A Functional Co-Design towards Safe and Secure Vehicle Platooning

by

Jiafa Liu

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science
(Computer and Information Science)
in The University of Michigan-Dearborn
2017

Master's Thesis Committee:

Associate Professor Di Ma, Chair
Associate Professor Jinhua Guo
Associate Professor Shengquan Wang

ACKNOWLEDGEMENTS

Firstly, I would like to express my sincere gratitude to my advisor Prof.Di Ma for the continuous support of my master study and related research, for her patience, motivation, and immense knowledge. Her guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my master study.

Besides my advisor, I would like to thank the rest of my thesis committee: Prof.Jinhua Guo, Prof.Shengquan Wang for their insightful comments and encouragement, but also for the hard question which incented me to widen my research from various perspectives.

Last but not least, my deepest gratitude goes to my family for their never ending support and love.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Cooperative adaptive cruise control (CACC) or platooning recently becomes promising as vehicles can learn of nearby vehicles' intentions and dynamics through wireless vehicle to vehicle (V2V) communication and advanced on-board sensing technologies. The complexity of automated vehicle platoon system opens doors to various malicious cyber attacks. Violation of cybersecurity often results in serious safety issues as been demonstrated in recent studies. However, safety and security in a vehicle platoon so far have been considered separately by different sets of experts. Consequently no existing solution solves both safety and security in a coherent way. In this paper, we show cyber attacks on an automated platoon system could have the most severe level of safety impact with large scale car crash and argue the importance of safety-security co-design for safety critical cyber physical systems (CPS). We propose a safety-security co-design engineering process to derive functional security requirements for a safe automated vehicle platoon system based on a deep comprehension on the interrelation of safety and security. To our best knowledge, we are the first to apply the safety-security co-design concept to a concrete application. Through this engineering process, we propose a general approach for designing a safe and secure platooning. Following the general approach, we come up with a new platoon control algorithm that takes into account both safety and security. Our defense mechanism implicitly defends against safety-related cyber-attacks and greatly shortens the safe distance required when the platoon is not protected.

CHAPTER 1

Introduction

1.1 Motivation

Vehicle platooning has been studied as a method of increasing the capacity of roads since the 1960's. In a vehicle platoon, a group of vehicles, following one another, acts as a single unit through coordinated movements. Because vehicles in a platoon travel together closely yet safely, this leads to a reduction in the amount of space used by the number of vehicles on a highway, thus has the great potential to *maximize highway throughput*. Cooperative adaptive cruise control (CACC) or automated vehicle platooning recently becomes promising as vehicles can learn of nearby vehicles' intentions and dynamics through wireless vehicle to vehicle (V2V) communication and advanced on-board sensing technologies. Automation-capable vehicles in tightly spaced, computer-controlled platoons offer additional benefits such as *improved mileage and energy efficiency* due to reduced aerodynamic forces, as well as *increased passenger comfort* as the ride is much smoother with fewer changes in acceleration.

The feasibility of automated vehicle platooning has been recently demonstrated successfully by the demonstrator platooning system developed by the European's Safe Road Trains for the Environment (SARTRE) project team [1]. The SAR-TRE project aims to develop strategies and technologies to allow vehicle platoons to operate on normal public highways. A demonstrator platooning system with trucks and cars has been developed and successfully tested on both test tracks and public motorways. According to the project director of intelligent transport systems,

technically the SARTRE platooning could be ready for rollout in 10 years.

The complexity of an automated vehicle platoon system – including inter-vehicle communications, vehicle's internal networking and its connection to external networks, as well as complicated and distributed platooning controllers – opens doors to malicious attacks. In-vehicle range sensors that are used to measure the preceding car's speed and location might be altered. For instance, it was recently demonstrated that radar and LIDAR sensors can be spoofed with a modulated laser [2]. The wireless communication channel (DSRC) is vulnerable to manipulation and wireless messages can be spoofed by a motivated attacker [3, 4, 5]. All these attacks could cause a wide array of problems in a deployed platoon, for example, an attacker could cause crashes, reduce fuel economy through inducing oscillations in spacing, prevent the platoon from reaching its (or each individual's) destination(s), or cause the platoon to break up. The full potential of automated vehicle platooning will not be realized until the issues related to communication and application security can be satisfyingly resolved.

1.2 Problem Statement

The violation of cybersecurity could result in serious safety violations such as car crashes. However, safety and security in a vehicle platoon have so far been considered separately by different sets of experts. On one hand, the safety discipline usually considers system failures (including systematic/random hardware and systematic software failures) or natural disasters as safety hazard resources. Safety solutions developed are usually not evaluated in an adversarial environment. On the other hand, the security discipline considers various attacks that can lead to different consequences such as loss of life, loss of privacy, financial loss, etc. The variety of security goals to address different types of attacks makes it very unlikely to be aligned with the goal of safety. Consequently security solutions proposed are rarely evaluated in terms of safety. For example, the model-based detection scheme [4], the only scheme proposed so far for platoon security, is

designed from the **security point of view** by *monitoring any misbehavior of the preceding car*. Although the scheme is able to detect vehicle misbehavior, whether it can lead to a safe platoon is not answered. To date, no existing platooning solution solves both safety and security in a reconciled and coherent way.

The need for a safety and security co-design is urgent today with the practicality of automated vehicle platooning technology. Actually there has been calls long ago for safety and security communities to work together [6]. Past efforts in the automotive industry have reached a consensus that functional safety hazards can arise from malicious activities in addition to systematic failures and random hardware failures [7]. So security should be considered as a pre-requisite for safety while safety should be one of the driving forces for security design. Although a couple of works have described a safety and security engineering process [7, 8], a lot of challenges need to be addressed to come up with a concrete safe and secure platoon system: How to reconcile different safety and security risks? How to align the goal of security with that of the safety? The most important, how to arrive at a solution that satisfies both the safety and security requirements? There are also performance challenges such as efficiency, real time, as well as maintaining the string stability of platoon.

On the other hand, current secure platooning studies have not taken safety issue into consideration. Some attacks only compromise vehicle traffic privacy, while others can be safety critical and cause traffic accidents. Safe vehicle platooning is an essential step on the way to realize automated vehicles. While many safe strategies and communication security have been researched, secure strategies, risks and opportunities of sensing technology have not been researched well. Therefore, understanding secure and safe platooning is a crucial requirement to be able to understand secure automated vehicles. Although many vehicle platooning control algorithms have been developed to achieve string stability, they have not been developed and analyzed under adversarial environment where an adversary wants to inhibit the performance of the control algorithm and hence cause instability of the system which may result in poor mileage, user discomfort, or even “intelligent

collision". Safety risks have to be carefully assessed and corresponding safety mechanisms have to be developed to the full realization of such autonomous systems.

Based on a joint functional safety and security analysis, we are able to reconcile different safety and security risks. For our purpose, we consider the subset of security threats that lead to safety consequences. This allows us to align our security goal with that of the safety. We propose a new platooning control algorithm that is designed from **the safety point of view**. Unlike the model-based detection scheme [4] which is designed from the security point of view where a vehicle treats the one before it as *potentially malicious*, in our scheme, a vehicle concentrates on self-safety, calculates *its own safety status* (instead of predicting other's misbehavior) based on the context information and adjusts its next movement based on one criterion: whether it is safe to do so. If it senses the next step is not safe, the vehicle will switch from the cooperative driving CACC mode to the collision avoidance ACC mode. By centralizing on self-safety, our scheme achieves safety by implicitly defending against cyber attacks that could result in safety consequences.

1.3 Thesis Contribution

Contributions: Our contributions are as follows:

- Based on a deep comprehension of the potential security and safety risks, we put emphasis on safety-security co-design and make recommendations related to security and safety in automated vehicle platooning by integrating cybersecurity into safety design strategies. To our best knowledge, we are the first to apply the safety-security co-design concepts to a concrete application (Chapter 4).

- To fully understand the severity of cybersecurity-induced safety risk, we introduce a leader crash attack to demonstrate the severity of such attack on the safety of vehicle platoon. To ensure the safety of platoons under attacks, We propose the concept of *safe distance*

for platoons. A platoon has to travel with at least the safe distance to avoid any collision (Chapter 4). Existing platoon systems leave no enough safe distance for the vehicles to decelerate when the platoon is driving on the highway with a minimal gap between each other. Leader vehicle crash will directly cause the following vehicles to collide, endangering the safety of all drivers .

- Based on the co-design analysis, we propose a general approach for designing a safe platooning. We inist that platoon should maintain a safe distance and at the same time, detect various potential cyber attacks. When the platoon is under cyber attack, it should switch to fail-safe scheme to avoid collision (Chappter 5).

- We propose a new platoon control algorithm emphasizing on self-safety. In our scheme, each vehicle cross-checks accelerations predicted by both the CACC and ACC controllers to determine whether the next move is safe. The cross-check mechanism is designed in a way such that it can guarantee string stability in normal operation and prevents collision attacks from happening by switching from the collaborative driving CACC mode to the collision avoidance ACC mode in abnormal situations. Our defense mechanism implicitly defends against cyber-attacks which may result in car collisions. Meanwhile, our defense mechanism greatly shortens the safe distance required when the platoon is not protected (Chapter 7).

- Unlike previous works which only use simulations, we demonstrate the effectiveness of the proposed scheme in achieving the safety goal as well as defending against cybersecurity attacks not just through vehicle network simulations but also through vigorous theoretical analysis (Chapter 8).

1.4 Thesis Organization

Organization: The organization of the paper is as follows. We overview related work in Chapter 2. We present system and attack models as well as platooning controllers in Chapter 3. We perform a joint safety and security risk analysis in Chapter 4 where we also introduce a leader crash attack to analyze the severity of such attack on safety. Then we propose a general approach for designing a safe platooning in Chapter 5. We present two safe platooning schemes in Chapter 6 and 7. Security analysis is presented in Chapter 8. Discussions and future work are presented in Chapter 10 and Chapter 11 concludes the paper.

CHAPTER 2

Related Work

Vehicle and Vehicle Network Security. Vehicle security is an emerging topic. A number of previous works have demonstrated many insecure designs in modern vehicles [9, 10, 11, 12]. Vehicle network security has been extensively studied. Many techniques such as efficient message authentication, anonymous authentication to address various aspects of communication security and privacy have been proposed [13, 14]. Chenxi Zhang [13] presents an efficient batch signature verification scheme for communications between vehicles and roadside units. Xiaodong Lin [14] proposes an efficient social-tier-assisted packet forwarding protocol, for achieving receiver-location privacy preservation in Vehicular Ad hoc Networks. The study of [9] shows an internal attacker who has prior physical access to the vehicle is able to adversely control a wide range of automotive functions and completely ignore driver input, including disabling the brakes, stopping the engine and so on. The authors also show it is possible to bypass rudimentary network security protections within the car and they also present composite attacks that can leverage individual weaknesses. The work of [10] demonstrates that remote exploitation is feasible via a broad range of attack vectors (including mechanics tools, CD players, Bluetooth, and cellular radio). Attack on the tire pressure-monitoring system (TPMS) [11] shows that eavesdropping is easily possible at a distance of roughly 40 m from a passing vehicle. They also find out that reverse-engineering of the underlying protocols reveal static 32 bit identifiers, which raises privacy concerns as vehicles can be tracked through these identifiers. The work of [12] demonstrates how to remotely hack a 2015 Jeep Cherokee and control its most vital functions. This attack puts hundreds of thousands of

Jeeps under risk and it forces a 1.4 million vehicle recall by FCA as well as changes to the Sprint carrier network.

Platooning Security. Traditional platooning research mainly focuses on four aspects: Inter-vehicle communication methodology, Collision avoidance and obstacle detection methodology, Design of lateral and longitudinal control systems for a platoon, last but not least, String stability of a platoon [15].

The security of autonomous platooning has been recently studied. Mani Amoozadeh [3] presents a first look at the effects of security attacks on the communication channel as well as sensor tampering of a connected vehicle stream. The adversary can use message falsification, spoofing or replay attacks to maliciously affect the vehicle stream. Another type of attack is tampering with vehicle hardware or software, which can be done by malicious insider at the manufacturing level or by an outsider in an unattended vehicle. The work of [4] introduces a set of insider attacks that can cause unexpected behavior in platoons. Mis-report attack is the attack that sends false message to the following vehicle to increase the following distance of the preceding car. Collision induction attack can cause dangerous accidents by broadcasting an acceleration message indicating that they are speeding up, while the attacker starts to aggressively brake. It suggests switching from CACC to ACC if a crash could happen. It focuses on the detection of false message attack from the proceeding car. In the false message attack, a malicious proceeding car driving at a low speed misreports it is driving at a high speed. The false message may mis-lead the following car to collide with the malicious proceeding car. From the security angle, the proposed solution is to let each vehicle monitor the behavior of the proceeding car. If the received status info from the proceeding car is different from the one the following car calculates, the following car will think the proceeding car may behave abnormally and switch to ACC. However it is not clear whether the switch from CACC to ACC can lead to a safe platoon. However, this passive solution has shortcomings. The

misbehavior detected cannot be delivered to the entire platoon immediately because the message broadcasting to following vehicles costs time and the message range is distance limited. Soodeh Dardras [5] presents that a single malicious controlled vehicle can destabilize a vehicular platoon. They show that an attacker is theoretically capable of gaining control over the individual position and velocity of other vehicles in the platoon.

Collision Avoidance. For collision avoidance, Gehrig and Stein [16] have proposed the concept of elastic bands and analyzed collision avoidance. Araki [17] presents a system which has automatic braking when the headway distance between the trailing vehicle and the selected vehicle crossed the safety threshold. Ferrara and Vecchio [18] propose a concept of a supervisor for the control system of every vehicle in the platoon to avoid collision. When a possible collision is detected, the control system switches from the normal “cruise mode" to “collision avoidance mode", causing the vehicle to stop following the preceding vehicle.

