2023
Volume 7, Number 4

ACM Transactions on

Cyber-Physical Systems

continued on back cover

ACM Transactions on Cyber-Physical Systems
Vol. 7 • No. 4 • 2023
Articles 22–29

Association for Computing Machinery

Advancing Computing as a Science & Profession

ACM Transactions on Cyber-Physical Systems

ACM
1601 Broadway, 10th Floor
New York, NY 10019-7434
Tel.: (212) 869-7440
Fax: (212) 869-0481
https://www.acm.org

Home Page: https://tcps.acm.org/

The ACM Transactions on Cyber-Physical Systems (ISSN: 2378-962X) is published quarterly in Spring, Summer, Fall, and Winter by the Association for Computing Machinery (ACM), 1601 Broadway, 10th Floor, New York, NY 10019-7434. Printed in the U.S.A.

For manuscript submissions, subscription, and change of address information, see inside backcover.

Association for Computing Machinery

Advancing Computing as a Science & Profession

Formal Specification, Verification and Repair of Contiki's Scheduler

HASSAN MOUSAVI, Department of Electrical and Computer Engineering, Isfahan University of Technology, Iran
ALI EBNENASIR, Department of Computer Science, Michigan Technological University, USA
ELHAM MAHMOUDZADEH, Department of Electrical and Computer Engineering, Isfahan University of Technology, Iran

This article presents an approach for model extraction, formal specification, verification, and repair of the scheduler of Contiki, which is an event-driven lightweight operating system for the Internet of Things (IoT). We first derive a state machine–based abstraction of the scheduler's modes of operation along with the control flow abstractions of the scheduler's most important functions. We then use a set of transformation rules to formally specify the scheduler and all its internal functions in Promela. Additional contributions with respect to the conference version of this article include (1) modeling nested function calls in the Promela model of the scheduler using a novel technique amenable to model checking in SPIN; (2) modeling protothreads in Promela; (3) specifying and formally verifying 12 critical requirements of the scheduler; (4) detecting new design flaws in Contiki's scheduler for the first time (to the best of our knowledge); (5) repairing the model and the source code of Contiki's scheduler towards fixing the flaws detected through verification, as well as regression verification of the entire model of the scheduler; and (6) experimentally analyzing the time and space costs of verification before and after repair. The proposed formal model of Contiki's scheduler along with novel modeling techniques enhance our knowledge regarding the most critical components of Contiki and provide reusable methods for formal specification and verification of other event-driven operating systems used in Cyber Physical Systems (CPSs) and the IoT.

CCS Concepts: • **Software and its engineering → Formal software verification**; **Model checking**; • **Computer systems organization → Embedded software;**

Additional Key Words and Phrases: Specification, verification, Contiki, operating system

ACM Reference format:
Hassan Mousavi, Ali Ebnenasir, and Elham Mahmoudzadeh. 2023. Formal Specification, Verification and Repair of Contiki's Scheduler. *ACM Trans. Cyber-Phys. Syst.* 7, 4, Article 22 (October 2023), 28 pages.
https://doi.org/10.1145/3605948

A preliminary version of this article appeared in the proceedings of CSI/CPSSI International Symposium on Real-Time and Embedded Systems and Technologies (RTEST) [30].

Authors' addresses: H. Mousavi and E. Mahmoudzadeh, Department of Electrical and Computer Engineering, Isfahan University of Technology, University Bulevard, Esteghlal Square, Isfahan, Islamic Republic of Iran, 84156-83111; emails: hassan.mousavi@ut.ac.ir, mahmoudzadeh@iut.ac.ir; A. Ebnenasir, Department of Computer Science, Michigan Technological University, 1400 Townsend Dr., Houghton, Michigan, USA, 49931; email: aebnenas@mtu.edu.

2378-962X/2023/10-ART22 $15.00
https://doi.org/10.1145/3605948

1 INTRODUCTION

This article presents a systematic approach for formal modeling, system and requirements specification, verification, and repair of Contiki's scheduler, which is an important operating system (OS) for low-end devices in the Internet of Things (IoT). The IoT devices are resource-constrained Cyber Physical Systems (CPSs) that have applications in numerous aspects of our lives (e.g., home appliances, infrastructure monitoring, smart buildings), making their dependability a crucial property. The most critical component of any OS includes its process/task scheduler, and we believe that a highly dependable scheduler can provide a Reliable Computing Base (RCB) for the entire OS. However, the highly nested structure of function calls as well as the event-driven nature of Contiki's scheduler — where two types of events can introduce non-determinism, namely, interrupts and process-generated events — makes it a daunting challenge to extract formal models of the scheduler and to verify them. The objective of this work is to generate a formal model of Contiki's scheduler and to verify such a model towards identifying potential defects in the scheduler and fixing them. In addition to increasing the dependability of the scheduler, a well-structured model of the scheduler will have significant research and educational impacts.

