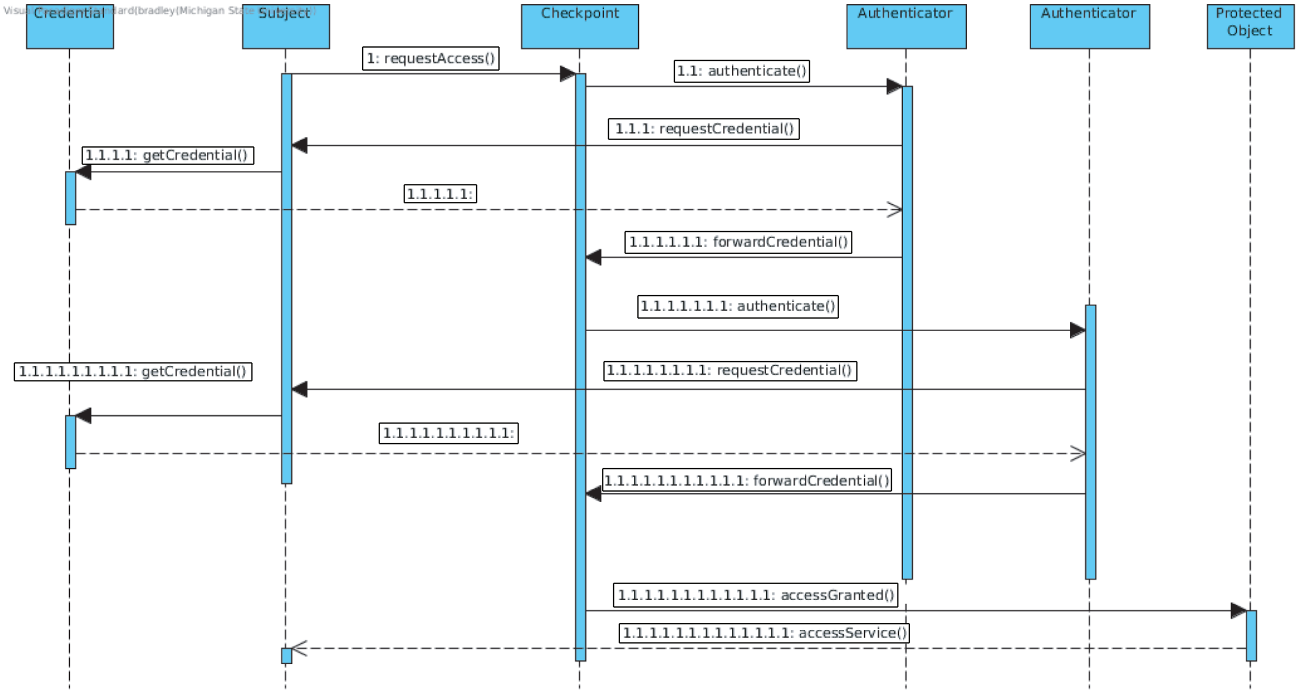
Credential
Subject
Checkpoint
Authenticator
Authenticator
Protected Object
1: requestAccess()
1.1: authenticate()
1.1.1: requestCredential()
1.1.1.1: getCredential()
1.1.1.1.1:
1.1.1.1.1.1: forwardCredential()
1.1.1.1.1.1.1: authenticate()
1.1.1.1.1.1.1.1.1: getCredential()
1.1.1.1.1.1.1.1: requestCredential()
1.1.1.1.1.1.1.1.1.1:
1.1.1.1.1.1.1.1.1.1.1: forwardCredential()
1.1.1.1.1.1.1.1.1.1.1.1: accessGranted()
1.1.1.1.1.1.1.1.1.1.1.1.1: accessService()

Figure 16 Sequence diagram for Multi-Factor Authentication Pattern.

Constraints

Automotive systems are limited in the amount of resources that can be provided. By requiring more overhead to authenticate *Subjects*, resources critical to safety may be affected. Furthermore, in VANET applications where an increased amount of nodes participating improves data collection and ability to route messages, adding additional authentication may prove challenging for some harmless nodes resulting in decreased participation.

Consequences

Table 10 describes how the Multi-factor Authentication pattern affects the following system characteristics.

Known Uses

In the given example of the Sybil attack on a VANET, where a single compromised node can fabricate many messages that appear to be from many nodes[61], a proposed solution to detecting the fabrication through the use of multiple points of message authentication has been developed [62]. Here, researchers propose the use of a GPS as a second form of authenticating a message from a given address. If a single node sends many messages with different addresses, then other nodes will be able to see the messages all originating from one node, implying a Sybil attack.

Relevant Security Principles

The Multi-factor Authentication pattern promotes the principles of Practicing Defense in Depth, Compartmentalization, and Being Reluctant to Trust.