CHAPTER 3

Models and Simulation Environment

In this section, we present the system model we use. We also provide information about the simulation platform we will use to carry out the work presented in this paper.

3.1 Adversary Model

We consider insider attacks that can lead to safety issues such as car crashes in this work. Attacks that result in different consequences such as system performance, driver privacy, financial loss, etc. are not considered in this paper as they can be treated in the regular way without considering safety.The adversary or the vehicle controlled by the adversary is part of the platoon system and thus is able to send valid V2V messages. However, there is no guarantee on the correctness of information in the messages it sends. Also the adversary does not need to follow the control law.

The adversary is able to control one or more vehicles, including the leader, in the platoon. However, it cannot control all the radars or radar signals of vehicles in the platoon because of the line-of-sight requirement.

3.2 System Model

We consider a platoon of K cars numbered from 0 to $K-1$ with car 0 being the leader vehicle. We assume the platoon is already formed and do not consider platoon formation and dissolve (we leave platoon dynamics as our future work). All cars drive on a straight line with string

stability. The order of cars does not change. We assume homogeneous cars which have the same physics, mechanics, and communication capabilities (this requirement will be relaxed in our future work). They are not immune to hardware/system failures, and cybersecurity attacks, so abnormal behaviors can happen.

3.2.1 Communication and Mobility Models

In order to study the safety and security of a CACC vehicle system, we utilize the PLEXE platform [19] for its built-in communication (IEEE 802.11p) and mobility models. PLEXE is an Open Source extension to the known and widely used Veins simulation framework [20] by adding platooning capabilities and controllers. Veins itself extends the OMNeT++ network simulator and the SUMO road traffic simulator. PLEXE combines the network and the mobility simulator by creating a network node in OMNeT++ for each vehicle traveling in SUMO. It provides a vehicular communication system based on IEEE 802.11p and a mobility model based on SUMO. When a vehicle moves, PLEXE repeats the movement in the corresponding OMNeT++ node by updating the mobility model in SUMO. OMNeT++ framework implements platooning protocols and application logic, while SUMO realizes the actuation of the applications decisions as well as part of application logic.

If we want to tell the vehicles what they should do, we need to refer to application layer logic. There is a BaseApp in PLEXE which simply extracts data out of packets coming from the protocol layer, and updates CACC data via TraCI if such data is coming either from the leader vehicle or from the preceeding vehicle. The SimplePlatooningApp extends BaseApp and and it tells the vehicles to use the controller requested by the user. We modify the SimplePlatooningApp to simulate Leader Crash Attack. We make the platoon drive at a fixed speed of 100 km/h and fixed gap of 5 meters. At a certain time, we send a messgae to leader vehicle to tell it to stop. We set the deceleration of the leader car extremely large so that the speed can decelerate to zero in a minimal

time interval. In this way, it acts just like a leader car crash. It allows us to see how the following vehicles will respond conducted by the CACC controller strategy. PLEXE implements a list of classic controllers for Cruise Control (CC), Adaptive Cruise Control (ACC), and Cooperative Adaptive Cruise Control (CACC) to realize platooning capabilities. In the following we provide a brief introduction to these controllers and users are referred to [21] for more details.

3.2.2 Cruise Control

CC is a technology which allows a driver to select a desired speed and the car is driven automatically at the desired speed until CC is switched off by the driver. PLEXE implements the classic Cruise Control algorithm (Equation 3.1) which is already available on several commercial cars [21].

$$\ddot{x}_{des} = -k_p(\dot{x} - \dot{x}_{des}) - \eta \tag{3.1}$$

where $\ddot{x}_{des}$ is the acceleration to be applied, $\dot{x}$ is the current speed and $\dot{x}_{des}$ is the desired speed, k_p is the gain of the proportional controller (set to 1 by default), while η is a random disturbance taking into account imprecision of the actuator and of the speed measure (default set to 0).

3.2.3 Adaptive Cruise Control

As CC only takes the desired and actual speed as inputs, the driver needs to manually switch off CC to avoid a collision when approaching a slower vehicle in the front. To avoid collision and also relieve the driver from this duty, high-end cars are now equipped with a radar or laser scanner to estimate distance to the preceding car. If a slower car is detected, the system decelerates and automatically maintains a safe distance. This technology is known as ACC. ACC will automatically slow down the vehicle whenever it finds obstacles in the way.

ACC makes use of radar to detect vehicles in front and calculate the desired acceleration with only preceding car into consideration. ACC will automatically slow down the vehicle whenever it

finds obstacles in the way. CACC primarily rely on wireless communication to broadcast driving information to each vehicle. After receiving the message from preceding vehicle and leader vehicle, the controller computes the desired acceleration for the current vehicle. Different from CACC, ACC makes use of radar to detect vehicles in front and calculate the desired acceleration with only preceding car into consideration. ACC will automatically slow down the vehicle whenever it finds obstacles in the way. The control law of ACC [21] used in PLEXE is defined as

$$\ddot{x}_{i\_des} = -\frac{1}{T}(\dot{\varepsilon}_i + \lambda\delta_i) \tag{3.2}$$

$$\delta_i = x_i - x_{i-1} + l_{i-1} + T\dot{x}_i \tag{3.3}$$

$$\dot{\varepsilon}_i = \dot{x}_i - \dot{x}_{i-1} \tag{3.4}$$

where T is the time headway in seconds and $\dot{\varepsilon}_i$ is the relative speed between two consecutive vehicles i and $i+1$. δ_i is the distance error which is the difference between the actual distance $(x_i - x_{i-1} + l_{i-1})$ and the desired distance $T\dot{x}_i$. λ is a design parameter which is strictly greater than 0 and set to 0.1 by default.

The ACC driving functionality in PLEXE is implemented through the use of both ACC and CC controllers. When the ACC driving mode is selected, a car follows the instruction of the one which predicts smaller acceleration rate:

$$\ddot{x}_{des} = min(\ddot{x}_{CC}, \ddot{x}_{ACC}) \tag{3.5}$$

Basically, if the CC decides to accelerate to reach the desired speed, but the ACC decides to slow down because of a vehicle in front, the car will follow the instructions of the ACC. On contrary, if the ACC decides to accelerate to follow the car in front, but the car has reached its desired speed, the CC will make the car to "detach" from the preceding one. If there is no car in front (assuming

that the radar detects no car in front) or the distance is larger than $250m$, the car only considers CC even when it is in the ACC mode. Notice that this might not be the best strategy to implement, but for the sake of simplicity PLEXE [19] chooses to use this straightforward switching mechanism.

3.2.4 Cooperative Adaptive Cruise Control

The CACC controller implemented in PLEXE is a representative CACC controller based on classical control theory [21]. The status of each vehicle depends on the acceleration and speed of the leading and preceding vehicle in order to keep close vehicle following. It is capable of maintaining a fixed distance between cars no matter what the platoon's speed is.

The control law of the i-th vehicle in the platoon is defined as

$$\ddot{x}_{i\_des} = \alpha_1 \ddot{x}_{i-1} + \alpha_2 \ddot{x}_0 + \alpha_3 \dot{\varepsilon}_i + \alpha_4 (\dot{x}_i - \dot{x}_0) + \alpha_5 \varepsilon_i \tag{3.6}$$

$$\varepsilon_i = x_i - x_{i-1} + l_{i-1} + gap_{des} \tag{3.7}$$

$$\dot{\varepsilon}_i = \dot{x}_i - \dot{x}_{i-1} \tag{3.8}$$

$\ddot{x}_{i\_des}$ is the desired acceleration of i-th vehicle. $\ddot{x}_{i-1}$ and $\ddot{x}_0$ are the acceleration of the preceding vehicle $i-1$ and the leading vehicle. $\dot{x}_i$ and $\dot{x}_0$ are the speed of i-th vehicle and leader vehicle. ε_i is the distance error based on a desired constant distance gap_{des} which is 5 meters by default. l_{i-1} is the length of car and the default value is 4 meters.

The α_i parameters in Equation 3.6 are defined as:

$$\alpha_1 = 1 - C_1; \qquad \alpha_2 = C_1; \qquad \alpha_5 = -\omega_n^2 \tag{3.9}$$

$$\alpha_3 = -(2\xi - C_1(\xi + \sqrt{\xi^2 - 1}))\omega_n \tag{3.10}$$

$$\alpha_4 = -C_1(\xi + \sqrt{\xi^2 - 1})\omega_n \tag{3.11}$$

Controller Strategy	Characteristic
CC	Maintain desired velocity
ACC	Collision avoidance
CACC	Fixed small gap String stability

Table 3.1: Comparison between controller strategies

C_1 is a weighting factor between the acceleration of the leader and the preceding vehicle, which is set to 0.5 by default. ξ is the damping ratio and set to 1 by default. ω_n is the controller bandwidth and set to 0.2 Hz by default.

In the implementation of the CACC functionality, the interaction between CC and CACC depends on the distance between vehicles. If a vehicle is less than 20 meters from the preceding one, the vehicle follows instructions from CACC: $\ddot{x}_{des} = \ddot{x}_{CACC}$, otherwise, the policy is the same as ACC: $\ddot{x}_{des} = min(\ddot{x}_{CC}, \ddot{x}_{CACC})$.

3.2.5 Controller Comparison

We use Table 3.1 to summarize distinguished characteristics of each controller strategy. From the table we can see that different controller strategies have diverse design objectives.

3.3 Simulation Environment

With this controller and default parameters, we conduct an experiment of Leader Crash Attack with PLEXE [19] which is based on Veins [20] and further extends the interaction through the TraCI interface in order to fetch vehicles' data from SUMO [**?**] to be sent to other vehicles in the platooning to realize CACC. PLEXE combines the network and the mobility simulator by creating a network node in OMNeT++ for each vehicle traveling in SUMO. It provides a vehicular communication system based on IEEE 802.11p and a mobility model based on SUMO. When a vehicle moves, PLEXE repeates the movement in the corresponding OMNeT++ node by updating

the mobility model in SUMO. OMNeT++ framework implements platooning protocols and application logics, while SUMO realizes the actuation of the applications decisions as well as part of application logics.

We use PLEXE for all the (attack and defense) simulations carried out in this work. As mentioned earlier, PLEXE extends Veins which further extends the OMNeT++ network simulator and the SUMO mobility simulator. The coupling between the network and the mobility simulation framework is done through the TraCI interface which SUMO exposes. PLEXE extends the interaction through the TraCI interface in order to fetch vehicles' data from SUMO to be sent to other vehicles in the platoon to realize CACC. Platooning protocols and the application logic are realized in the OMNeT++ framework.

To simulate the adversary, we need to provide the functionality that informs the adversary vehicle how to launch the attack. To achieve this, we need to refer to application layer logic. There is a BaseApp in PLEXE which simply extracts data out of packets coming from the protocol layer and updates CACC data via TraCI if such data is coming either from the leading vehicle or from the preceding vehicle. SimplePlatooningApp extends BaseApp and it tells the vehicle to use the controller requested by the user. We modify the SimplePlatooningApp so that to let vehicles follow the instructions of what we want them to do. We make the platoon drive at a fixed speed of 100 km/h and fixed gap of 5 meters. At a certain time, we send a message to leader vehicle to tell it to stop. We set the deceleration of the leader car extremely large so that the speed can decelerate to zero in a minimal time interval. In this way, it acts just like a leader car crash. It allows us to see how the following vehicles will respond conducted by the CACC controller strategy.

CHAPTER 4

Safety and Security co-Design

Safety has a long tradition in many engineering disciplines and has had successful standardization efforts. In automotive systems, the international standard ISO 26262 is the state of the art standard for safety critical system development.

Automotive security has evolved quite recently with networked systems and concerns about privacy, data integrity, authenticity and protection. As long as safety critical systems were not networked, the two fields did not have to interact and as a result, the two domains have evolved separately so far with little overlap. As cyber-physical systems evolved into networked systems, security became a relevant issue for safety critical systems.

The Vehicle Cybersecurity Systems Engineering Committee of SAE has been working on J3061 Cybersecurity Guidebook for Cyber-Physical Vehicle Systems [22]. J3061 takes the initial step to standardize the cybersecurity risk assessment process for connected vehicles. However, this work is very much initial steps rather than a mature process. J3061 Cybersecurity Guidebook [22] is an overall guidebook on implementing cybersecurity for the entire vehicle. The safety-security co-design is being discussed in the secure software SAE committee at the moment and there is no final product yet. We are able to work with several key members of the SAE cybersecurity committee to understand the concepts and requirements as well as discuss the proposed safety-security engineering process.

4.1 Safety-Security co-Design

We propose a safety-security co-design engineering process which consists of four main steps: (1) Define the safety goal for the system; (2) Define attack model; (3) Derive security goals; (4) Derive functional security requirements.

Safety Goal. Safety is very important in automotive industry and therefore highly regulated. For end users, it means that users do not face any risk or danger coming from the motor vehicle or its spare parts. Unacceptable consequences for safety are loss of human life and injuries. The safety goal of individual vehicle is to protect users from injuries and life threatening risks. In our context, we set up the safety goal of vehicle platoon as avoiding car collisions that can cause human life and injuries. (Although there are different kinds of safety hazards such as no seatbelts, poor brakes and tires, non collapsible steering columns, doors that opened on impact, etc. In this paper, we consider safety hazards from nontraditional sources – cybersecurity attacks.)

Attack Model and Security Goal. Unlike safety, cybersecurity has a broader range of unacceptable consequences such as human life and injury (safety), human security, financial loss, loss of privacy [23, 24], etc. Figure 4.1 shows the interrelation of safety and security. From Figure 4.1, we can see that safety can be an objective (or impact) of a security attack. It can also be an unintended consequence caused by hardware or software bugs. Meanwhile, cyber attacks can have different impacts. The intersection part concerns both safety and security, or safety-related security risks, which is of interest of this paper.