Most existing methods for model extraction and verification of embedded software focus on creating a high-level abstraction of kernel components (e.g., scheduler) where internal details of functions/modules are abstracted away. Moreover, few approaches address the formal specification, verification, and *repair* of an event-driven scheduler with a programming model based on protothreads. For example, Penix et al. [34] use predicate abstraction to formally specify the Honeywell DEOS scheduling kernel in Promela (which is the modeling language of the SPIN model checker [22]) towards verifying time partitioning, which ensures that each thread has access to its complete CPU budget during each scheduling period. Aoki et al. [8] specify a highly abstract Promela model of REL OS's scheduler, combining testing techniques and formal verification to verify safety properties. Choi [15] specifies the core service functions of Trampoline[1] [9] in Promela and verifies some safety properties using Software Fault Tree Analysis (SFTA). Marti et al. [29] extract a Promela model of Topsy's round-robin scheduler, where Topsy [19] is an operating system designed for pedagogical purposes. Their Promela model is geared towards the verification of task isolation, which ensures that user threads cannot access kernel memory. Yu et al. [40] use model checking and theorem proving for formal verification of the μC/OS-II kernel [6]. While there are several methods [12, 28, 35] for formal specification and analysis of modules of Contiki, to the best of our knowledge, none of them focuses on Contiki's scheduler. For example, Peyrard et al. [35] present a case study on deductive verification of the encryption-decryption module of Contiki. They verify the absence of runtime errors, such as invalid memory accesses. Mangano et al. [28] verify the memory allocation module of Contiki, called *memb*. Their proof guarantees the absence of out of bound accesses to the block array in the *memb* module. Blanchard et al. [12] use Frama-C/WP to automatically verify the properties of linked lists in Contiki.

What makes the model extraction and formalization of Contiki's scheduler challenging is its highly nested function call depth as well as its event-driven nature, where two types of events can introduce non-determinism, namely, interrupts and process-generated events. Moreover, the synchronous and asynchronous nature of the process-generated events adds another level of complexity to how the scheduler behaves. As such, interrupts and events can result in complicated interleavings in the execution of the scheduler. For example, Figure 1 illustrates a scenario where there are context switches amongst five processes due to interrupts and process-generated (synchronous and asynchronous) events. A model that captures such concurrency-related behaviors of the scheduler as well as the logic of process scheduling is a valuable asset to researchers, educators,

[1]Trampoline is an RTOS developed for automotive embedded systems based on the OSEK/VDX standard.

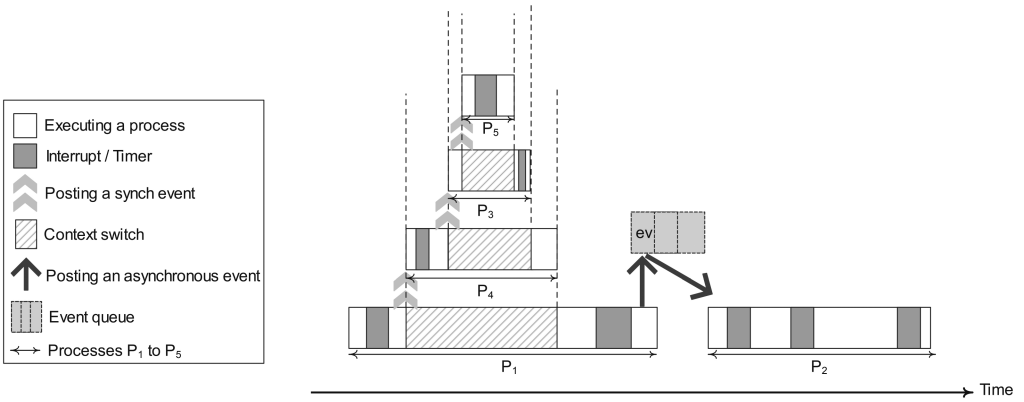


Fig. 1. Complex interleavings in Contiki's scheduler generated by interrupts, synchronous and asynchronous events.

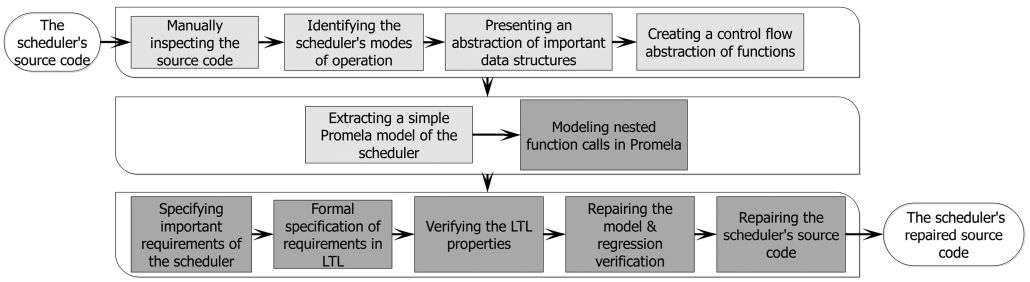


Fig. 2. Different stages of the proposed framework. (Green boxes illustrate new contributions with respect to the conference version [30]).

and developers of Contiki. More importantly, increasing our confidence in the correctness of the scheduler can have a significant impact on any domain of application where Contiki is used.

In order to gain a better understanding of how Contiki's scheduler works and to enhance the dependability of Contiki, this article presents a framework for extracting a hierarchical formal model of the scheduler in Promela, where we can simulate and verify the behaviors of the scheduler (Figure 2). The proposed Promela model is hierarchical in that it captures both the high-level abstraction for the modes of operations of the scheduler as well as the low-level functions of the scheduler. In contrast, most aforementioned methods generate a high-level abstraction of schedulers or kernels of embedded and real-time operating systems in order to enable efficient verification. In the conference version of this article [30], we manually inspect the source code of Contiki's scheduler (written in C) to gain an understanding of its modes of operation. Then, we capture the scheduler's modes of operation as an abstract state machine. The extracted artifact provides a high-level view of how the scheduler executes when interrupts and/or events occur along with the actions it takes in each mode of operation. Subsequently, we present an abstraction of important data structures such as the process and event queues. We also create the control-flow abstractions of some of the important functions of the scheduler. These functions provide an in-depth view of what the scheduler performs in each mode of operation and how it manages processes and events. We then generate a Promela model of the scheduler using some C-to-Promela transformation rules. In [30], we also verified three simple properties. In this work, we make the following additional