Related Patterns

The Multi-factor Authentication pattern is related to the Authenticator and can make use of the Credential [7].

6.2.4. Symmetric encryption

Symmetric Encryption is a structural-based security pattern.

Intent

Symmetric Encryption is used to encrypt messages so that only a sender and receiver can read the contents.

Table 10 Consequences of Multi-Factor Authentication Pattern.

Accountability:	Accountability Is Improved as a System Is Able to More Accurately Authenticate Actors
Confidentiality:	Private data that may be accessible from compromising an actor is more secured.
Integrity:	The ability to infiltrate and alter data or inject data is made more difficult through more rigorous authentication.
Availability:	Not applicable.
Performance:	Overhead may be incurred by requiring additional authentication steps.
Cost:	The use of other services for additional authentication may incur a cost.
Manageability:	Not applicable.
Usability:	A system may become more difficult to use as multiple steps are required to authenticate legitimate use.

Motivation

Applications that communicate with external entities may send information that needs to be protected from unintended recipients [7]. In an automotive system that communicates with external entities, such as in a VANET where a vehicle can be communicating with other vehicles and infrastructure, the inability to accept and process only authentic messages may leave a system susceptible to spoofing and dDoS attacks [3]. In the case of the LESPP system, researchers seek to mitigate the authentication and privacy preservation challenges facing VANETs [49]. As it pertains to SAE J3061 [11], the use of Symmetric Encryption aligns with the principles of **protecting personal identifiable information and sensitive data**. The implementation of the pattern also aligns with the Viega and McGraw principles of **promoting privacy** and being **reluctant to trust** [15].

Properties

The Symmetric Encryption can be used to satisfy the Confidentiality property.

Applicability

The pattern is applicable to attack Prevention.

Structure

Both the *Sender* and *Receiver* have a *Key* that allows them to decipher a sent *Message* (Figure 17). A *Sender* will give its *Key* and the *Message* to an *Encryptor* object which will use an *Algorithm* to encrypt the *Message*. The receiver will give its key and the *Encrypted Message* to the *Decryptor* to obtain the original *Message*.

Participants

- *Principal*
  - Main actor that may send and receive messages. Can be defined by either of the inheriting classes *Sender* and *Receiver*.
- *Key*
  - Cryptographic key used by the *Principal* to encrypt and decrypt a message
- *Message*
  - The information to be sent and received. Has inheriting class *Encrypted Message* which is the resulting message after a *Message* undergoes encryption.
- *Encryptor*
  - Uses a *Message* and the *Sender*'s *Key* to create an *Encrypted Message*. Leverages the *Algorithm* Object.
- *Decryptor*
  - Uses an *Encrypted Message* and the *Receiver*'s *Key* to decrypt an *Encrypted Message*. Leverages the *Algorithm* Object.
- *Algorithm*
  - Given a *Message* and a *Key*, either decrypts an *Encrypted Message* or encrypts a *Message*.

Collaborations

A *Principal* has a *Key*, *Messages*, and *Encrypted Messages*. The specialized *Principal Sender* has *Encryptors* and the specialized *Principal Receiver* has *Decryptors*. A given *Message* can have up to one associated *Encrypted Messages*. A *Message* has one associated *Decryptor* and one associated *Encryptor*, thus maintaining consistency in the type of encryption and decryption techniques used. Similarly the *Encryptors* and *Decryptors* are associated with a single Algorithm.

Behavior

In the case of sending a *Message*, a *Sender* will send a message and *Key* to a *Encryptor* (Figure 18). The *Encryptor* will send the information along to the *Algorithm* to encrypt. The *Encryptor* can then create the *Encrypted Message* object and return it to the *Sender*.

Constraints

In an automotive system where constraints exist on real-time processing and resources, performing the algorithms used in cryptography can be costly in terms of overall system performance. Moreover, overhead is incurred when processing messages to ensure authenticity or to decrypt. In the case of LESPP, computation overhead is reduced by 41.33–77.6% compared to existing public key infrastructures and in simulations, the authors show nearly 0 ms network delay [49].