To derive our attack model that lead to safety, we summarize various of attacks, targeting at automotive platoon systems, extensively studied by researchers in the literature and their corresponding possible consequences in Table 4.1. From the table, we can see that there are five attacks which can lead to car collisions, result in safety issues, and thus belong to the intersection in Figure 4.1. Our security goal is to develop a system that is resilient to these attacks.

Of the five attacks that can cause collisions, message spoofing and replay attacks belong to

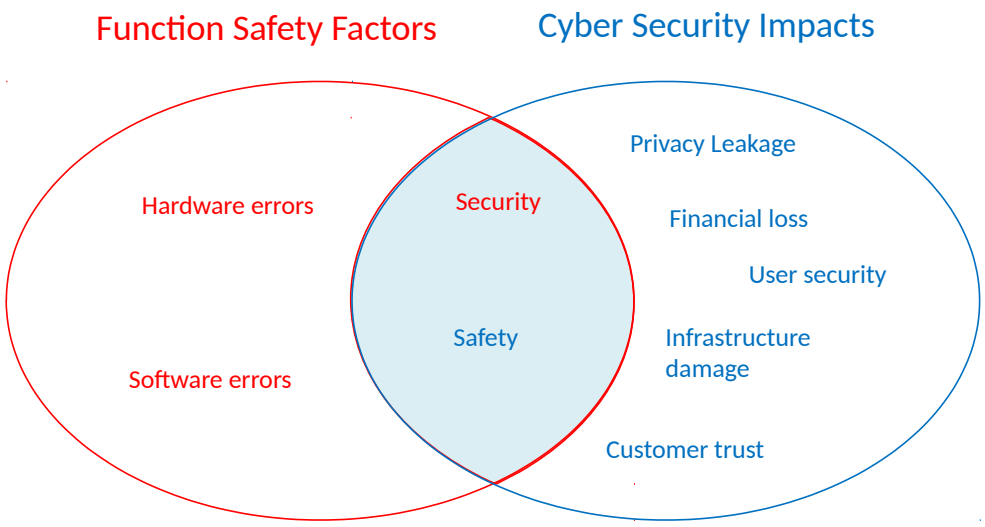

Figure 4.1: Interrelation of Safety and Security

outsider attacks. In this paper, we assume the platoon system uses standard security techniques such as digital signature, time-stamp or nonces to defend against these outsider attacks. In this way, we can narrow our focus on defense mechanisms against insider attacks. We briefly summarize the insider attacks as follows and users are referred to their original papers for details.

In *collision induction attack* [4], the driver (the attacker) broadcasts an acceleration message indicating that it is speeding up which causes the following vehicle to accelerate while actually the attacker starts to decelerate abruptly.

In *Message falsification attack* [25], the driver sends a false message to the following vehicle. Unlike collision induction attack, the attacker still moves according to the CACC controller strategy.

(We note the message falsification attack defined in [25] is actually the same as the mis-report attack defined in [4]. However, they are shown to have different negative consequences: collision or decreased performance. Our security analysis in Section 8 will explain this inconsistency.)

If a vehicle is a victim of *System tampering* attack, we mean an attacker is able to control the vehicle remotely through compromised hardware or software. The victim vehicle will behave like the one in either collision induction attack or message falsification attack, without the awareness and involvement of the driver. For us, we only need to focus on attack behaviors without worrying about who, the driver or a remote attacker, initiates the attack. Therefore in the following of the paper, we only consider collision induction attack and message falsification attack and ignore who initiates the attack.

Based on the discussion above, we derive our attack model as follows:

Adversary Model: We consider insider attacks that can lead to safety issues such as car crashes in this work. Attacks that result in different consequences such as system performance, driver privacy, financial loss, etc. are not considered in this paper as they can be treated in the regular way without considering safety. The adversary or the vehicle controlled by the adversary is part of the platoon system and thus is able to send valid V2V messages. However, there is no guarantee on the correctness of information in the messages it sends. Also the adversary does not need to follow the control law. The adversary is able to control one or more vehicles, including the leader, in the platoon. However, it cannot control all the radars or radar signals of vehicles in the platoon because of the line-of-sight requirement.

Scope of this work. Our adversary model defines the scope of this work: we only consider safety-related security risks in this work. For other security risks such as financial loss or privacy leakage, we understand they can be at least addressed in a parallel way.

Functional Security Requirements. From our analysis above, we can derive functional security requirements as follows:

- It shall not be able for an attacker to spoof a message;
- It shall not be possible to replay an old message;

Reference	Attack	Impact
[25]	Message falsification attack	Collision
	Message spoofing	Collision
	Message replay	Collision
	DoS (jamming)	Dissolved platoon
	System tampering	Collision
[4]	Collision induction attack	Collision
	Reduced headway attack	Decreased string stability
	Joining without radar	Decreased string stability
	Mis-report attack	Decreased performance
	Non-attack abnormalities	Decreased performance and string stability
[5]	Destabilization attack	Decreased string stability
	Platoon control taken attack	Dissolved platoon

Table 4.1: Attacks and impacts

- It shall not be possible for an attacker to broadcast a message with false information without being detected;
- The system shall be able to take a response action whenever such misbehavior is detected;
- The system shall ensure there is enough time for the system to respond.

4.2 Severity Analysis

The EU project EVITA provides a risk model to measure the security of in-vehicle systems [26]. The risk analysis rationale of EVITA is that as it is too costly to protect against every threat, it is necessary to rank risks in order to prioritize countermeasures. Risk associated with a security attack depends on (1) **severity** of impact and (2) **probability** of successful attack. In this section, we analyze the severity as well as the probability of platooning attacks by using the EVITA model.

In response to various safety risks, ISO 26262 severity classification defines four severity levels (S0, S1, S2 and S3) in terms of the estimated personal injury that could result from the risk. S0 refers to no injuries. S1 refers to light or moderate injuries. S2 means severe to life-threatening

injuries (survival probable). S3 means life threatening (survival uncertain) or fatal injuries. The EVITA model extends the ISO 26262 safety classification by including a fifth level S4 which means fatal injuries of **multiple vehicles** as cyber security attacks may have more widespread implication than unintended hardware or software bugs can cause.

In this way, we are able to define the severity of different kinds of collision attack. Leader crash attack belongs to S4 without doubt because it will lead to multi car crash and cause damage and even death to drivers.

Previous work [4] has shown that message falsification and collision induction attacks can result in serious safety issue. However, it is not clear the severity level of such attacks. To understand the severity level of a collision that is resulted from a cyber attack, we introduce a new attack called leader crash attack by extending the collision induction attack proposed in [4]. In the leader crash attack, the leading car stops suddenly (intentionally due to being remotely controlled by an attacker or not) and causes the following cars to crash over each other. This crash attack can be mounted by any insider, not just the leader, in the platoon. However it is very likely a crash attack induced by the leader can have the most severe consequence.

Previous work focuses on attacks from insiders and show that even insider attacks can cause significant damage in the CACC vehicle stream. Intuitively if the leader is malicious, it can cause the most severe damage to the platoon.

We firstly argue collision induction attack is very possible (**probability**). As shown in Section 2, it has been demonstrated successfully on several modern vehicle models that an attacker can totally control a vehicle by compromising its hardware or software locally or remotely through a wide range of attack vectors [9, 10, 11, 12]. When a leader or any insider of the platoon is compromised and can be remotely controlled, an attacker can issue an instruction to the victim vehicle to brake abruptly so that the following cars will crash into the front ones. The risk of insider crash attack will become more serious with the advancement in vehicle automation. If an insider car is a compromised *driverless* automated vehicle, such an attack can be mounted with

severe consequence at a low cost. Also, we do not exclude the case when the driver himself is reckless.

Any insider car can become malicious if its hardware or software is tampered. In current platoon system demonstrations like SARTRE, the leader vehicle is usually a truck with an experienced trained driver. However SARTRE does not focus on security and so it does not consider a malicious leader vehicle. However, the leading driver can be malicious in the real world. He just needs to hit the brake abruptly when the vehicle stream is on the highway with a high speed and minimal gap. Under this circumstance, the following vehicles will crash together for there is no enough safe distance for them to decelerate.

The risk of insider crash attack will become more serious with the advancement in vehicle automation. If the leader car is a compromised driverless automated vehicle, such an attack can be mounted with severe consequence at a low cost or effort. When the platoon is driving on the highway, the attacker just needs to remotely control the leader vehicle to brake abruptly so that the following cars will all crash into the front ones. Whether which kind of attack it is, it can cause severe accidents and endanger the driver's life.

We use the PLEXE simulator to demonstrate the consequence of this attack (**severity**). In this simulation, initially a platoon of four vehicles is driving at the speed of 100 km/h with a gap of 5 meters (we will use the same platoon as a concrete example throughout the rest of the paper). At the time of 50s, we instruct the leader vehicle to stop. We set the deceleration of the leader car extremely large so that the speed can decelerate to zero in a very short time interval. In this way, the leader vehicle acts just like it suddenly hits the brake or crashes into something like a wall so that it stops immediately. We see how the following vehicles will respond under the CACC controller strategy.

Mobility traces of the platoon are collected and shown in Figure 4.2. From the figure, we can see that following vehicles crash into preceding vehicles at 50.41s, 50.75s and 51.10s respectively. To obtain an insight of speed changing of the platoon in the crash, we utilize the statistics collected

from PLEXE which are shown in Figure 4.3. In Figure 4.3, Vehicle 0 with the red line is the leader vehicle. Vehicle 0 decelerates from 100 km/h (27.77 m/s) to 0 km/h in a very short time interval. The following vehicles are trying to prevent crash by decelerating, but the 5-meter gap is not long enough for them to fully stop before they crash into the car before it. The above three lines terminating at different time spot shows that each of them has crashed into the leader vehicle.

```
Loading configuration... done.
Loading net-file from 'freeway.net.xml'... done (14ms).
Loading done.
Simulation started with time: 0.00
Warning: Teleporting vehicle 'platoon.1'; collision with 'platoon.0', lane='edge_0_0_0', gap=-0.12, time=50.41 stage=1.
Warning: Vehicle 'platoon.1' ends teleporting on edge 'edge_0_1', simulation time 50.41.
Warning: Teleporting vehicle 'platoon.2'; collision with 'platoon.0', lane='edge_0_0_0', gap=-0.25, time=50.75 stage=1.
Warning: Vehicle 'platoon.2' ends teleporting on edge 'edge_0_1', simulation time 50.75.
Warning: Teleporting vehicle 'platoon.3'; collision with 'platoon.0', lane='edge_0_0_0', gap=-0.15, time=51.10 stage=1.
Warning: Vehicle 'platoon.3' ends teleporting on edge 'edge_0_1', simulation time 51.10.
```

Figure 4.2: Leader Crash Attack

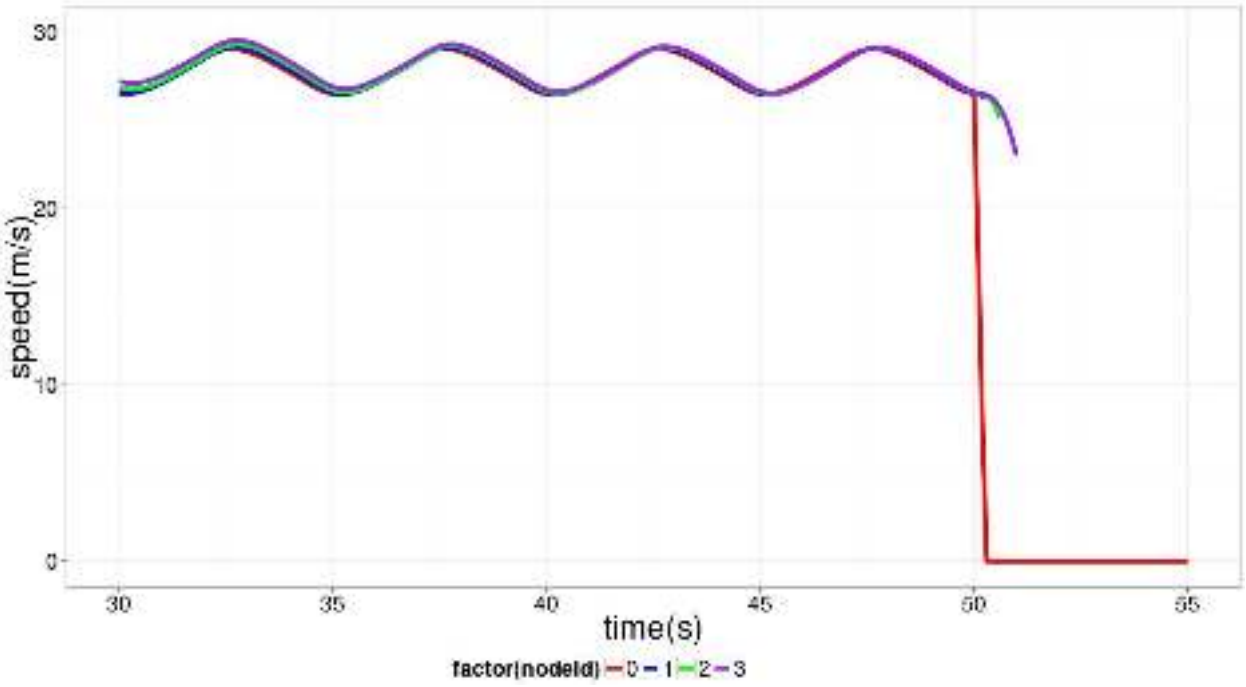

Figure 4.3: Speed Changes of Platoon during the Crash

More on severity. The above simulation clearly demonstrates that the leader car crash attack can potentially result in multiple car damage and life injuries and has the highest level of safety

severity. However, the maximum safety impact of security attack demonstrated is only a **local** event to several vehicles. We believe the worst security impact can potentially be *nation-wide* impacting thousands or millions of cars and suggest a new severity level of **S5: nation-wide wide spread and harmful impact**. For example, in the platoon context, suppose there is a security weakness that has an impact due to forged DSRC messages, also suppose future smart-phones are DSRC enabled and malware spread on smartphones, we can easily see a nation-wide attack platform to attack the platoon mechanism. Due to the severity of security attacks on platoon systems, we strongly argue the importance of designing safe and secure platoon systems.