contributions. We present a verification-efficient method for modeling nested function calls in the Promela model of Contiki's scheduler. The proposed method enables efficient simulation and verification of the scheduler concurrency behaviors. We also introduce a technique for modeling protothreads, which are stackless threads in Contiki. To verify the generated model in SPIN, we elicit and formally specify 12 important requirements of the scheduler based on our understanding of Contiki's documents, its in-line comments, and the mailing list discussions of its developers. We then use SPIN and verify the scheduler's model with respect to the Linear Temporal Logic (LTL) specifications of the requirements. Upon the failure of the verification attempt for some requirements, we inspect the counterexamples with respect to Contiki's source in order to validate the detected flaws. We then fix the flaws in both the Promela model and the source code of the scheduler, and re-verify the repaired model for all properties again (i.e., regression verification). To analyze the impact of our modeling methods and the agility of the generated Promela model on the cost of verification, we experimentally analyze the time and space costs of verification before and after repair.

A direct benefit of the proposed formal specification and verification of Contiki's scheduler includes the identification of several important and subtle flaws in the design and implementation of the scheduler. For example, a compromised process can overwhelm the scheduler (hence, the kernel) of Contiki by creating new processes that have no threads of execution (i.e., dummy processes) because the scheduler does not kill such processes even if a kill signal is sent to them! (See Requirement #7 in Section 6.3.) To the best of our knowledge, this work is the first method that enables the detection of such corner-case flaws. The discovery of these flaws indicates the importance of the proposed method in terms of generating a hierarchical Promela model for a complex system such as Contiki's scheduler, which involves several challenging tasks such as the modeling of (synchronous and asynchronous) event processing intertwined with interrupt processing and nested function call modeling. More importantly, we have repaired the generated Promela model and have successfully re-verified the revised model for all properties. Accordingly, we have revised the source code of the scheduler towards fixing the detected bugs. These defects are temporal and it would have been very difficult to detect them without using the proposed approach. The C source code, the control-flow abstractions, and the generated Promela model of the scheduler (along with the repaired model and source code) are available in [5].

Organization. Section 2 discusses some basic concepts of Contiki and Promela. Section 3 presents the modes of operation of Contiki's scheduler and the control flow abstraction of its important functions. Section 4 puts forward a formal model of Contiki's scheduler in Promela. Section 5 introduces a method for capturing how the scheduler actually executes processes in the programming model of Contiki. Section 6 discusses the results of verifying the Promela model of the scheduler with respect to some of its critical requirements. Subsequently, Section 7 presents the way we have repaired the scheduler towards fixing the flaws detected during verification. Section 8 provides experimental results. Section 9 focuses on related work. Finally, Section 10 makes concluding remarks and discusses future work.

2 PRELIMINARIES

This section presents the basic components of Contiki's kernel, such as process states, process control blocks, process list and event queue (Section 2.1). Section 2.2 also discusses the basic concepts of the Promela modeling language [1, 22].

2.1 Contiki

This section discusses Contiki [32], its scheduling algorithm, process list, and event queue. Contiki is an open-source operating system (written in the C programming language), which has

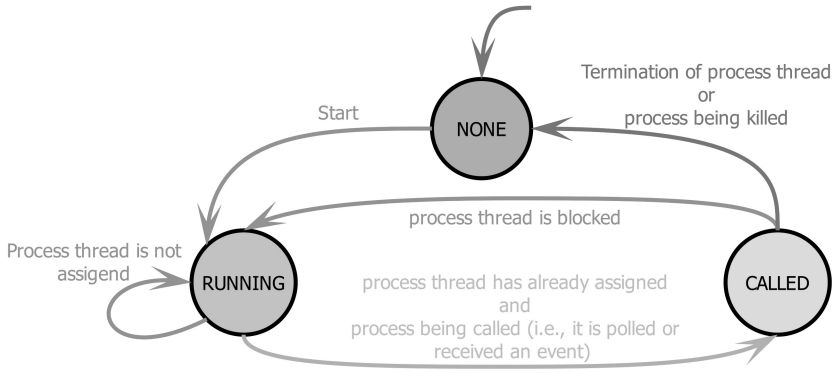


Fig. 3. State diagram of a process.

high portability and runs on different platforms. The required memory for running Contiki is in kilobytes. Thus, it is possible to use Contiki in memory-constrained devices. Contiki has an event-driven kernel, which makes it suitable for use in embedded systems, such as low-power IoT devices. Contiki's kernel comprises two important data structures: a *process list*, which includes some active processes, and an *event queue* that involves some unprocessed events. Each process includes a *process thread* as the code of the process and a Process Control Block (PCB). The process thread is a C function that could be blocked on an event and is terminated when all tasks are done. Contiki uses the protothread mechanism [16, 17] to support a lightweight per-thread context switching. A *protothread* is a stackless thread. All protothreads run on the same stack. The PCB includes some information about each process (such as process state, name, pointer to the thread, etc.), which is internally used by the kernel. A process can transit between three states, namely, NONE, RUNNING, and CALLED (Figure 3).

A process is in the NONE state if and only if it is not in the process list, which includes active processes. When a process is created, the kernel initializes its PCB. Upon starting a process, the kernel adds it to the process list and the process state transits from NONE to RUNNING. A process stays in the RUNNING state as long as it is not scheduled for execution. A process may be scheduled for execution due to an interrupt or an event from another process. When an interrupt occurs, a flag is set to indicate that the process needs to be *polled*. That is, its process thread should be executed. A process transits to the CALLED state if it is polled or called by an event. A CALLED process remains in this state until it is blocked due to waiting for some event/resource. In this case, it transits to RUNNING again. A process moves to NONE from CALLED upon termination or due to receiving an exit/kill signal. Next, we introduce Contiki's scheduler (Section 2.1.1), the process list (Section 2.1.2) and the event queue structures (Section 2.1.3).