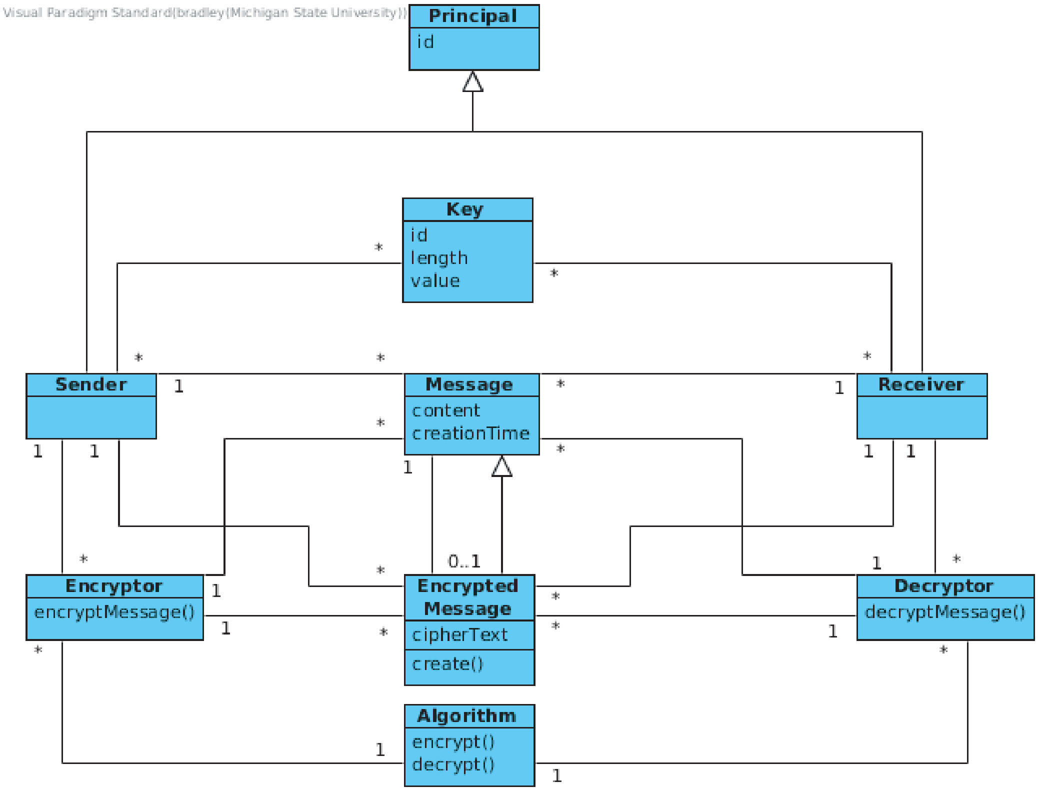

Figure 17 Class diagram for Symmetric Encryption Pattern.

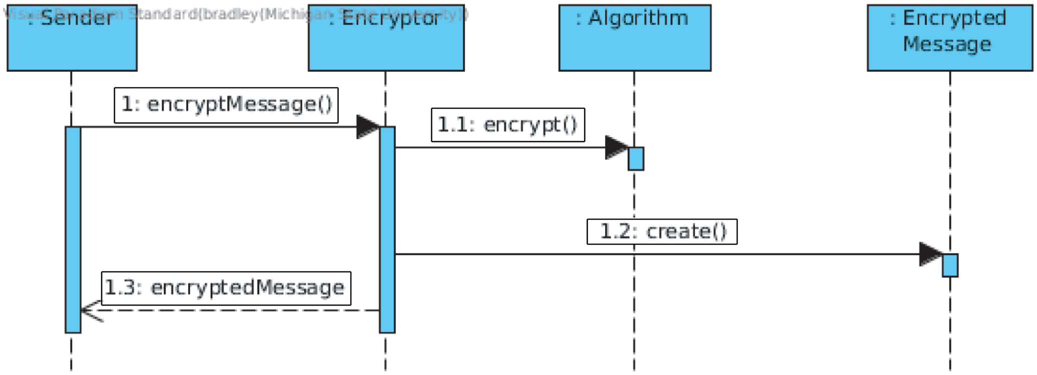

Figure 18 Sequence diagram for Symmetric Encryption Pattern.

Consequences

Table 11 describes the consequences of using the Symmetric Encryption pattern.

Known Uses

As described above, the use of symmetric cryptography has been used in the LESPP system described by Wang *et al.* [49]. By verifying a signature on messages sent over a VANET, LESPP can avoid DDOS and spoofing attacks from unauthenticated nodes.

Supported Principles

Symmetric Encryption supports Principle 7 (**Promoting Privacy**) as well as Principle 9 (**Be Reluctant to Trust**).

Related Security Patterns

The pattern is related to Asymmetric Encryption and Digital Signatures [7].

Table 11 Consequences of Symmetric Encryption Pattern.

Accountability:	Overall Accountability of a System Improves as VANET Communication Can Be Authenticated
Confidentiality:	In the case of LESPP, because the key management center is the only entity that can expose a vehicle's identity, confidential information is better protected.
Integrity:	System information and resources can be better protected from malicious actors.
Performance:	May require additional hardware.
Cost:	Additional hardware to process signatures may incur a cost.
Manageability:	May require additional management overhead for managing certificates of infrastructure and other vehicles.
Usability:	Usability of system resources may be affected by overhead of verifying signatures.

VANETs, Vehicle Ad-hoc Networks.