CHAPTER 5

Safe Platooning:General Approach

Based on the safety-security co-design analysis, we propose a general approach to design a safe platooning. There are three steps to take in order to maintain safety and security. First, vehicles in the platoon need to keep a safe distance from preceding vehicle, so that when abnormal driving happens, they have enough distance to brake. Second, vehicles are supposed to detect security attacks while driving. These attacks include message falsification attack, collision induction attack, abnormal driving and so on. There are also many countermeasures to detect these attacks which will be discussed in the following part. Third, when abnormalities are detected, vehicles should switch to fail-safe scheme such as Adaptive Cruise Control (ACC) or Emergency Brake Assist (EBA) to avoid collision.

5.1 Safe Distance

Platoon is designed to keep a small distance between each vehicle so that it can increase the highway capacity. The platoon in SARTRE Project is driving at 90 km/h with a 4-meter gap between vehicles. Meanwhile, Energy ITS maintains a 80 km/h platoon with 4-meter gap. However, when accident happens, there is no enough distance for these vehicles to decelerate. As shown above, leader crash attack can cause multiple vehicle damage and life injuries.

For a safe platooning design, we need to maintain the safe distance to defend against extreme incidents. [27] presents a unique optimal control method of velocity and distance for platooning using model predictive control. Estimation of safe distance is also dependent on the fail-safe scheme

adopted. This paper aims to develop an optimal control system for safe distance of platooning. In out paper, we also want to identify the exact safe distance for platooning with the corresponding fail-safe scheme. We will use a concrete example to present the method of obtaining safe distance.

5.2 Attack Detection

To ensure safety, platoon has to detect various cyber attacks. In Bruce DeBruhl's paper [4], he proposes a set of insider attacks that can cause unexpected behaviors in platoons. Mis-report attack is the attack that sends false message to the following vehicle to increase the following distance of the preceding car. Collision induction attack can cause dangerous accidents by broadcasting an acceleration message indicating that they are speeding up, while the attacker starts to aggressively brake. The work of [4] suggests switching from CACC to ACC if a crash could happen. It focuses on the detection of false message attack from the preceding car. In the false message attack, a malicious preceding car driving at a low speed mis-reports it is driving at a high speed. The false message may mis-lead the following car to collide with the malicious preceding car. From the security angle, the proposed solution is to let each vehicle monitor the behavior of the preceding car. If the received status info from the preceding car is different from the one the following car calculates, the following car will think the preceding car may behave abnormally and switch to ACC. However it is not clear whether the switch from CACC to ACC can lead to a safe platoon.

In our paper, we will show that platoon with a safe distance is able to avoid collision in worst cases by switching to ACC. In our scheme, a vehicle concentrates on self-safety, calculates *its own safety status* (instead of predicting others' misbehavior) based on the context information and adjusts its next movement based on one criterion: whether it is safe to do so. If it senses the next step is not safe, the vehicle will switch from the cooperative driving CACC mode to the collision avoidance ACC mode. By centralizing on self-safety, our scheme achieves safety by implicitly defending against cyber attacks that could result in safety consequences.

5.3 Switch to Fail-safe Scheme

Fail-safe is a mechanism which is automatically triggered by failure that reduces or eliminates harm [28]. A fail-safe is not supposed to prevent failure but mitigates failure when it happens. For example, railway trains commonly have air brakes that get applied automatically when the main brake system fails to work. Flight control computers are typically designed with redundancy so that when one goes down another will continue to function. Similarly, platoon's safety is threatened by different kinds of cyber attacks and in some worst cases, such as leader crash attack, vehicles need to switch to fail-safe scheme to eliminate harm. Under this circumstance, vehicles are suggested to switch to ACC or EBA to avoid collision. Moreover, this defense mechanism can only succeed on one condition: there is a safe distance between vehicles. In the following section, we will use an example to show how safe distance can guarantee the safety of platoon and how to shorten the safe distance with attack detection.

CHAPTER 6

Safe Platooning: First Attempt

In this section, we propose a naive solution to deal with the worst case in platoon, such as leader crash attack. In the naive solution, we do not detect cyber attacks and we only rely on safe distance to ensure safety. Results show that relying on safe distance alone is not feasible for a safe platooning. We have to follow the three steps in general approach to design a safe platooning.

The platoon in this paper is traveling at 100km/h with a 5-meter gap. From the previous simulation on leader crash attack, we can see that when the leader vehicle in the platoon crashes suddenly, such a short distance between cars is not enough for the following vehicles to decelerate. To avoid collision, without changing the underlying CACC controller, the straightforward idea is to simply increase the distance between cars so that a car can stop before it crashes over the preceding car. Under this circumstance, we present a naive solution by increasing the gap between each vehicle in the platoon.

We use PLEXE to test this naive idea to find the required safe distance. In the SimplePlatooningApp, we set the constant space to a specific value at first by using the TraCI interface to update the CACC data. We increase the constant gap between vehicles step by step until we find that when the car-to-car distance is increased to 47 meters, the following vehicles will not crash into preceding vehicles. 47 meters equal to a headway of $T = 1.7s$ when the vehicle is traveling at 100km/h. Figure 6.1 shows speed changing of vehicles when the gap is increased to 47 meters. We can see the leader vehicle stops at 50.38s and the following vehicles stops at 56.16s, 58.42s and 60.9s. The last three vehicles set their accelerations according to the CACC controller strategy

during the crash.

Figure 6.2 shows the distance changing of each vehicle. Here the distance refers to the gap between the current vehicle and its preceding vehicle. So the figure only shows the distances of Vehicle 1, 2, and 3. We can see that initially the three following vehicles maintain a fixed gap of 67 meters with their processing cars. The gaps begin to decrease as vehicles decelerate. Vehicle 1 stops right after the leading vehicle with a distance of almost zero while Vehicle 2 and 3 stop with enough distance with proceeding cars.

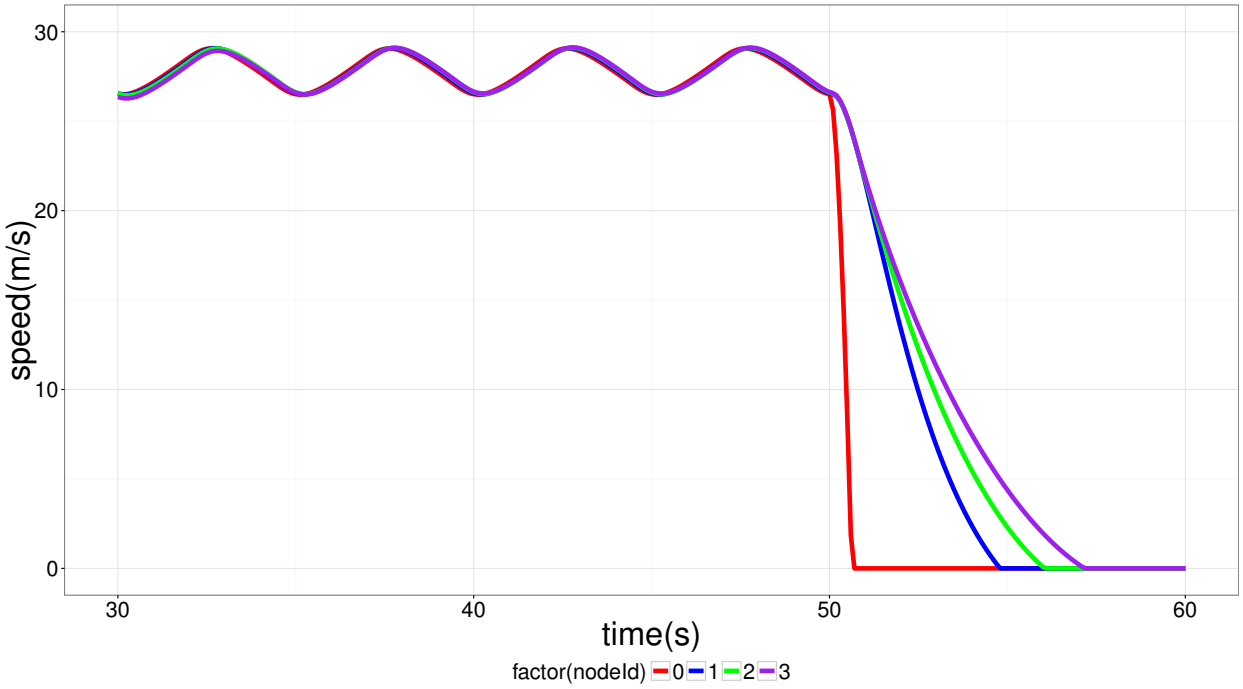

Figure 6.1: Naive Solution: Speed Changing

6.1 Theoretical Analysis of CACC Safe Distance

In order to prove that we find the correct safe distance for a platoon under the leader crash attack, we calculate the theoretical safe distance value by using MATLAB and do a cross check with our simulation result. In the CACC controller, the acceleration of each vehicle is calculated based on on

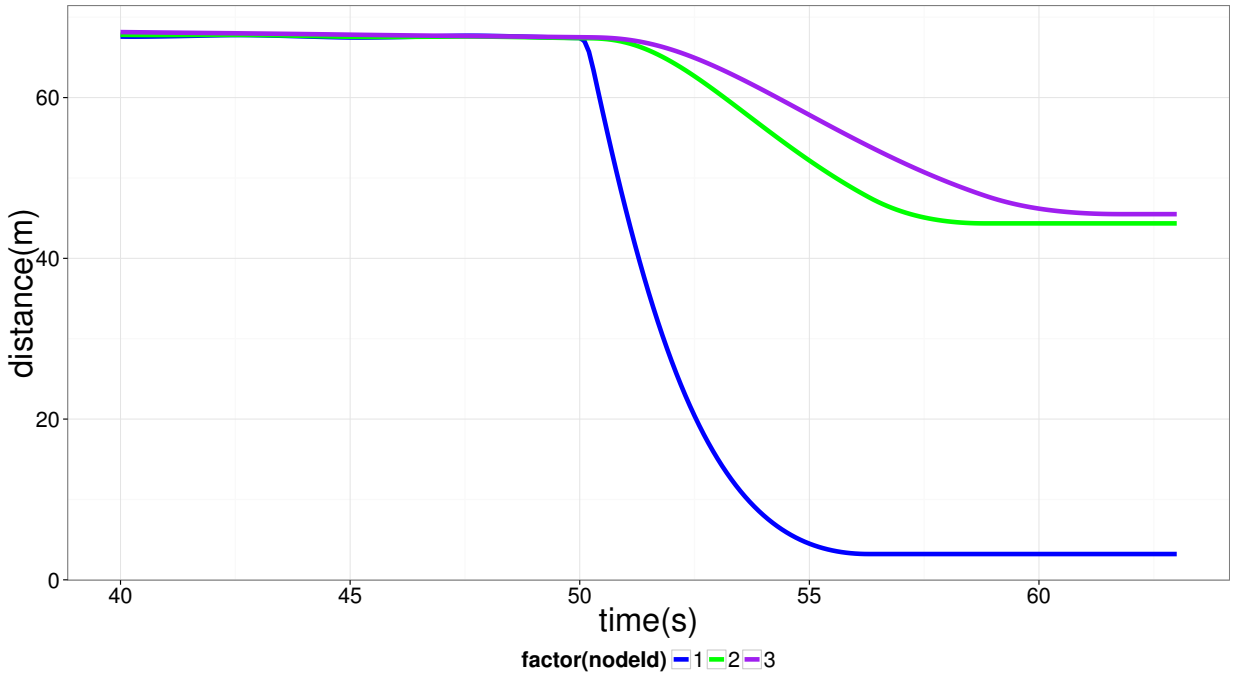

Figure 6.2: Naive Solution: Distance Changing

the leader vehicle and preceding vehicle. Vehicle 0 is leader vehicle and we study the performance of vehicle 1. If vehicle 1 does not crash into vehicle 0, the following two vehicles will not crash either as there is longer distance for them to decelerate.

In Equation (3.6), we set $\ddot{x}_0$ and $\dot{x}_0$ to 0 and other variables to their default values. x refers to location of the vehicle. $\dot{x}$ and $\ddot{x}$ refers to the speed and acceleration of the vehicle. In this way, we transfer Equation (3.6) to Equation (6.1).

$$\ddot{x}_1 = -0.4\dot{x}_1 - 0.04(x_1 - x_0 + l_0 + gap_{des}) \tag{6.1}$$

Obviously, Equation (6.1) is a second order differential equation and x_0, l_0, gap_{des} are constant values. x_0 is the location where the leader vehicle crashes. l_0 is the length of vehicle 0 and gap_{des} is the distance between two vehicles. By solving this equation, we acquire the relation between vehicle location x and time t. By differentiating the equation between vehicle location x and time

t, we can obtain the relation of a vehicle speed $\dot{x}$ and the time t. With the time t when the vehicle speed decreases to 0, we are able to obtain the location x where the vehicle stops. If location x is smaller than the location of the leader vehicle x_0, we say there is enough safe distance for the platoon.