2.1.1 Contiki's Scheduler. Polling in Contiki is a particular type of event with high priority. The priority of a poll event precedes other events in the event queue, which involves some asynchronous events. Also, polling is a way to make a process run in an Interrupt Service Routine (ISR) as soon as an interrupt occurs. Thus, an essential task in Contiki's kernel is to schedule the process execution based on the occurrence of events in the event queue. In fact, the scheduler continuously performs two tasks. First, it dequeues an event from the event queue and sends it to the receiving process(es). Second, it polls the processes that need to be polled.

2.1.2 The Structure of the Process List. Figure 4 illustrates the structure of the process list. The `state` field in the PCB of each process illustrates the state of the process. Moreover, the `name` and

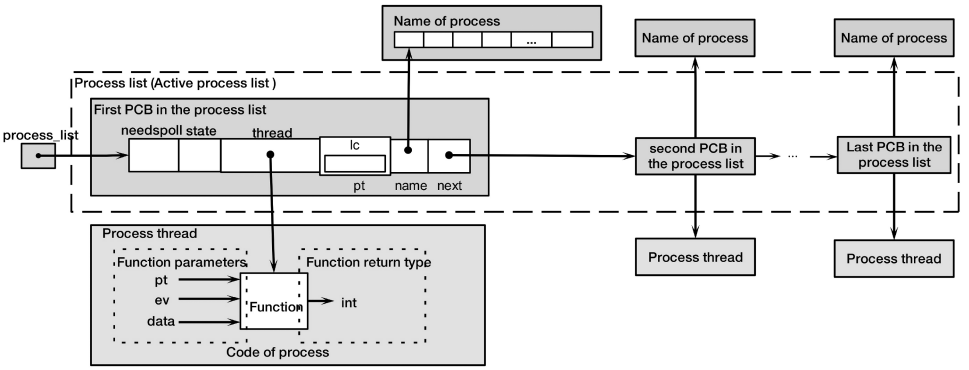


Fig. 4. Structure of the process list.

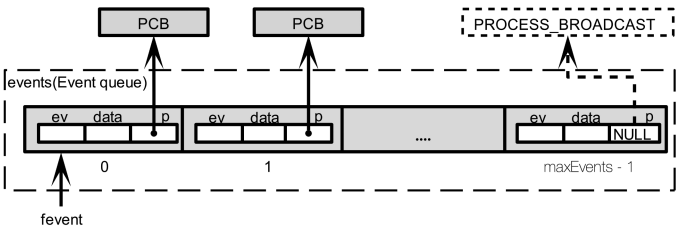


Fig. 5. Structure of the event queue.

needspoll fields respectively capture the name of a process and whether it needs to be polled. The thread field contains a pointer to the process thread. A process thread has three parameters — ev, data, and pt — where ev and data are used to send an event and data to that process, respectively. pt is a pointer to the protothread control structure, which uses an lc field as a *local continuation*. A local continuation is similar to a temporary variable that holds the value of the program counter of a protothread when a context switch occurs. Such a lightweight mechanism for context switching avoids using a stack. The return value of the process thread is an integer that specifies whether the process thread is terminated or blocked (see Figure 4). When the process thread is blocked, it means that the process is waiting for an event and when that event occurs, the process execution resumes. The process list may contain several PCBs.

2.1.3 The Structure of the Event Queue. The event queue in Contiki determines the order of process execution (as shown in Figure 5), where the fevent points to the first event in the queue, and p points to the PCB of the receiver of the event. If p is NULL (PROCESS_BROADCAST), the receiver of the event is all processes in the process list. Moreover, data contains the information that should be sent to the receiver. The ev field captures the event type, namely, synchronous (sync) or asynchronous. A synchronous event is immediately sent to the receiver, whereas an asynchronous event is pushed into the event queue and will be processed later.

2.2 Promela

Promela [1] is the modeling language of the SPIN model checker [22] that is mainly used to model concurrent applications and distributed systems. A Promela model includes processes, variables, and channels. The behavior of each process should be declared within a *proctype*. The instantiation of a proctype is done by running it or activating it at the time of declaration [22]. Processes in

Promela can be run concurrently. Variables and communication channels can be defined globally or locally. Channels are used for message passing and inter-process communication. There are two kinds of channels: synchronous and asynchronous. Upon sending a message to a synchronous channel, the sender is blocked until the receiver gets the message. That is, the sender and receiver perform a rendezvous at the channel. However, an asynchronous channel behaves like a FIFO. Also, sending a message to an asynchronous channel is non-blocking. Statements in Promela are guarded commands (a.k.a. *actions*) in the form of $grd \rightarrow stmt$, where grd is a Boolean expression in terms of the model variables. When the grd holds (i.e., the action is *enabled*), the statement $stmt$ is executed. An action can be executed atomically by putting it in an atomic block. If a proctype has multiple enabled actions, one is selected for execution non-deterministically.

3 CONTROL FLOW ABSTRACTION OF THE SCHEDULER

This section presents a high-level abstraction of the control flow of the scheduler in the form of its modes of operation (Section 3.1). We also provide the control flow abstraction of some functions used by the scheduler (Section 3.2).