6.2.5. Third-party validation

Third-Party Validation is a structural based security pattern.

Intent

The Third-Party Validation pattern is intended to provide validation of messages broadcasted in a given network. If a spoofed message is sent to a node and the receiving node cannot validate the message with a trusted node somewhere else in the network, then the receiving node can disregard the message.

Motivation

In many networks that rely on messages being propagated to other nodes, there is concern that a compromised node may alter the message or inject a false message. This type of attack is known as a *Masquerading Attack* that can lead to fabrication, alteration, and replay attacks [37]. In a VANET, many applications, such as an application that reports traffic jams to other vehicles within the VANET, rely on nodes reporting on current conditions as they perceive them and propagating the messages of the nodes around the VANET to create a consensus for the road conditions. If compromised nodes in the VANET send falsified information to another node in the network, the automotive system could behave in an inappropriate manner and jeopardize safety [37].

Properties

The Third-Party Validation can be used to satisfy the Integrity property, and the Authentication property.

Applicability

Third-Party Validation is applicable to Detection and Mitigation.

Structure

The Third-Party Validation structure of a system can be captured in terms of their relationships (Figure 19). An entity broadcasting a *Message* is captured by a *Sender* object. Both the *TrustedNode* and *Receiver* actively listen and receive messages on a broadcast network.

Participants

The following section describes the participating classes in the pattern structure.

- *Sender*
  - Broadcasts a given *Message*.
- *Receiver*
  - Listens on a network for a broadcasted *Message*.
- *TrustedNode*
  - Listens on a network for a broadcasted *Message*. Responds to verification requests from *Receivers*
- *Message*
  - Unit of communication over the network.

Collaborations

A given *Sender* broadcasts a *Message* on a network, creating an association with the *Message*. Meanwhile, the other nodes on the network (*Receivers* and *TrustedNodes*) listen for *Messages*, creating associations with the *Messages*. Every given *Receiver* node has a one or more *TrustedNodes* used for verification.

Behavior

The Third-Party Validation pattern seeks to solve the concerns pertaining to integrity by obtaining information from more than one source. Importantly, if there are more trusted nodes within a network that are more secure, they can be leveraged for performing the validation and thus providing a more robust validation scheme. In the case of an application on a VANET, information received by a node can be compared to information received by some other node(s) in the network. If the information is not within some degree of similarity, then the information can be disregarded (see Figure 20).

Constraints

Many automotive systems must function in real-time environments to ensure safety. In the context of Third-Party Validation, adding

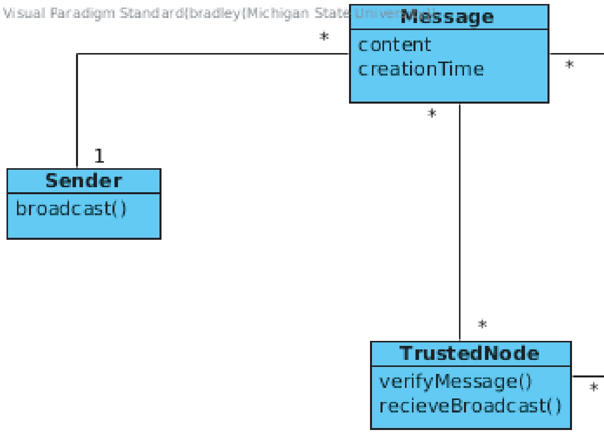

Figure 19 Class diagram for Third-Party Validation Pattern.

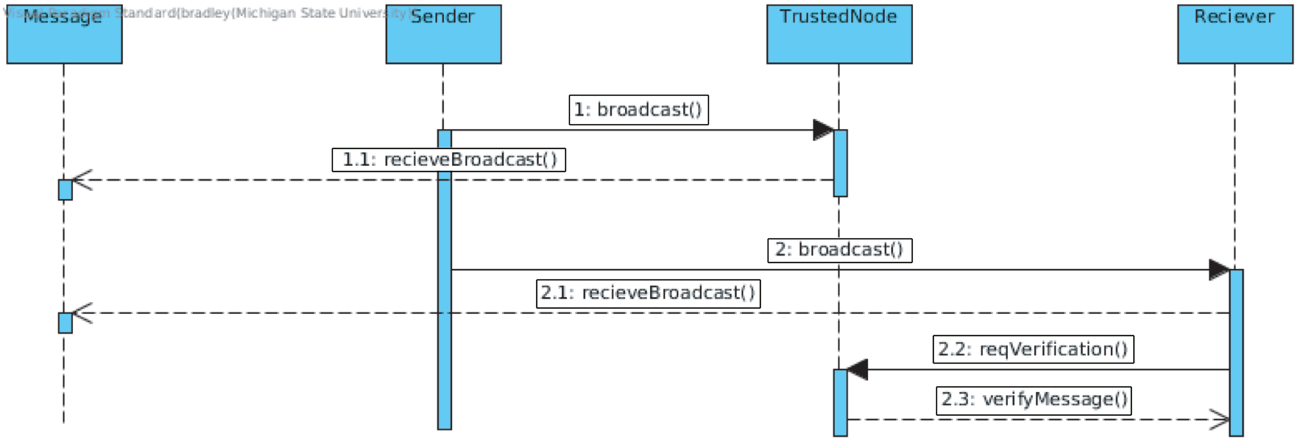

Figure 20 Sequence diagram for Third-Party Validation Pattern.

additional overhead in the verification steps of message processing could add an unacceptable degree of delay.