Algorithm 1 Safe Distance Calculation

Input: Vehicle Location location, Vehicle Speed speed
Output: Final Position of the Vehicle

Use $dsolve$ method to find the relation between location x and time t. Original state is x(0) = location, Dx(0) = speed
$x = dsolve(D2x + 0.4Dx + 0.04x - 0.04(x_0 - l_0 - gap), t)$

Use $diff$ method to find the relation between speed and time
$speed = diff(x)$

Find the time when Vehicle 1 decelerates to 0
$speedChar = char(speed)$
$time = solve(speedChar,' t')$

Find the position where Vehicle 1 finally stops
$f = inline(x)$
$answer = f(time)$

Algorithm 1 is the MATLAB program which is used to estimate the theoretical safe distance a platoon needs to maintain to defend against a leader crash attack. From the statistics collected by OMNeT++, we can get the final position of leader vehicle x_0 which is 1432 meters. The speed of Vehicle 1 $Dx(0)$ in initial state is 27.77 m/s (100 km/h). The default length of vehicle l_0 is 4 meters. Taking the vehicle length into consideration, we hope that the final position of Vehicle 1 which is the output of the algorithm should be 1428 meters. By adjusting the input gap_{des} and the original location of Vehicle 1 $x(0)$, we need to achieve $gap_{des} + x(0) = x_0 - l_0$. Therefore, the gap_{des} is the exact safe distance for the platoon to decelerate during a leader crash attack. After executing the algorithm, we find out that when the gap_{des} equals 51 meters and the position of Vehicle 1 $x(0)$ equals 1377 meters where it starts to decelerate, then it will stop at the distance of

1428 meters. In this way, the safe distance is the difference between two positions which is 51 meters.

From the discussion above, we can see that CACC does not help in achieving better safety under urgent situations. Although increasing vehicle distance can help to achieve safety, this naive solution kills the space efficiency a platoon brings as a headway of $T = 1.7s$ is enough for a human driver to stop safely from a speed of 100km/h.

CHAPTER 7

Proactive Safe Platooning

As shown in previous section, the naive approach to safe platooning does not work as it totally removes the space efficiency which is the major reason why vehicle platoon is designed for. This motivates us to design a secure and safe CACC algorithm which achieves safety and security without losing space efficiency. Therefore, in this section, we follow the three steps in general approach to design a safe and feasible platooning.

7.1 Attack Detection

[4] focuses on the safe status of preceding vehicle. It compares the expected behavior and measured behavior of front car, if the error is larger than a specific threshold, current car switches to ACC controller strategy. The parameters of preceding car they choose are acceleration and velocity. Different from previous work, we concentrate on current vehicle's safety status. The parameters we choose are acceleration and distance. The reason why we don't consider velocity is that for ego vehicle, its speed is able to vary all the time and change abruptly in accidents. So we cannot use velocity to determine whether current vehicle's status is safe or not. In the following, we will show how to use acceleration and distance to ensure both safety and security.

7.1.1 Acceleration & Distance

For acceleration, to detect cyber attacks and avoid collision in CACC function, our idea is to include ACC in the design of the CACC function as ACC is designed for collision avoidance. It relies on range sensors like radar or laser scanner to estimate the distance to the preceding car. Real time distance information can be further used to estimate the preceding car's velocity and acceleration. We utilize ACC acceleration as the baseline for safe situation determination. In normal situations, the acceleration rate calculated by CACC is less than the acceleration rate calculated by ACC (i.e., speed change in CACC is usually more smoothly than that in ACC for improved user comfort). Let Δ denote the fluctuation range of ACC acceleration. When the acceleration rate of a vehicle falls into this range $[ACC - \Delta, ACC + \Delta]$ (here, ACC refers to acceleration in ACC controller strategy), it indicates there is no immediate crash threat. On the contrary, if the acceleration of a vehicle is out of the range, it indicates abnormal situation and we need to switch to fail-safe scheme to avoid collision.

Here are some details on how range is defined. From Equation (3.6) and Equation (3.2), the accelerations of CACC and ACC in normal state can be calculated. By setting parameters to their default values, we are able to get Equation (7.1) and Equation (7.2). Actually, we can calculate the difference between CACC and ACC acceleration anytime and then we can determine the threshold exactly the maximum difference.

$$\begin{aligned}\ddot{x}_{i\_cacc} = 0.5\ddot{x}_{i-1} + 0.5\ddot{x}_0 - 0.4\dot{x}_i + 0.3\dot{x}_{i-1} + 0.1\dot{x}_0 - 0.04x_i \\ +0.04x_{i-1} - 0.04l_{i-1} - 0.04gap_{des}\end{aligned} \tag{7.1}$$

$$\ddot{x}_{i\_acc} = -\frac{1}{T}(\dot{x}_i - \dot{x}_{i-1} + 0.1x_i - 0.1x_{i-1} + 0.1l_{i-1} + 0.1T\dot{x}_i) \tag{7.2}$$

Let us define

$$\Delta = max(|\ddot{x}_{i\_cacc} - \ddot{x}_{i\_acc}|) \tag{7.3}$$

In normal situation, the platoon is driving with a fixed speed and constant gap between cars. Therefore, we can assume that $\ddot{x}_{i-1}$ and $\ddot{x}_0$ are 0 with $\dot{x}_i$, $\dot{x}_{i-1}$ and $\dot{x}_0$ are equal. Meanwhile we have $\dot{x}_{i-1} - \dot{x}_i = gap_{des} + l_{i-1}$ so that the acceleration of CACC $\ddot{x}_{i\_cacc}$ is 0. For ACC, T is headway and then we have $\ddot{x}_{i\_acc} = -\frac{1}{T}(0.1T\dot{x}_i - 0.1gap_{des})$. Finally, the guideline for range is achieved with $\Delta = max(|\frac{1}{T}(0.1T\dot{x}_i - 0.1gap_{des})|) = 0.1max(\dot{x}_i) - \frac{0.1gap_{des}}{T}$. $max(\dot{x}_i)$ is the max speed of Vehicle i. This max speed is not the maximum mechanical speed a car can have. Instead, it is the max speed set by automatic driving. Usually the max speed in automatic driving is less than the actual max speed a car can have because passenger comfort is usually an important factor in automatic driving.

We analyze the relationship between accelerations calculated from CACC and ACC in normal situation (when there is no attack) and find out that difference between these two acceleration values acts well as the range. When the CACC acceleration is within the range of ACC acceleration, we follow CACC controller strategy to maintain string stability. Otherwise, we should switch from CACC to ACC to maintain safety.

Based on this observation, we can set the platoon to CACC mode when the difference between CACC and ACC accelerations is under the range (normal situation) while switch the platoon to ACC mode when the difference is larger than the range (abnormal situation).

Since we can compute the acceleration of ACC and CACC at the same time, we can combine them together to present a new driving mode which is called Proactive CACC Algorithm. On one hand, we can utilize the received V2V messages to compute the acceleration for CACC. On the other hand, we can use radar data to compute the acceleration for ACC. Meanwhile, we define a range for Proactive CACC Algorithm. In normal cases, the difference between two kinds of acceleration will be within the range. Therefore, we set the platoon to CACC mode so that it can maintain string stability. In abnormal cases, the difference between CACC acceleration and ACC acceleration will be larger than the range. Then we switch the platoon to ACC mode to ensure safety. With this defense mechanism, we can achieve string stability and safety at the same time.

For distance, we use a straightforward way to detect cyber attacks. When platooning is driving in normal situation, it maintains a fixed desired gap. The idea is to check whether the gap between vehicles is smaller than desired value, if it is, then we identify a cyber attack.

7.1.2 Switch to Fail-safe Scheme

Once cyber attacks are mounted on platoon, we need to switch to fail-safe scheme to reduce or eliminate harm. When crash happens, we believe in such urgent situation, autonomous driving responds quicker than human drivers. So we choose to switch cooperative CACC to non-cooperative ACC. There might be other autonomous emergency plans which we will take further investigation in the future. In the following, we will show the fail-safe schemes of different parameters (acceleration,distance) respectively.

Acceleration & Distance

The key point of Proactive CACC Algorithm is that whether the difference of CACC and ACC accelerations is within the range. From the discussion above we know that, in normal situation, the acceleration to CACC is 0 while the acceleration to ACC is $\ddot{x}_{i\_acc} = -\frac{1}{T}(0.1T\dot{x}_i - 0.1gap)$. The range defined is $0.1max(\dot{x}_i) - \frac{0.1gap}{T}$. Hence, the difference of two accelerations is within the range. Otherwise any abnormal event happens, the difference is larger than the range which will trigger the defense mechanism.

Based on the above analysis, for acceleration parameter, we propose Proactive CACC Algorithm which is shown as Algorithm 2. In our algorithm, a vehicle calculates desired acceleration in both CACC and ACC controller strategy. It switches to ACC if the defense mechanism ($|\ddot{x}_{i\_cacc} - \ddot{x}_{i\_acc}| > \Delta$) is triggered.

In the Proactive CACC Algorithm, $\_cc$, $\_acc$ and $\_cacc$ are the methods computing the desired acceleration based on CC, ACC and CACC controller strategy respectively. $gap2pred$ is the dis-

Algorithm 2 Proactive CACC Algorithm

```
1: Input: parameters (vehicle information)
2: Output: desiredAcceleration
3:
4: ccAcceleration = _cc(parameters)
5: accAcceleration = _acc(parameters)
6: caccAcceleration = _cacc(parameters)
7:
8: if |caccAcceleration - accAcceleration| <= Δ then
9:     desiredAcceleration = caccAcceleration
10: else
11:     desiredAcceleration = accAcceleration
12: end if
13: if gap2pred >= 20 then
14:     desiredAcceleration = min(desiredAcceleration,ccAcceleration)
15: end if
```

tance to the preceding vehicle. Δ is the threshold that we use to select the proper controller strategy between ACC and CACC.

For distance, the fail-safe scheme is to switch to ACC when the gap between vehicles is smaller than the desired value, otherwise platoon still follows CACC to maintain fixed gap and string stability.

7.1.3 Safe Distance

Safe distance is the prerequisite in general approach for designing a safe platooning. Here we present the safe distances under different fail-safe schemes and attack detection mechanisms.

Acceleration

To demonstrate whether the proposed Proactive CACC Algorithm works and to find the safe distance, we conduct a simulation with the example platoon under leader crash attack. When we set the range value to 0.2 m/s^2, the platoon is safe with a safe distance of 9 meters (0.32s).

We modify the PLEXE to make sure that platoon can switch between CACC and ACC when-

ever it needs. MSCFModel_CC is a vehicle driving model which implements the CACC, ACC and other control strategies, like Cruise Control (CC) and human driving mode. Within the _v() method in MSCFModel_CC, it computes the desired acceleration of each type of controller strategy and then choose the requested one. When it comes to CACC controller strategy, we use CACC proactive algorithm to calculate the desired acceleration. If the difference between CACC and ACC acceleration is within the range Δ, we return CACC acceleration, otherwise we return ACC acceleration. In the experiment, we find out that when we set the range to 0.2 m/s^2, we can achieve string stability and safety both. At the same time, we also achieve a much shorter safe distance which is 9 meters (0.32s).

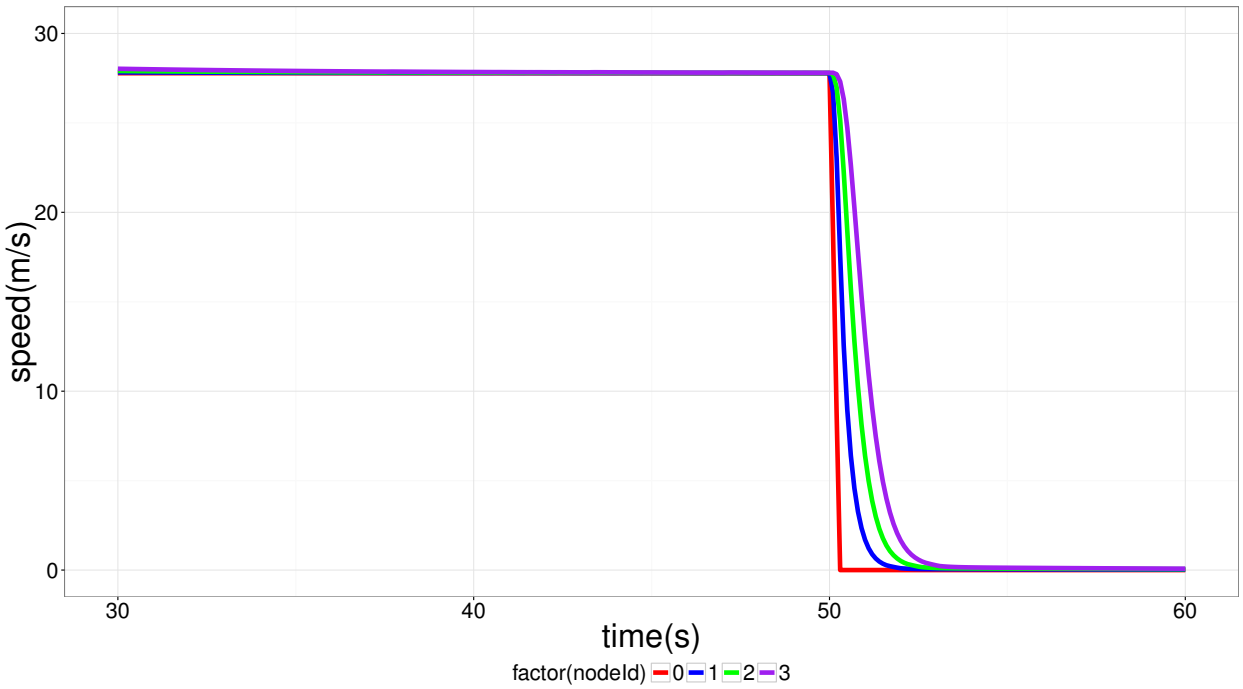

Figure 7.1: Speed Changes of Platoon : Acceleration

From Figure 7.1, we can see that all the vehicles maintain the same velocity. Meanwhile, the acceleration of platoon is a constant value. Therefore, $\ddot{x}_{i\_cacc} = 0$. The highest speed $\dot{x}_i$ is 27.78 m/s and the gap in this experiment is 9 meters (0.32s). Therefore, we obtain $\Delta = 0.22\ m/s^2$ which is close to our simulation result. So our algorithm achieves safety and efficiency. In the next

section, we prove its security under the two insider attacks.