3.1 Scheduler's Modes of Operation

The scheduler executes processes in the process list based on the event queue and whether they need to be polled. The main task of the scheduler includes two steps, namely, polling and event processing, that execute in an infinite loop. First, the scheduler checks processes that need to be polled. Then, it removes an event from the event queue, and sends the event to its receiver. The receiver can be a specific process or all processes in the process list. If the receiver is a process, that process executes. If the receiver is PROCESS_BROADCAST, then the event must be broadcast to all processes. In this case, before executing each process in the process list, processes are checked to see whether they should be polled. Each process executes until it is terminated or blocked, i.e., *non-preemptive scheduling*.

Figure 6 illustrates the control flow abstraction of (nested) function calls in Contiki's scheduler. There are seven functions —main, process_run, do_poll, do_event, call_process, process thread (i.e., protothread), and exit_process — that are executed when they are invoked in a nested way depending on the mode of operation of the scheduler. During system boot-up, the program control transfers into main() (S_0 in Figure 6), which performs some initialization and enters a non-terminating loop. The scheduler starts by invoking process_run() (S_1 in Figure 6) that first checks whether there is any process that needs to be polled, i.e., its needspoll flag is set. If so, the scheduler moves to state S_2 where the function do_poll() is invoked. Then, the scheduler invokes do_event() (S_3 in Figure 6) to process an event. That is, if the event queue is not empty, the scheduler removes an event from the event queue, and invokes call_process() (S_4 or S_8 in Figure 6) to call one or more receiver processes. If the receiver is a specific process, the scheduler invokes the call_process() (S_4 in Figure 6). Otherwise, for each process in the process list, the scheduler invokes do_poll() and call_process() (S_5 and S_8 in Figure 6), respectively. In call_process() (S_4 or S_8 in Figure 6), if the state of the process is RUNNING, the scheduler executes the process thread (S_6 or S_9 in Figure 6). The process thread executes until it is terminated or blocked. In the case of termination, the scheduler invokes exit_process() (S_7 or S_{10} in Figure 6) to remove the process from the process list. When the process thread is blocked, the control is transferred to call_process() (S_4 or S_8 in Figure 6) to change the state of the process to RUNNING. Observe that one can also think of the call graph in Figure 6 as the modes of operation of the scheduler. The aforementioned execution path illustrates the high depth of nested function calls that can challenge the modeling and verification phase of the scheduler. We address this issue in Section 4.

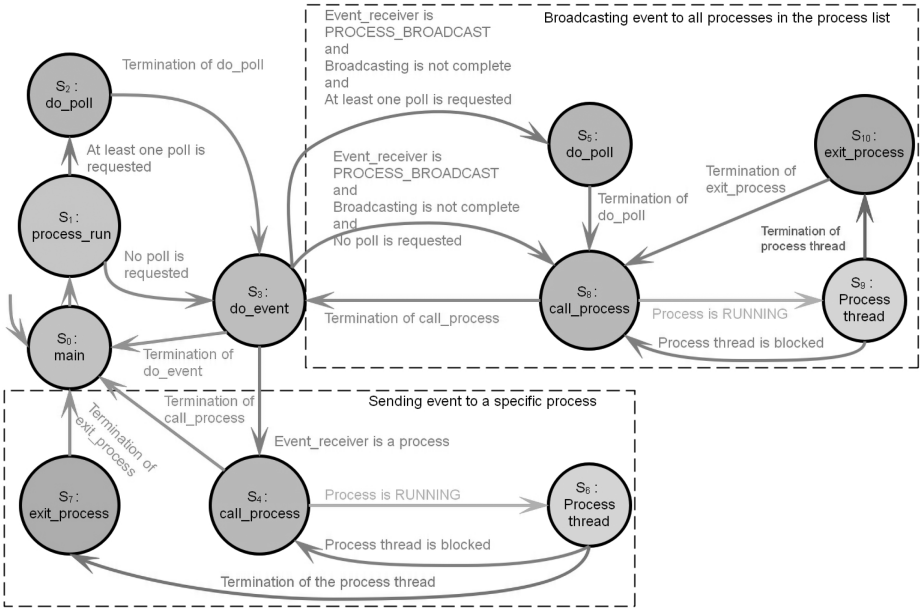


Fig. 6. Abstract call graph of the scheduler's functions.

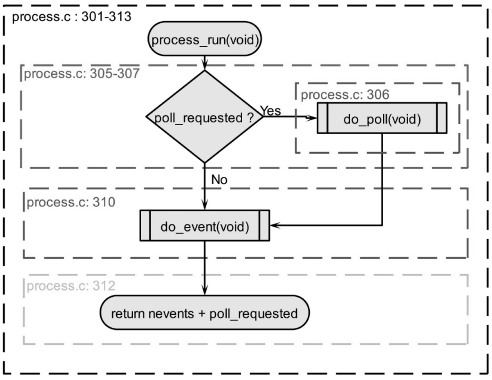


Fig. 7. Control flow abstraction of process_run().

3.2 Control Flow Abstraction of the Scheduler Functions

In this section, we present the details of some functions using their control flow abstraction. For validation purposes, we use the filename.c: Line number format to indicate the corresponding part of Contiki's source code [32] in the file "process.c". Figure 7 illustrates the control flow abstraction of process_run().

poll_requested in Lines 305 and 312 in Figure 7 is a flag that is 1 when an ISR invokes process_poll() to poll a process [32]. nevents in Line 312 in Figure 7 represents the number of events in the event queue. Line 306 in Figure 7 indicates the invocation of do_poll().

Figure 8 illustrates that in do_poll(), the scheduler invokes the call_process function [5] to send an event PROCESS_EVENT_POLL (Line 235 in Figure 8) to each process in the process list (Lines 231–237 in Figure 8) whose needspoll flag is set.