Consequences

Table 12 describes how theThird-Party Validation pattern affects the following system characteristics.

Table 12 Consequences of Third-Party Validation Pattern.

Accountability:	Not Applicable
Confidentiality:	Not Applicable.
Integrity:	Integrity of the system is improved as modified messages are flagged in the validation steps.
Availability:	Not applicable.
Performance:	Overhead from validation may incur performance loss.
Cost:	Not applicable.
Manageability:	Not Applicable.
Usability:	Resources may become more scarce as they are being used for validation.

Known Uses

In a proposed solution by Gillani *et al.* [37], when a node receives a message in a VANET, the node notifies a Road Side Unit (RSU) and requests to check the message correctness. The RSU will contact the RSU closest to the sending node to validate that the message was actually sent, and not spoofed or modified in transit.

Relevant Security Principles

Principles related to the Third-Party Validation pattern include Promoting Privacy, and Being Reluctant to Trust.

Related Patterns

This pattern is related to the Enrolling Using Third-Party Validation pattern described by [19].

7. CONCLUSION

As software and external communication become further integrated into modern automotive systems, it is critical that security is incorporated explicitly into the design and development process. By preventing, detecting, and mitigating attacks, automotive systems can continue to provide a better customer experience while also ensuring safety. With the use of automotive-focused security patterns, automotive software developers can incorporate known solutions to security problems into their systems more easily.

In this paper, we discussed security pattern concepts in the context of automotive systems and described the initial set of patterns in our repository. As part of this work, we introduced a template for describing automotive-focused security patterns that extends previously-developed security pattern templates [7,25]. Moving forward, we are working with our industrial collaborators to expand the security pattern repository, including making use of taxonomies of automotive security attacks [54]. We are also exploring how the security patterns can be incorporated into security and safety-focused development processes, similar to that proposed by Amorim *et al.* [47]. We are also exploring socio-technical approaches to securing automotive cybersecurity vulnerabilities [68]. Finally, with the upcoming release of the ISO/SAE 21434 Automotive Cybersecurity standard due for release in 2020 [70,71]. We will update the security patterns to incorporate the appropriate terminology modifications, process development guidelines, risk assessment strategies, and the interactions with safety standards defined in ISO26262 [71].

NOTE ADDED

A shortened and preliminary version of this work was previously published in the MODELS Workshop for Modeling for Automotive Systems (MASE2019). Descriptions for only two of the patterns were included in the workshop paper. This paper includes the descriptions for all of the patterns in the repository. Additional background, related work, and future work directions have also been added to this extended paper.

ACKNOWLEDGMENTS

We greatly appreciate the feedback provided by the anonymous reviewers, which has greatly helped to improve the paper.

This work has been supported in part by grants from NSF (CNS-1305358 and DBI-0939454), Ford Motor Company, General Motors Research, and ZF; and the research has also been sponsored by Air Force Research Laboratory (AFRL) under agreement numbers FA8750-16-2-0284 and FA8750-19-2-0002. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Air Force Research Laboratory (AFRL), the U.S. Government, National Science Foundation, Ford, GM, ZF, or other research sponsors. We are grateful to B. Czerny, I. Hawkes, B. Murray, and other industrial collaborators for their valuable feedback on this work.

REFERENCES

[1] Weimerskirch, A. Automotive and industrial data security. In: Cybersecurity and cyber-physical systems workshop; 2012.

[2] Miller, C, Valasek, C. Remote exploitation of an unaltered passenger vehicle. tech. rep., 2015.

[3] Checkoway, S, McCoy, D, Kantor, B, Anderson, D, Shacham, H, Savage, S, Koscher, K, Czeskis, A, Roesner, F, Kohno, T, et al. Comprehensive experimental analyses of automotive attack surfaces. In: Proceedings of the 20th USENIX Conference on Security, SEC'11; 2011.

[4] Parkinson, S, Ward, P, Wilson, K, Miller, J. Cyber threats facing autonomous and connected vehicles: future challenges. IEEE Trans Intell Transportation Syst 2017;18;2898–915.