Distance

For distance, we also conduct an experiment to show how this method defend against leader crash attack. The experiment environment is the same as above setting. We can see from Figure 7.2 that result is the same too. The safe distance is 9 meters (0.32s).

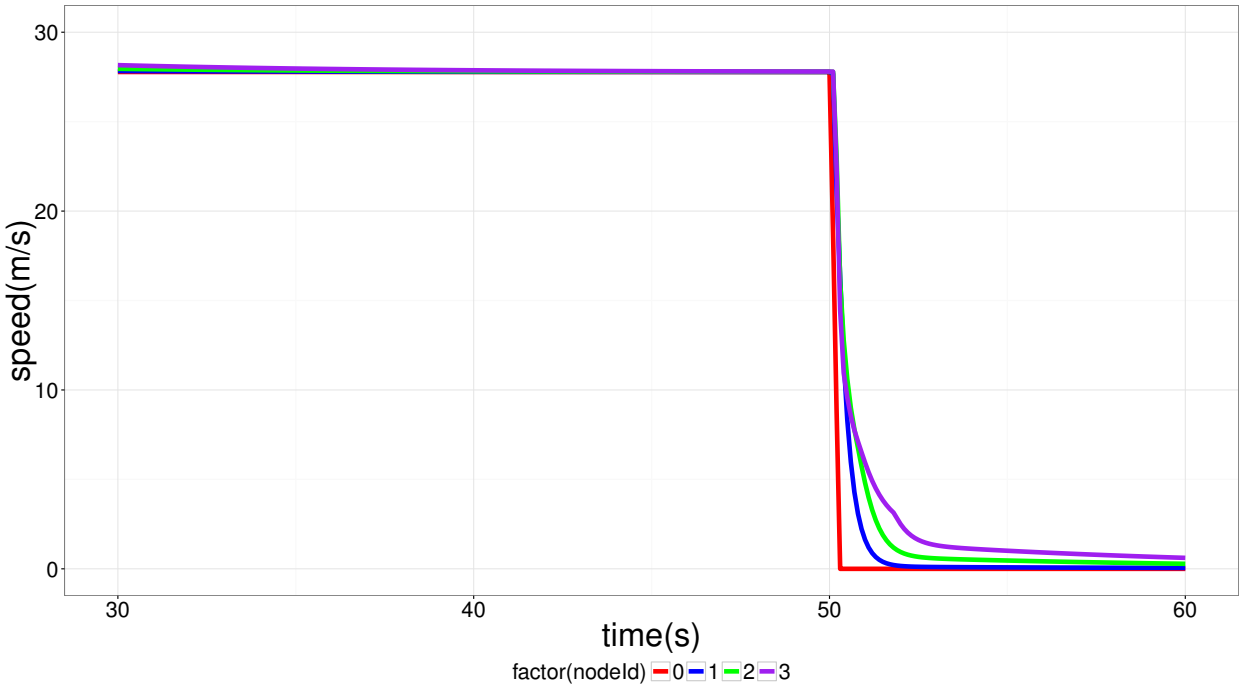

Figure 7.2: Speed Changes of Platoon : Distance

Comparison

In this paper, we recommend using acceleration defense mechanism to defend against attacks rather than distance defense mechanism. As shown in Table 7.1, when cyber attack happens, acceleration defense mechanism responds immediately, namely, the difference between CACC and ACC acceleration will be larger than the specific range. For distance defense mechanism, it

Attack Defense	Response Time	Leader Crash	Collision Induction	Message Falsification
Acceleration	Fast	Yes	Yes	Yes
Distance	Slow	Yes	Yes	No

Table 7.1: Comparison between acceleration and distance defense mechanism

will only be triggered when the vehicle gap is decreased. In the leader crash attack which starts at 50.00s, the ego vehicle switches to ACC at 50.02s by using acceleration defense mechanism and 50.07s by using distance defense mechanism. Furthermore, the kinds of cyber attacks can be defended by distance defense mechanism is limited. For example, in message falsification attack, the malicious vehicle broadcasts false message to tell current car to decelerate. Using distance defense mechanism cannot deal with such situation, but using acceleration defense mechanism can defend against it. Therefore, for the safe platooning design, we recommend using acceleration defense mechanism.

CHAPTER 8

Security Analysis

Platoon relies heavily on broadcasting messages via wireless communication to maintain a close gap and string stability. However, the messages are easy to be compromised and the following vehicles can't receive the correct signals to make the right maneuver decisions. Once the platoon is driving on the highway with a minimal gap, security threats from message falsification attack can cause severe accidents and endanger drivers' lives. The Proactive CACC Algorithm we proposed can completely defense diverse message attacks.

In this chapter, we analyze the security of the proposed proactive safe CACC algorithm in terms of resilience to the collision induction attack and message falsification attack which may lead to car collisions. We want to show that such an attack will cause a big difference between CACC and ACC acceleration values and thus trigger the vehicle to switch from CACC to ACC in order to avoid collision. For each attack type, we will provide a theoretical analysis on its resistance first and then demonstrate the effectiveness of the proposed scheme through simulation by using a concrete platoon example.

8.1 Security Resistance to Collision Induction Attack

8.1.1 Theoretical Analysis

In collision induction attack [4], an attacker broadcasts an acceleration message indicating that it is speeding up which causes the following vehicle to accelerate while actually the attacker starts to

decelerate abruptly. With platoon driving on highway with a close gap, this is very likely to lead to dangerous accidents. In order to prove that our proactive safe CACC algorithm can really defend against Collision Induction Attack, we determine to perform theoretical analysis of Collision Induction Attack.

Assume initially the platoon is already formed and traveling at a fixed velocity with a fixed gap gap_{des} between cars. In this situation, the acceleration of platoon is $\ddot{x}\_cacc = 0$. Every vehicle broadcasts its current speed ($\dot{x}$) and acceleration ($\ddot{x}\_cacc$) to its following car. Suppose Vehicle $i-1$ is the attacker and it begins to mount collision induction attack at certain time. It decelerates at a fixed deceleration rate and at the same time it sends false state information (speed and acceleration) to Vehicle i. We consider two cases of false state information. In Case 1, the attacker tells the following car that it is traveling the same as before with the same velocity and acceleration rate of 0 while it is actually decelerating. In Case 2, the attacker tells the follower that it is speeding up with a non-zero positive acceleration rate. In the following, we will show that the attack in both cases will trigger our defense mechanism. The following car, Vehicle i, is able to switch from CACC to ACC.

First of all, we show that Case 1 will trigger defense mechanism so that Vehicle i will switch from CACC to ACC. CACC makes use of broadcasting message to calculate the acceleration Vehicle i should take, while ACC utilize the information detected by radar to compute the corresponding acceleration. The condition whether we should switch on ACC depends on if the difference between two accelerations is beyond range.

Assuming the platoon is already driving at a fixed speed with a fixed gap, as shown in Chapter 7, we have $\Delta = max(|\frac{1}{T}(0.1T\dot{x}_i - 0.1gap_{des})|) = 0.1\dot{x}_i - \frac{0.1gap_{des}}{T}$.

In Case 1, the attacker $i-1$ pretends it is driving at the same parameters as before. As inputs to Vehicle i's CACC algorithm remain the same, i.e., $\ddot{x}_{i\_cacc} = 0$ in our assumed setting. Vehicle i

calculates its ACCC rate by using Equation (7.2):

$$\begin{aligned}\ddot{x}_{i\_acc} &= -\frac{1}{T}(\dot{x}_i - \dot{x}_{i-1} + 0.1x_i - 0.1x_{i-1} + 0.1l_{i-1} + 0.1T\dot{x}_i) \\ &= -\frac{1}{T}(\dot{x}_i - \dot{x}_{i-1}) - \frac{0.1(x_i - x_{i-1} + l_{i-1})}{T} - 0.1\dot{x}_i \\ &= -\frac{1}{T}(\dot{x}_i - \dot{x}_{i-1}) + \frac{0.1gap'}{T} - 0.1\dot{x}_i\end{aligned} \tag{8.1}$$

Hence we have:

$$|\ddot{x}_{i\_cacc} - \ddot{x}_{i\_acc}| = \frac{1}{T}(\dot{x}_i - \dot{x}_{i-1}) - \frac{0.1gap'}{T} + 0.1\dot{x}_i \tag{8.2}$$

When Vehicle $i-1$ starts to mount collision induction attack, it broadcasts message to Vehicle i while decelerating. After a message broadcasting time interval, Vehicle i receives the message and begins to adjust its movement. During that time interval, because the attacker Vehicle $i-1$ decelerates and the gap decreases, therefore, we have $\dot{x}_i > \dot{x}_{i-1}$ and $gap' < gap_{des}$ where gap' is the latest measured gap which Vehicle i measures using a radar sensor. It is obvious the difference between $\ddot{x}_{i\_cacc}$ and $\ddot{x}_{i\_acc}$ is larger than Δ , which will trigger the defense mechanism and Vehicle i will follow the instructions of ACC in the next step.

The above analysis can be easily extended to analyze Case 2. As Vehicle i will receive a message from the attacker to speed up, so we have $\ddot{x}_{i\_cacc} > 0$. $\ddot{x}_{i\_acc}$ is calculated the same as in Equation (8.1): $\ddot{x}_{i\_acc}$ is negative and its absolute value is larger than $THRESHOLD$. Therefore, the absolute difference between CACC and ACC acceleration is larger than $THRESHOLD$. This will trigger our defense mechanism and Vehicle i will follow ACC in the next step. .

8.1.2 Attack & Defense Simulation

We conduct experiments to show that the proposed Proactive CACC Algorithm can defend against Collision Induction Attack. In the experiment, Vehicle 1 starts to send false messages to Vehicle 2 at 50.0 second. Instead of sending its real velocity to Vehicle 2, Vehicle 1 chooses to broadcast

a velocity value that is twice of its real speed so that Vehicle 2 will follow the message to speed up. Meanwhile, Vehicle 1 decelerates from 100 km/h (27.77 m/s) to 80 km/h (22.22 m/s) at a deceleration of 9 m/s^2. In order to show Proactive CACC Algorithm can help platoon maintain string stability, we make the platoon velocity oscillates at a frequency of 0.2 Hz with an average speed of 100 km/h (27.77 m/s).

When the attacker decreases its speed from 100km/h to 80km/h (in contrast to the leader crash attack where the attacker decreases its speed from 100km/h to 0), our defense mechanism can prevent collision from happening even when the platoon is with a 5-meter gap. The simulation is completed by modifying MSCFModel_CC and SimplePlatooningApp. In MSCFModel_CC, we pass false messages from Vehicle 1 to Vehicle 2 and in SimplePlatooningApp we instruct Vehicle 1 to decelerate. The range is set to 1.6 m/s^2 to achieve string stability.

Figure 8.1: Collision Induction Attack Defense

We run two simulations. The first simulation is to show the vulnerability of the platoon under the collision induction attack when there is no defense mechanism enforced. The second simulation is

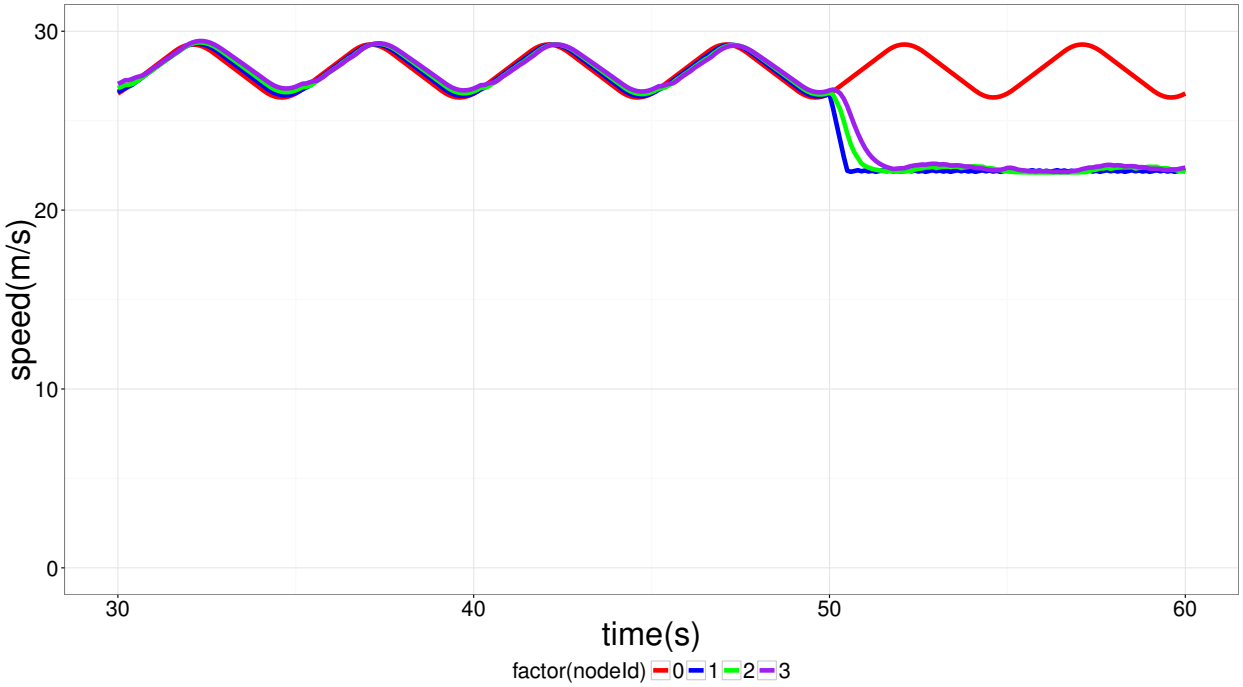

Figure 8.2: Collision Induction Attack Defense

to show how effective of the proposed Proactive CACC Algorithm in defending against the attack. From Figure 8.1 we can see that, vehicle 2 follows vehicle 1's false message to speed up while vehicle 1 is actually decelerating, so vehicle 2 and vehicle 3 crash into vehicle 1. Speed changes of vehicles in the second simulation are plotted in Figure 8.2. From the figure, we can see following vehicles 2 and 3 can successfully avoid collisions even when the platoon is under the mis-report attack. They switch to ACC immediately after the attacker decelerates and detach themselves from the platoon. Eventually Vehicles 1, 2, and 3 reach to ACC stability by following the attacker's speed.