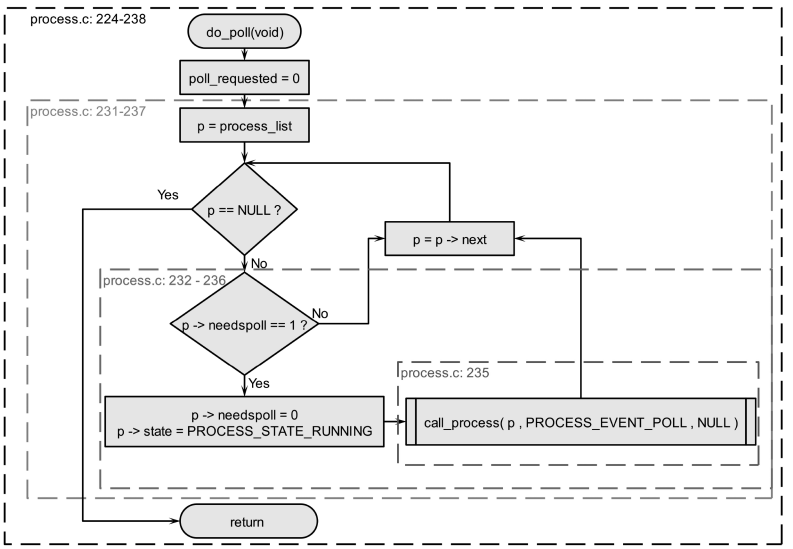


Fig. 8. Control flow abstraction of do_poll().

In the call_process function, if the received event is PROCESS_EVENT_EXIT or the process thread has returned PT_EXITED or PT_ENDED, then the scheduler invokes exit_process() to kill the corresponding process. In exit_process(), the scheduler removes the process p from the process list and notifies other processes (through sending an event) that p is terminated. After checking poll_requested and invoking do_poll() if it needs (in Lines 305–307 in Figure 7), in Line 310 in Figure 7 the scheduler invokes do_event() (Figure 9) in which the kernel sends an event to a process or broadcasts another event. After removing an event from the event queue, the event will be sent to the receiver by invoking call_process() [5] on the receiver(s) (Line 296 in Figure 9). Due to space constraints, we skip the details of call_process() and exit_process(), which are available in [5].

4 PROMELA MODEL OF THE SCHEDULER

In this section, we present a Promela model for Contiki's scheduler. Section 4.1 describes some techniques for modeling nested function calls. Section 4.2 presents the Promela model of data structures used in the scheduler.

4.1 Modeling Nested Function Calls

This section presents how to generate the Promela model of the nested function calls. One of the important challenges in modeling real-world sequential applications in finite-state model checkers includes the issue of nested function calls (for functions that eventually terminate). Since such function calls could potentially be unbounded, a direct modeling thereof would certainly lead to the state space explosion in a model checker. The main reason behind such a state space explosion is twofold. First, the depth of the nested calls is unbounded, whereas most model checkers have an upper bound for the number of concurrent processes (e.g., SPIN can handle at most 255 concurrent processes). Second, model checkers have a specific policy for killing processes that are spawned off dynamically. For example, in the scheduler of SPIN, any terminated process (i.e., a process that reaches the end of its proctype's body) can die (i.e., be inactive) only after its childern die. However, if there is another enabled action in other concurrent processes, then the terminated process may not be killed at a specific moment, even if itself and its children have been terminated.

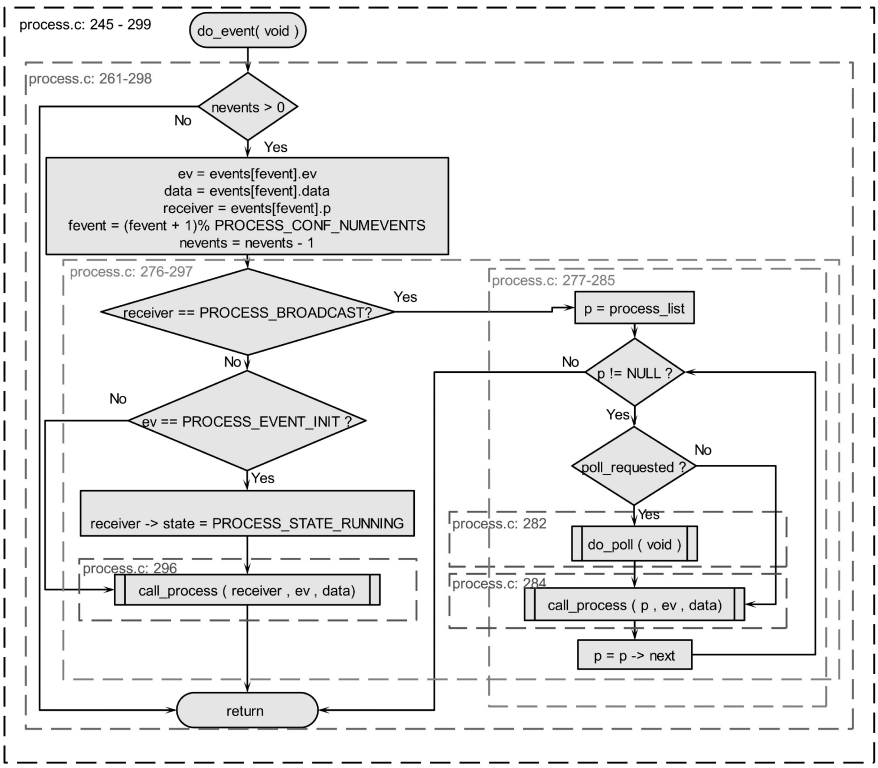


Fig. 9. Control flow abstraction of do_event().