[5] Zoppelt, M, Tavakoli Kolagari, R. What today's serious cyber attacks on cars tell us: consequences for automotive security and dependability. In: Papadopoulos, Y, Aslansefat, K, Katsaros, P, Bozzano, M, (eds.)Model-based safety and assessment, Cham: Springer International Publishing; 2019, pp. 270–85.

[6] Gamma, E, Helm, R, Johnson, R, Vlissides, J, Design patterns: abstraction and reuse of object-oriented design. In: European Conference on Object-Oriented Programming, Springer; 1993, pp. 406–31.

[7] Fernandez-Buglioni, E. Security patterns in practice: designing secure architectures using software patterns, John Wiley & Sons; 2013.

[8] Lin, CW, Zheng, B, Zhu, Q, Sangiovanni-Vincentelli, A. Security-aware design methodology and optimization for automotive systems. ACM Trans Des Automation Electronic Syst 2015;21;18.

[9] Wassermann, R, Cheng, BHC. "Security patterns," tech. rep., Department of Computer Science and Engineering, Michigan State University, East Lansing, Michigan 48824, 2003. (A shortened version appeared in the Proceedings of Requirements Engineering "Using Security Patterns to Model and Analyze Security Requirements" (with S. Konrad, L. Campbell, and R. Wassermann), IEEE Workshop on Requirements for High Assurance Systems, (RHAS03), September 2003, Monterey, California.).

[10] Fernandez, EB, Yoshioka, N, Washizaki, H. Patterns for security and privacy in cloud ecosystems. In: Evolving Security and Privacy Requirements Engineering (ESPRE), 2015 IEEE 2nd Workshop on, IEEE; 2015, pp. 13–8.

[11] Vehicle Cybersecurity Systems Engineering Committee. J3061 cybersecurity guidebook for cyber-physical vehicle systems, tech. rep., SAE International; 2016.

[12] Booch, G, Rumbaugh, J, Jacobson, I. Unified modeling language user guide. 2nd ed. Addison-Wesley object technology series, Addison-Wesley Professional; 2005.

[13] National Highway Traffic Safety Administration and others. Cybersecurity best practices for modern vehicles. Report No. DOT HS; 2016.

[14] Microsoft Corporation. Security development life cycle; June 2018.

[15] Viega, J, McGraw, G. Building secure software: how to avoid security problems the right way, Addison-Wesley; 2002.

[16] Gamma, E. Design patterns: elements of reusable object-oriented software, Pearson Education India; 1995.

[17] Ito, Y, Washizaki, H, Yoshizawa, M, Fukazawa, Y, Okubo, T, Kaiya, H, Hazeyama, A, Yoshioka, N, Fernandez, EB. Systematic mapping of security patterns research. In: Proceedings of the 22nd Conference on Pattern Languages of Programs, The Hillside Group; 2015, p. 14.

[18] Schumacher, M, Fernandez-Buglioni, E, Hybertson, D, Buschmann, F, Sommerlad, P. Security patterns: integrating security and systems engineering, John Wiley & Sons; 2013.

[19] Kienzle, DM, Elder, MC, Tyree, D, Edwards-Hewitt, J. Security patterns repository version 1.0, Washington, DC: DARPA; 2002.

[20] Yoshioka, N, Washizaki, H, Maruyama, K. A survey on security patterns. Prog Informatics 2008;5;35–47.

[21] Fernandez, EB, Washizaki, H, Yoshioka, N. Abstract security patterns. In: Proceedings of the 15th Conference on Pattern Languages of Programs, ACM; 2008, p. 4.

[22] Fernandez, EB, Yoshioka, N, Washizaki, H, Yoder, JW. Abstract security patterns for requirements specification and analysis of secure systems. In: WER; 2014.

[23] Uzunov, AV, Fernandez, EB, Falkner, K. Securing distributed systems using patterns: a survey. Comput Secur 2012;31; 681–703.

[24] Dougherty, C, Sayre, K, Seacord, R, Svoboda, D, Togashi, K, Secure design patterns, Tech. Rep. CMU/SEI-2009-TR-010, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 2009.

[25] Konrad, S, Cheng, BHC, Campbell, LA, Wassermann, R. Using security patterns to model and analyze security requirements. Requirements Eng High Assurance Syst (RHAS'03), 2003;11; 13–22.

[26] Corrigan, S. Introduction to the controller area network - texas instruments, Tech. Rep. SLOA101, Texas Instruments; 2016.

[27] Alam, MSU, Iqbal, S, Zulkernine, M, Liem, C. Securing vehicle ecu communications and stored data. In: ICC 2019 - 2019 IEEE International Conference on Communications (ICC); 2019, pp. 1–6,

[28] Avatefipour O, Malik, H. State-of-the-art survey on in-vehicle network communication "can-bus" security and vulnerabilities. Int J Comput Sci Netw 2017;6;720–7.