8.2 Security Resilience to Message Falsification Attack

8.2.1 Theoretical Analysis

Message falsification attack [25] or mis-report attack as known in [4] is an attack where a vehicle broadcasts a message with false information. It could be done by an insider attacker who does not trust the CACC controller strategy. When the platoon is driving on the highway with a minimal gap, the larger the gap between current car and following car is, the safer the driver will feel. Under this circumstance, the driver may send a false message to the following vehicle to tell it to keep a smaller velocity compared with the actual platoon speed so that the gap will be enlarged. Unlike collision induction attack, the attacker still moves according to the CACC controller strategy.

The work of [25] shows that message falsification attack can result in car collision while the work of [4] shows such attack only decreases system performance. It is interesting to understand what causes this inconsistency in attack impact. Our analysis shown below explains the cause of this inconsistency. Whether a message falsification attack can cause a collision or not depending the level of cheating on its actual speed.

However, when the vehicle trusts the false message and starts to decelerate to a lower speed, the rest following cars may not have enough time to react and crash into the preceding vehicle. Therefore, the False Message Attack could also cause severe collision and endanger driver's life.

Assume initially the platoon is already formed and traveling at a fixed speed with a fixed gap. Suppose Vehicle $i-1$ is the attacker and it wants to mount mis-report attack. It maintains the speed while broadcasting false message to Vehicle i to tell it to slow down. The attacker defines a **mis-report percentage** $\alpha \in [0,1]$ and then send the false speed $\alpha \dot{x}_{i-1}$ to Vehicle i. The mis-report percentage α determines the level of cheating on its speed. Apparently the smaller α is, the more the attacker cheats on its speed. In the following, we will show that depending on the cheating value of α, mis-report attacks with small α values (high cheating levels) will trigger our defense mechanism while mis-report attacks with large α values (low cheating levels) will only decrease

platoon performance.

Since we assume the platoon is driving at a fixed speed in initial stage, we can assume that all the vehicles' speed are the same ($\dot{x}_i = \dot{x}_{i-1} = \dot{x}_0$) initially. After a simulation time step, Vehicle i receives a false message from the attacker Vehicle $i-1$. It calculates the accelerations by using both CACC and ACC. From Equation (7.1) and Equation (7.2), we can get

$$\begin{aligned} \ddot{x}_{i\_cacc} &= -0.3\dot{x}_i + 0.3\alpha\dot{x}_{i-1} - 0.04(x_i - x_{i-1}) \\ &\quad - 0.04(l_{i-1} + gap_{des}) \\ &= -0.3\dot{x}_i(1-\alpha) - 0.04(gap_{des} - gap') \end{aligned} \tag{8.3}$$

$$\begin{aligned} \ddot{x}_{i\_acc} &= -\frac{1}{T}(\dot{x}_i - \dot{x}_{i-1} + 0.1x_i - 0.1x_{i-1} + 0.1l_{i-1} + 0.1T\dot{x}_i) \\ &= -\frac{1}{T}(\dot{x}_i - \dot{x}_{i-1}) - \frac{0.1(x_i - x_{i-1} + l_{i-1})}{T} - 0.1\dot{x}_i \\ &= -0.1\dot{x}_i + \frac{0.1gap'}{T} \end{aligned} \tag{8.4}$$

So we have

$$\begin{aligned} \ddot{x}_{i\_cacc} - \ddot{x}_{i\_acc} = -0.2\dot{x}_i + 0.3\alpha\dot{x}_{i-1} - 0.04(gap_{des} - gap') \\ -\frac{0.1gap'}{T} \end{aligned} \tag{8.5}$$

Equation (8.5) is linear function of α, so there exists a value α_1 such that when $\alpha = \alpha_1$, the left side of the equation becomes $\ddot{x}_{i\_cacc} - \ddot{x}_{i\_acc} = 0$ or $\ddot{x}_{i\_cacc} = \ddot{x}_{i\_acc}$. When $\alpha > \alpha_1$, it is easy to prove that $|\ddot{x}_{i\_cacc} - \ddot{x}_{i\_acc}| < \Delta$, and it will not trigger the defense mechanism. Now we consider the opposite situation when $\alpha < \alpha_1$. We have

$$\begin{aligned} |\ddot{x}_{i\_cacc} - \ddot{x}_{i\_acc}| - \Delta = 0.1\dot{x}_i - 0.3\alpha\dot{x}_{i-1} \\ +0.04(gap_{des} - gap') + \frac{0.1gap' + 0.1gap_{des}}{T} \end{aligned} \tag{8.6}$$

Similarly there exists a value α_2 such that when $\alpha = \alpha_2$, the left of the equation $|\ddot{x}_{i\_cacc} - \ddot{x}_{i\_acc}| - \Delta = 0$ or $|\ddot{x}_{i\_cacc} - \ddot{x}_{i\_acc}| = \Delta$. In this way, it is easy to see that when $\alpha < \alpha_2$, we have

$|\ddot{x}_{i\_cacc} - \ddot{x}_{i\_acc}| > \Delta$ and it will trigger the defense mechanism.

In summary, whether a mis-report attack can trigger the defense mechanism depends on the level it wants to cheat on its speed. When $\alpha < \alpha_2$, a mis-report attack will trigger our defense mechanism. When $\alpha > \alpha_2$, which means a less severe mis-report attack, the attack only breaks the string stability of the platoon, leads to platoon oscillation, and thus decreased platoon performance.

8.2.2 Attack & Defense Simulation

Same as in the defense simulation on collision induction attacks, we run two simulations to show how a platoon will respond with and without applying our defense mechanism. In the simulation, the attacker Vehicle 1 does not send its real velocity. It broadcast a false speed that is half of its true speed. That is $\alpha = 0.5$.

Figure 8.3: Mis-report Attack Defense

Simulation result is shown in Figure 8.3 and Figure 8.4. In Figure 8.3, vehicle 2 follows vehicle 1's false message to slow down. Then vehicle 3 crashes into vehicle 2. The reason why vehicle

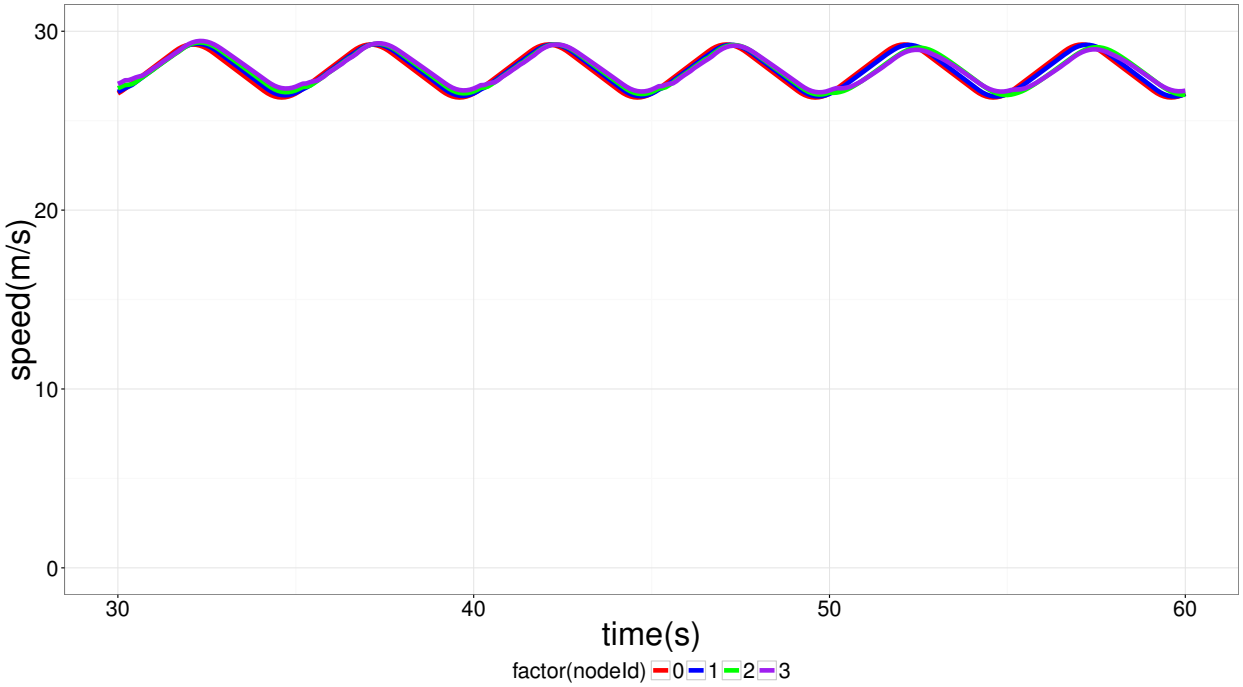

Figure 8.4: Mis-report Attack Defense

2 speed up later is that vehicle 2 has been detached from platoon. Vehicle 2 then follows CC algorithm instead of CACC algorithm and maintain the desired velocity. From Figure 8.4, we can see mis-report attack has no impact on platoon when platoon is equipped with our proactive algorithm.

CHAPTER 9

Source Codes of Experiment

Listing 9.1: MSCFModel_CC Source Codes

```
  SUMOReal
MSCFModel_CC::_v(const MSVehicle* const veh, SUMOReal gap2pred,
   SUMOReal egoSpeed, SUMOReal predSpeed, SUMOReal desSpeed, enum
   CONTROLLER_INVOKER invoker) const {
   //acceleration computed by the controller
   double controllerAcceleration;
   //acceleration actually actuated by the engine
   double engineAcceleration;
   //speed computed by the model
   double speed;
   //acceleration computed by the Cruise Control
   double ccAcceleration;
   //acceleration computed by the Adaptive Cruise Control
   double accAcceleration;
   //acceleration computed by the Cooperative Adaptive Cruise Control
   double caccAcceleration;
```

```
//variables needed by CACC
double predAcceleration, leaderAcceleration, leaderSpeed;
bool debug = true;
// Add these codes to launch leader crash attack
if(MSNet::getInstance()->getCurrentTimeStep() >=
   string2time("50.00") && veh->getID() == "platoon.0") {
            if(egoSpeed > 0) setFixedAcceleration(veh, 1, -20);
            else setFixedAcceleration(veh, 1, 0);
}
// End of codes
// Add these codes to launch collision induction attack
if(MSNet::getInstance()->getCurrentTimeStep() >=
   string2time("50.00") && veh->getID() == "platoon.1") {
            if(egoSpeed > 80/3.6) setFixedAcceleration(veh, 1, -9);
            else if(egoSpeed < 80/3.6) setFixedAcceleration(veh, 1,
               1.5);
            else setFixedAcceleration(veh, 1, 0);
}
// End of codes
VehicleVariables* vars = (VehicleVariables*)
   veh->getCarFollowVariables();
if (vars->activeController != Plexe::DRIVER &&
   vars->useFixedAcceleration) {
   controllerAcceleration = vars->fixedAcceleration;
}
else {
   switch (vars->activeController) {
```

```
case Plexe::ACC:
   ccAcceleration = _cc(veh, egoSpeed, vars->ccDesiredSpeed);
   accAcceleration = _acc(veh, egoSpeed, predSpeed, gap2pred,
      vars->accHeadwayTime);
   if (gap2pred > 250 || ccAcceleration < accAcceleration) {
      controllerAcceleration = ccAcceleration;
   }
   else {
      controllerAcceleration = accAcceleration;
   }
   break;
case Plexe::CACC:
   if (invoker == MSCFModel_CC::FOLLOW_SPEED) {
      predAcceleration = vars->frontAcceleration;
      leaderAcceleration = vars->leaderAcceleration;
      leaderSpeed = vars->leaderSpeed;
   }
   else {
      /* if the method has not been invoked from
         followSpeed() then it has been
       * invoked from stopSpeed(). In such case we set all
          parameters of preceding
       * vehicles as they were non-moving obstacles
       */

      predAcceleration = 0;
      leaderAcceleration = 0;
```

```
        leaderSpeed = 0;
    }
    //TODO: again modify probably range/range-rate controller
        is needed
    ccAcceleration = _cc(veh, egoSpeed, vars->ccDesiredSpeed);
    caccAcceleration = _cacc(veh, egoSpeed, predSpeed,
        predAcceleration, gap2pred, leaderSpeed,
        leaderAcceleration, vars->caccSpacing);
// Add these codes to launch message Falsification attack
    if(MSNet::getInstance()->getCurrentTimeStep() >=
        string2time("50.00") && veh->getID() == "platoon.2")
        caccAcceleration = _cacc(veh, egoSpeed, predSpeed/2,
            predAcceleration, gap2pred, leaderSpeed,
            leaderAcceleration, vars->caccSpacing);
    // End of codes
    // Add these codes to launch collision induction attack
    if(MSNet::getInstance()->getCurrentTimeStep() >=
        string2time("50.00") && veh->getID() == "platoon.2")
        caccAcceleration = _cacc(veh, egoSpeed, predSpeed*2,
            predAcceleration, gap2pred, leaderSpeed,
            leaderAcceleration, vars->caccSpacing);
    // End of codes
    //if CACC is enabled and we are closer than 20 meters, let
        it decide
    if (gap2pred < 20) {
        controllerAcceleration = caccAcceleration;
    }
```

```
        else {
            controllerAcceleration = fmin(ccAcceleration,
                caccAcceleration);
        }
        // Add these codes to defense against security attacks,if
            there is abnormal situation, we switch CACC to ACC
        accAcceleration = _acc(veh, egoSpeed, predSpeed, gap2pred,
            vars->accHeadwayTime);
        if(fabs(caccAcceleration-accAcceleration) <=
            ACCELERATION_THRESHOLD)
            controllerAcceleration = caccAcceleration;
        else
            controllerAcceleration = accAcceleration;
        // End of codes
        break;
    case Plexe::FAKED_CACC:

        if (invoker == MSCFModel_CC::FOLLOW_SPEED) {
            //compute ACC acceleration that will be then used to
                check for vehicles in front
            vars->followAccAcceleration = _acc(veh, egoSpeed,
                predSpeed, gap2pred, vars->accHeadwayTime);
        }
        else {
            //compute ACC acceleration that will be then used to
                check for vehicles in front
```

```
            vars->freeAccAcceleration = _acc(veh, egoSpeed,
                predSpeed, gap2pred, vars->accHeadwayTime);
        }
        ccAcceleration = _cc(veh, egoSpeed, vars->ccDesiredSpeed);
        caccAcceleration = _cacc(veh, egoSpeed,
            vars->fakeData.frontSpeed,
            vars->fakeData.frontAcceleration,
            vars->fakeData.frontDistance,
            vars->fakeData.leaderSpeed,
            vars->fakeData.leaderAcceleration, vars->caccSpacing);
        controllerAcceleration = fmin(ccAcceleration,
            caccAcceleration);
        break;
    case Plexe::DRIVER:
        std::cerr << "Switching to normal driver behavior still
            not implemented in MSCFModel_CC\n";
        assert(false);
        break;
    default:
        std::cerr << "Invalid controller selected in
            MSCFModel_CC\n";
        assert(false);
        break;
  }
}
//compute the actual acceleration applied by the engine
```

```
    //engineAcceleration = \_actuator(veh, controllerAcceleration,
       vars->egoAcceleration);
    // here, we want to apply the desired acceleration immediately
    engineAcceleration = controllerAcceleration;
    //compute the speed from the actual acceleration
    speed = MAX2(SUMOReal(0), egoSpeed +
       ACCEL2SPEED(engineAcceleration));
    //if we have to ignore modifications (e.g., when this method is
       invoked by the lane changing logic)
    //DO NOT change the state of the vehicle
    if (!vars->ignoreModifications) {
       if (invoker == MSCFModel_CC::FOLLOW_SPEED &&
          vars->followSpeedSetTime !=
          MSNet::getInstance()->getCurrentTimeStep()) {
          vars->controllerFollowSpeed = speed;
          vars->followSpeedSetTime =
             MSNet::getInstance()->getCurrentTimeStep();
          vars->followControllerAcceleration = controllerAcceleration;
       }
       if (invoker == MSCFModel_CC::FREE_SPEED) {
          vars->controllerFreeSpeed = speed;
          vars->freeControllerAcceleration = controllerAcceleration;
       }
    }
    return speed;
}
```