This phenomenon occurs due to the nondeterministic execution of actions, as the scheduler may repeatedly select other enabled actions for execution instead of killing a terminated process. Thus, a terminated process would still be considered as an active process (and counted in towards the upper bound of 255 active processes) as long as it has not been killed. For this reason, most related works [8, 15, 23, 29, 34, 40] ignore the issue of modeling nested function calls. Among a few who actually model nested function calls, Jiang's work [24] is close to ours. In his model, when a function is invoked, the program control is transferred to the callee function, and the caller waits on a synchronous channel until the callee returns control. After the termination of the callee function, its output value as well as the program control are returned to the caller function. Figure 10 illustrates an example of Jiang's function call model [24]. In Figure 10, process_run() invokes do_event() (see Line 13) and is blocked on ret_chan (see Line 14). After do_event() returns its value (see Line 41 in Figure 10), it may be killed at the time when it finishes its execution, and may remain active if there is another concurrent process that has an enabled action.

Our solution is similar to Jiang's to a large extent; however, we introduce a novel trick that forces the scheduler of SPIN to kill any dangling child process whose computation is terminated. Specifically, we apply Jiang's approach, but in the caller we insert a waiting condition right after the return of the callee process (see Line 16 in Figure 11), where we ensure that the number of processes before and after a call remains the same. To this end, we also record the number of active processes in the caller process right before initiating an invocation (see Line 13 in Figure 11). The structure of the callee process (in Figure 10) remains similar to Jiang's approach. The net result is that, in our approach, the caller will continue its execution only when its child process returns and becomes inactive.

```
 1    proctype process_run(chan syn_chan)
 2    {
3-12    //Other statements
 13     run do_event(ret_chan);
 14     ret_chan ? 0;
15-17   //Other statements
 18   }
```

(a) process_run()

```
 1    proctype do_event(chan syn_chan)
 2    {
3-40     //Other statements
 41      syn_chan ! 0
 42   }
```

(b) do_event()

Fig. 10. An example of Jiang's function call model [24].

```
 1    proctype process_run(chan syn_chan)
 2    {
3-12    //Other statements
 13     np = _nr_pr;
 14     run do_event(ret_chan);
 15     ret_chan ? 0;
 16     _nr_pr == np;
 17     //Other statements
 18   }
```

(a) process_run()

```
 1    proctype do_event(chan syn_chan)
 2    {
3-40     //Other statements
 41      syn_chan ! 0
 42   }
```

(b) do_event()

Fig. 11. An example of our proposed function call model.

While the proposed improvement in Jiang's method may seem incremental, its impact on the modeling of nested function calls is significant. To understand this impact, consider the call graph of Figure 12, which represents the complex structure of nested function calls in Contki's scheduler. Each node in the graph (Figure 12) represents a function and an arc captures a function call. Consider an execution path where the functions main(), process_run(), do_event(), do_poll(), call_process(), and exit_process() are invoked in a nested way respectively (follow the solid red line in Figure 12). Note that each function is an instantiated process with a unique id. In Jiang's model [24], exit_process() can remain active after returning the control to the call_process (). Likewise, call_process() cannot die after termination (i.e., returning its value to do_poll), and so forth. Thus, all functions in an execution path can remain active. By contrast, in our proposed model, the instances of the forked processes exit_process(), call_process(), do_poll(), do_event(), and process_run() terminate as soon as they return their values to their corresponding caller function. This way, we avoid increasing the number of active processes. On the other hand, in Jiang's model [24], when the scheduler invokes process_run() for the second time (see the blue node in Figure 12), all the yellow processes are active. During verification, this may cause errors such as "out of memory", "too many channels", and "too many queues" — especially when there are loops and non-terminating loops.

4.2 Promela Model of Data Structures

We present the Promela model of the PCB structure and the process list in Contiki (Section 4.2.1). Then, we generate the Promela model of event structure and the event queue (Section 4.2.2).

4.2.1 PCB and the Process List. The PCB structure is a user-defined structure in Contiki's scheduling algorithm. Figure 13 illustrates the Promela model of the PCB structure (Lines 5–11) and process_list (Line 13) discussed in Figure 4.

We transform the PCB structure (shown in Figure 4) into a Promela typedef (Lines 5–11 in Figure 13). Also, we use an array with constant size (Line 12 in Figure 13) to model all PCBs. These