[29] Kaiser J, Mock, M. Implementing the real-time publisher/subscriber model on the controller area network (can). In: Object-Oriented Real-Time Distributed Computing, 1999. (ISORC'99) Proceedings. 2nd IEEE International Symposium on, IEEE; 1999, pp. 172–81.

[30] Nilsson, DK, Larson, UE, Picasso, F, Jonsson, E. A first simulation of attacks in the automotive network communications protocol flexray. In: Proceedings of the International Workshop on Computational Intelligence in Security for Information Systems CISIS'08, Springer; 2009, pp. 84–91.

[31] Van Herrewege, A, Singelee, D, Verbauwhede, I. Canauth-a simple, backward compatible broadcast authentication protocol for can bus. In: ECRYPT Workshop on Lightweight Cryptography, vol. 2011; 2011.

[32] Kent, D, Cheng, BHC, Siegel, J. Assuring Vehicle Update Integrity using Asymmetric Public Key Infrastructure (PKI) and Public Key Cryptography (PKC). SAE Int J Transportation Cybersecurity Privacy 2020;in press (escar USA 2020 Special Issue).

[33] Rouf, I, Miller, R, Mustafa, H, Taylor,T, Oh, S, Xu, W, Gruteser, M, Trappe, W, Seskar, I. Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study. In: Proceedings of the 19th USENIX Conference on Security, USENIX Security'10, (USA), USENIX Association; 2010, p. 21.

[34] Wang, J, Liu, J, Kato, N. Networking and communications in autonomous driving: a survey. IEEE Commun Surveys Tutorials 2019;21;1243–74.

[35] Huang, J, Zhao, M, Zhou, Y, Xing, C. In-vehicle networking: protocols, challenges, and solutions. IEEE Netw 2019;33;92–8.

[36] Rasheed,A, Gillani, S, Ajmal, S, Qayyum, A. Vehicular ad hoc network (vanet): A survey, challenges, and applications. In: Laouiti, A, Qayyum, A, Mohamad Saad, MN, editors. Vehicular ad-hoc networks for smart cities, Singapore: Springer Singapore; 2017, pp. 39–51.

[37] Gillani, SA, Shahzad, F, Qayyum, A, Mehmood, R. A survey on security in vehicular ad hoc networks. In: Communication technologies for vehicles, Springer; 2013, pp. 59–74.

[38] Jain, M, Saxena, R. Vanet: Security attacks, solution and simulation. In: Bhateja, V, Tavares, JMR, Rani, BP, Prasad, VK, Raju, KS, editors. Proceedings of the Second International Conference on Computational Intelligence and Informatics, Singapore: Springer Singapore, 2018, pp. 457–66.

[39] Mitra, S, Cheng, BHC, Survey on cybersecurity for automotive systems, tech. rep., Computer Science and Engineering, Michigan State University, East Lansing, MI, 2017.

[40] Miller, C, Valasek, C. Adventures in automotive networks and control units. In: DEF CON 21 Hacking Conference. Las Vegas, NV: DEF CON, 2013.

[41] Pagliery, J. Chryslers can be hacked over the internet, July 2015.

[42] Brooks, R, Sander, S, Deng, J, Taiber, J. Automobile security concerns. Vehicular Technol Mag 2009;4;52–64.

[43] Samara, G, Al-Raba'nah, Y. Security issues in vehicular ad hoc networks (VANET): a survey. International Journal of Sciences Applied Research, 2015; 2(4); 50–55.

[44] Ray, S, Chen, W, Bhadra, J, Al Faruque, MA. Extensibility in automotive security: current practice and challenges. In: Design Automation Conference (DAC), 2017 54th ACM/EDAC/IEEE, IEEE; 2017, pp. 1–6.

[45] Koushanfar, F, Sadeghi, AR, Seudie, H. Eda for secure and dependable cybercars: challenges and opportunities. In: Proceedings of the 49th Annual Design Automation Conference, ACM; 2012, pp. 220–8.

[46] Cho, KT. Shin, KG. Fingerprinting electronic control units for vehicle intrusion detection. In: USENIX Security Symposium; 2016, pp. 911–927.

[47] Amorim, T, Martin, H, Ma, Z, Schmittner, C, Schneider, D, Macher, G, Winkler, B, Krammer, M, Kreiner, C. Systematic pattern approach for safety and security co-engineering in the automotive domain. In: Computer Safety, Reliability, and Security; 2017, pp. 329–42.

[48] Vai, MM, Khazan, RI, Utin, DM, O'Melia, SR, Whelihan, DJ, Nahill, BR. Secure embedded systems, tech. rep., MIT Lincoln Laboratory Lexington United States, 2016.