Listing 9.2: SimplePlatooningApp Source Code

```
#include "SimplePlatooningApp.h"
#include "crng.h"
#include "WaveShortMessage_m.h"
#include "MacPkt_m.h"
#include "Mac1609_4.h"
#include <BaseProtocol.h>
Define_Module(SimplePlatooningApp);
void SimplePlatooningApp::initialize(int stage) {
  BaseApp::initialize(stage);
  if (stage == 1) {
    //get the oscillation frequency of the leader as parameter
    leaderOscillationFrequency =
       par("leaderOscillationFrequency").doubleValue();
    //should the follower use ACC or CACC?
    const char *strController = par("controller").stringValue();
    //for now we have only two possibilities
    if (strcmp(strController, "ACC") == 0) {
      controller = Plexe::ACC;
    }
    else {
      controller = Plexe::CACC;
    }
    //headway time for ACC
    accHeadway = par("accHeadway").doubleValue();
    //leader speed
    leaderSpeed = par("leaderSpeed").doubleValue();
```

```
if (myId == 0) {
  //ACC speed is 100 km/h
  traci->commandSetCruiseControlDesiredSpeed(traci->getExternalId(),
     leaderSpeed / 3.6);
  //leader uses the ACC
  traci->commandSetActiveController(traci->getExternalId(),
     Plexe::ACC);
  //leader speed must oscillate
  changeSpeed = new cMessage();
  scheduleAt(simTime() + SimTime(0.1), changeSpeed);
}
else {
  //followers speed is higher
  traci->commandSetCruiseControlDesiredSpeed(traci->getExternalId(),
     (leaderSpeed + 30) / 3.6);
  //followers use controller specified by the user
  traci->commandSetActiveController(traci->getExternalId(),
     controller);
  //use headway time specified by the user (if ACC is employed)
  traci->commandSetACCHeadwayTime(traci->getExternalId(),
     accHeadway);

  changeSpeed = 0;
}
//every car must run on first lane
traci->commandSetFixedLane(traci->getExternalId(), 0);
```

```
    // Add these codes to set cacc constant spacing
    traci->commandSetCACCConstantSpacing(traci->getExternalId(), 5);
    // End of codes
  }
}
void SimplePlatooningApp::finish() {
  BaseApp::finish();
  if (changeSpeed) {
    cancelAndDelete(changeSpeed);
    changeSpeed = 0;
  }
}
void SimplePlatooningApp::onData(WaveShortMessage *wsm) {
}
void SimplePlatooningApp::handleSelfMsg(cMessage *msg) {
  //this takes car of feeding data into CACC and reschedule the self
     message
  BaseApp::handleSelfMsg(msg);
  if (msg == changeSpeed && myId == 0) {
    //make leader speed oscillate
    traci->commandSetCruiseControlDesiredSpeed(traci->getExternalId(),
       (leaderSpeed + 10 * sin(2 * M_PI * simTime().dbl() *
       leaderOscillationFrequency)) / 3.6);
    scheduleAt(simTime() + SimTime(0.1), changeSpeed);
  }
}
```

CHAPTER 10

Limitations and Future Work

Our analysis, simulation, and evaluation are performed over PLEXE. The controllers (CACC, ACC, and CC) used in PLEXE are classical and representative. Although we believe our approach is general enough to extend to other controllers, still we think it is necessary to evaluate the proposed solution over other realizations of platoon systems using different controllers. We are currently communicating with the Crash Avoidance Metrics Partnership (CAMP) for potential collaboration on its platoon implementation.

In our study, we assume a homogeneous platoon system. In reality, apparently, a platoon consists of heterogeneous vehicle systems. Experiences, lessons and recommendations gained from this study may not apply to a heterogeneous platoon system. Heterogeneous platooning has been studied by the transportation research community. We are going to extend our work by considering heterogeneous vehicle platoon systems.

Our study is based on theoretical analysis and simulation. In reality, the situation can be more complicated. We believe our work provides some insights on the safety and security of platooning and hope it can open a new area of research towards safer and more secure platoon mechanisms.

CHAPTER 11

Conclusions

In this paper, we present our work towards a safe and secure platoon co-design. We have shown that cyber attack on a platoon system can have the most severe and widespread safety impact as defined by the EVITA vehicle security risk model. We argue the importance of safety-security co-design for safety critical cyber physical systems and make the first effort toward a safety-security co-design engineering process which allows functional security requirements to be derived for a safe automated vehicle platoon system. Based on the co-design analysis, we present a general approach for designing a safe and secure platooning. Following the general approach, we propose a new platoon control algorithm that takes into account both safety and security. The effectiveness of the proposed scheme in achieving the safety goal as well as defending against security attacks has been analyzed, demonstrated, and evaluated through both theoretical analysis and vehicle network simulation.

Publications

- Secure and Safe Automated Vehicle Platooning. Jiafa Liu, Di Ma, Haojin Zhu, et al. IEEE Reliability Magazine 2016.
- A Functional Co-Design towards Safe and Secure Vehicle Platooning. Jiafa Liu, Di Ma, Haojin Zhu, et al. AsiaCCS Workshop 2017.
- Who Moved My Cheese: Towards Automatic and Fine-grained Classification and Modeling Ad Network. Jiafa Liu, Haojin Zhu, Di Ma, et al. IEEE GLOBECOM 2016.

Bibliography

[1] The sartre project. http://www.sartre-project.eu/en/Sidor/default.aspx.

[2] Researcher hacks self-driving car sensors. http://spectrum.ieee.org/cars-that-think/transportation/self-driving/researcher-hacks-selfdriving-car-sensors.

[3] Elyes Ben Hamida, Hassan Noura, and Wassim Znaidi. Security of cooperative intelligent transport systems: Standards, threats analysis and cryptographic countermeasures. *Electronics 4.3*, pages 380–423, 2015.

[4] Bruce DeBruhl. Is your commute driving you crazy?: a study of misbehavior in vehicular platoons. In *Proceedings of the 8th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2015.

[5] Dadras Soodeh, Ryan M. Gerdes, and Rajnikant Sharma. Vehicular platooning in an adversarial environment. In *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, 2015.

[6] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, and Carl Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Trans. on Dependent and Secure Computing*, 1:11–33, 2004.

[7] Simon Burton, Juergen Likkei, Priyamvadha Vembar, and Marko Wolf. Automotive functional safety = safety + security. In *First International Conference on Security of Internet of Things (SecureIT 2012)*, 2012.

[8] B. Czemy. System security and system safety engineering: different similarities and a system security engineering process based on the ISO 26262 process framework. *SAE Int. J. Passeng. Cars - Electron. Electr. Syst.*, 6:349–359, 2013.

[9] K Koscher, A Czeskis, F Roesner, S Patel, T Kohno, S Checkoway, and S Savage. Experimental security analysis of a modern automobile. In *In Security and Privacy (SP), 2010 IEEE Symposium on (pp. 447-462). IEEE*, 2010, May.

[10] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, and T. Kohno. Comprehensive experimental analyses of automotive attack surfaces. In *USENIX Security Symposium*, 2011, August.

[11] Ishtiaq Roufa, R. M., H. Mustafaa, Travis Taylora, S. O., W. Xua, M. Gruteserb, W. Trappeb, and I. Seskarb. Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study. In *In 19th USENIX Security Symposium, Washington DC (pp. 11-13)*, 2010, February.

[12] C. Miller and C Valasek. Remote exploitation of an unaltered passenger vehicle. In *Black Hat USA*, 2015.

[13] Chenxi Zhang, Rongxing Lu, Xiaodong Lin, P-H Ho, and Xuemin Shen. An efficient identity-based batch verification scheme for vehicular sensor networks. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*. IEEE, 2008.

[14] Xiaodong Lin, Rongxing Lu, Xiaohui Liang, and Xuemin Shen. Stap: A social-tier-assisted packet forwarding protocol for achieving receiver-location privacy preservation in vanets. In *INFOCOM, 2011 Proceedings IEEE*, pages 2147–2155. IEEE, 2011.

[15] Pooja Kavathekar. Cognitive vehicle platooning in the era of automated electric transportation. In *UTAH STATE UNIVERSITY*, 2012.

[16] S. Gehrig and F. Stein. Collision avoidance for vehicle-following systems. *IEEE Transactions on Intelligent Transportation Systems*, pages 233–244, June 2007.

[17] H. Araki, K. Yamada, Y. Hiroshima, and T. Ito. Development of rear-end collision avoidance system. *Intelligent Vehicles Symposium*, pages 224–229, Sept 1996.

[18] A. Ferrara and C. Vecchio. Second order sliding mode control of vehicles with distributed collision avoidance capabilities. *Mechatronics 19*, pages 471–477, 2009.

[19] Michele Segata. Plexe: a platooning extension for veins. In *Vehicular Networking Conference (VNC), IEEE*, 2014.

[20] M. Segata, F. Dressler, R. Lo Cigno, and M. Gerla. A simulation tool for automated platooning in mixed highway scenarios. *ACM SIGMOBILE Mobile Computing and Communications Review*, pages 46–49, Oct 2012.

[21] Rajesh Rajamani. Vehicle dynamics and control. springer science and business media. 2011.

[22] SAE International. Cybersecurity guidebook for cyber-physical vehicle systems. http://standards.sae.org/wip/j3061/.

[23] Huaxin Li, Haojin Zhu, Suguo Du, Xiaohui Liang, and Xuemin Shen. Privacy leakage of location sharing in mobile social networks: Attacks and defense. *IEEE Transactions on Dependable and Secure Computing*, 2016.

[24] Huaxin Li, Zheyu Xu, Haojin Zhu, Di Ma, Shuai Li, and Kai Xing. Demographics inference through wi-fi network traffic analysis. In *Computer Communications, IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on*, pages 1–9. IEEE, 2016.

[25] Mani Amoozadeh, A. Raghuramu, C. Chuah, D. Ghosal, H. Zhang, J. Rowe, and K. Levitt. Security vulnerabilities of connected vehicle streams and their impact on cooperative driving. *Communications Magazine, IEEE*, pages 126–132, 2015.

[26] Evita: E-safety vehicle intrusion protected applications. http://www.evita-project.org/.

[27] Xin Zhao, Dongmei Wu, Yichun Yeh, and Harutoshi Ogai. Development of optimal control system for safe distance of platooning using model predictive control. In *Asia-Pacific Conference on Simulated Evolution and Learning*, pages 65–74. Springer, 2010.

[28] 10 fail-safe examples. http://simplicable.com/new/fail-safe.