[49] Wang, M, Liu, D, Zhu, L.Xu, Y, Wang, F. Lespp: lightweight and efficient strong privacy preserving authentication scheme for secure VANET communication. Computing 2016;98;685–708.

[50] Wolf, M, Gendrullis, T. Design, implementation, and evaluation of a vehicular hardware security module. In: International Conference on Information Security and Cryptology, Springer; 2011, pp. 302–18.

[51] Rizvi, S, Willet, J, Perino, D, Marasco, S, Condo, C. A threat to vehicular cyber security and the urgency for correction. Procedia Comput Sci 2017;114;.100–105.

[52] Wang, Q, Sawhney, S. A practical security framework to protect the can bus of vehicles. In: Internet of Things (IOT), 2014 International Conference on the; IEEE; 2014, pp. 13–8.

[53] Zhang, T, Antunes, H, Aggarwal, S. Defending connected vehicles against malware: challenges and a solution framework. IEEE Internet Things J 2014;1;10–21.

[54] Sommer, F, Dürrwang, J, Kriesten, R. Survey and classification of automotive security attacks. Information 2019;10;148,

[55] Ma, Z, Schmittner, C. Threat modeling for automotive security analysis, Advanced Science and Technology Letters, 2016; 139; 333–339.

[56] Macher, G, Armengaud, E, Brenner, E, Kreiner, C. A review of threat analysis and risk assessment methods in the automotive context; in Proceedings of International Conference on Computer Safety, Reliability, and Security, 2016;130–41.

[57] Macher, G, Sporer, H, Berlach, R, Armengaud, E, Kreiner, C. Sahara: a security-aware hazard and risk analysis method. In: 2015 Design, Automation Test in Europe Conference Exhibition (DATE); 2015, pp. 621–24.

[58] Zoppelt, M, Tavakoli Kolagari, R, A security abstraction model for automotive software systems," in Sam: a security abstraction model for automotive software systems. In: Hamid, B, Gallina, B, Shabtai, A, Elovici, Y, Garcia-Alfaro, J, editors. Security and Safety Interplay of Intelligent Software Systems, Cham: Springer International Publishing; 2019, pp. 59–74,

[59] Mell, P, Scarfone, K, Romanosky, S. Common vulnerability scoring system, IEEE Secur Privacy, 2006;85–89.

[60] Daeinabi, A. Rahbar, AG. Detection of malicious vehicles (dmv) through monitoring in vehicular ad-hoc networks, Multimedia Tools Appl, 2013;325–338.

[61] Rawat, A, Sharma, S, Sushil, R. VANET:Security attacks and its possible solutionsJ Inf Operations Manag2012;3;301.

[62] Yan, G, Wen, D, Olariu, S, Weigle, MC. Security challenges in vehicular cloud computingIEEE Trans Intell Transportation Syst2013;14;284–294,

[63] Larson, UE, Nilsson, DK, Jonsson, E. An approach to specification-based attack detection for in-vehicle networks. In: 2008 IEEE Intelligent Vehicles Symposium; June 2008, pp. 220–5.

[64] Picconi, F, Ravi, N, Gruteser, M, Iftode, L. Probabilistic validation of aggregated data in vehicular ad-hoc networks. In: Proceedings of the 3rd international workshop on Vehicular ad hoc networks, ACM; 2006, pp. 76–85.

[65] Ghosh, M, Varghese, A, Kherani, AA, Gupta, A. Distributed misbehavior detection in VANETs. In: Wireless Communications and Networking Conference, 2009. WCNC 2009, IEEE; 2009, pp. 1–6.

[66] Mirkovic J, Reiher, P. A taxonomy of ddos attack and ddos defense mechanismsACM SIGCOMM Comput Commun Rev 2004;34;39–53.

[67] Raya M, Hubaux, JP. The security of VANETs. In: Proceedings of the 2nd ACM international workshop on Vehicular ad hoc networks, ACM; 2005, pp. 93–94

[68] Kennedy, J, Holt, T, Cheng, B. Automotive cybersecurity: assessing a new platform for cybercrime and malicious hacking. J Crime Justice 2019;42;632–645,

[69] Schmittner, C, Griessnig, G, Ma, Z. Status of the development of iso/sae 21434. In: 25th European Conference, EuroSPI 2018, Bilbao, Spain, September 5-7, 2018, Proceedings, EuroSPI, 01 2018, pp. 504–13.

[70] Schmittner, C, Macher, G, Ma, Z. Automotive cybersecurity standards-relation and overview, in Computer Safety, Reliability, and Security (A. Romanovsky, E. Troubitsyna, I. Gashi, E. Schoitsch, and F. Bitsch, eds.), (Cham), Springer International Publishing, 2019. pp. 153–165.

[71] Road vehicles — functional safety — part 2: Management of functional safety. ISO 26262-2:2018(E).