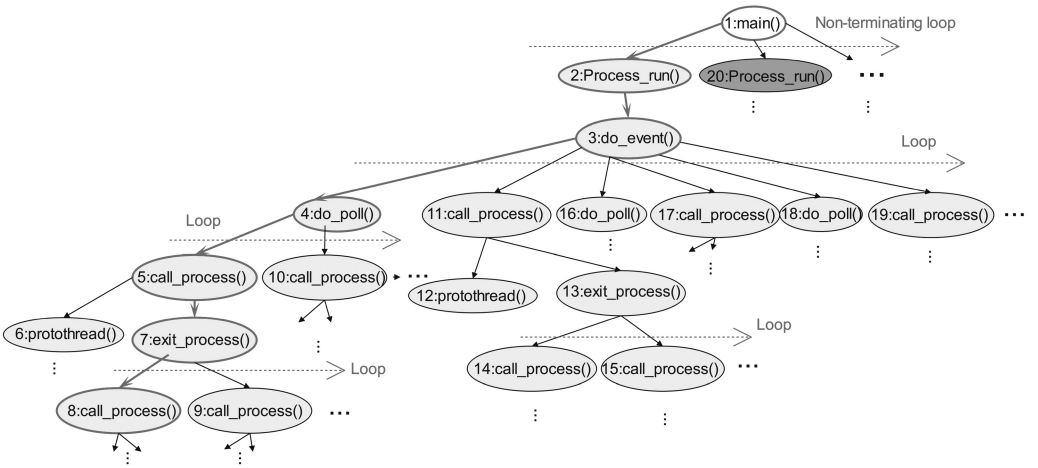


Fig. 12. A complex call tree in Contiki's scheduler.

```
1  #define max_nProcesses 5
2  #define NULL 1000
3  mtype: boolType = {False,True}
4  mtype: proc_state = {PROCESS_STATE_NONE,
                        PROCESS_STATE_CALLED,
                        PROCESS_STATE_RUNNING}
5  typedef process {
6     int next
7     byte name[9];
8     int thread;
9     mtype: boolType needspoll;
10    mtype: proc_state state;
11 }
12 process processes[max_nProcesses];
13 int process_list=NULL    // process.c:54
```

Fig. 13. The PCB and the process list.

processes may or may not be in the process list. We translate a pointer to a process into an integer that refers to an index of the processes array (defined in Line 12 in Figure 13). Thus, the model uses two integers to model process_list (Line 13 in Figure 13) and next (Line 6 in Figure 13), respectively, as the pointer to the first node in the process list and a pointer to the next node in each PCB (Figure 4).

We use an array with a constant size (Line 7 in Figure 13) to capture the process name. Also, the model uses two mtype definitions, boolType and proc_state (Lines 3 and 4 in Figure 13) to model needspoll and the state of PCB (Lines 9 and 10 in Figure 13). Moreover, we transform NULL for each NULL pointer (Line 13 in Figure 13) into 1000 with a C-style macro definition (Line 2 in Figure 13), which means that the pointer is not pointing to any element of an array.

4.2.2 The Event Queue. Figure 14 illustrates a Promela model of the event structure (Lines 1–5 in Figure 14) and event queue (Line 6 in Figure 14) based on Figure 5.

event_data (Line 1 in Figure 14) is a typedef in the model of the event structure. The data types of ev (Line 2 in Figure 14) and data (Line 3 in Figure 14) are, respectively, mtypes process_event_t and process_data_t. Furthermore, we translate p (Line 4 in Figure 14) as a pointer to the receiver to an

```
1 typedef event_data {
2    mtype: process_event_t ev;
3    mtype: process_data_t data;
4    int p;
5 }
6 event_data events[PROCESS_CONF_NUMEVENTS]
```

Fig. 14. Promela model of an event and the event queue structure.

```
1 init()
2 {
3     chan ret_chan = [0] of int;
4     run Processes_initialization(ret_chan);
5     ret_chan ? 0;
6     run main()
7 }
```

Fig. 15. Modeling boot-up using init().

integer, and it indicates the index of the receiver in the processes array (Line 12 in Figure 13). Moreover, the model uses an array of constant size PROCESS_CONF_NUMEVENTS (Line 6 in Figure 14) to model an event queue where PROCESS_CONF_NUMEVENTS is a C-style macro.

5 ENVIRONMENT

This section presents how to model the environment of the scheduler, which is one of the most complicated parts of our work. The environment of the scheduler consists of any component with which the scheduler interacts (e.g., application, interrupts and protothreads). The most significant challenge is due to the non-determinism and high-level concurrency of interrupt occurrence, sync and asynchronous process generated events, protothread mechanism, and process creation. Thus, modeling the environment needs many technical considerations. Section 5.1 discusses how to model the creation of processes in applications when the OS boots up. Section 5.2 presents the Promela model of interrupts and their occurrence. Section 5.3 describes how to model protothreads.

5.1 OS Boot Up and Applications

This section presents how we model process creation in applications when the OS boots up. An application in Contiki creates each process using a macro, namely, PROCESS. This macro initializes the PCB and assigns a process thread to it. The application determines whether or not it should add each process to the auto start list, the list of processes that start automatically. We use init (Figure 15), a predefined process used for initialization, to model booting up in which a Processes_initialization() proctype models the creation of application processes. This proctype instantiates _Process (in Line 7 in Figure 16) to create a max number of processes defined as a macro called max_nProcesses and assigns a process thread to each process non-deterministically. Then, it adds processes to the auto start list (Lines 9–14 in Figure 16). The main() function invokes autostart_start() to start processes in the auto start list (Lines 5–8 in Figure 17).

5.2 Interrupts and the Interrupt Service Routine (ISR)

This section discusses how we model interrupts and their occurrence in Promela. Interrupts may occur non-deterministically during the execution of the scheduler and invoke process_poll() to poll a process. When an interrupt occurs, it preempts the scheduler and the program control is transferred to the associated ISR. After the termination of the ISR, the control returns to the scheduler. Figure 18 illustrates the Promela model of ISR.

```
1  proctype Processes_initialization(chan syn_chan)
2  {
3    chan ret_chan = [0] of int;
4    int p;
5    do
6    :: (p < max_nProcesses) − >
7         run _Process(p,ret_chan);
8         ret_chan ? 0;
9         if
10        :: (nAutoStartProcesses < max_nAutoStartProcesses) − >
11             autostart_processes[nAutoStartProcesses] = p;
12             nAutoStartProcesses = nAutoStartProcesses+1
13        ::else
14        fi
15        p++
16   :: (1) − > nProcesses = p;
17        break
18 od
19   syn_chan ! 0
20 }
```

Fig. 16. Promela model of process initialization.

```
1  active proctype main()
2  {
3     chan ret_chan = [0] of int;
4     int tmp, np;
5     np = _nr_pr;
6     run autostart_start(ret_chan);
7     ret_chan ? 0;
8     np ==_nr_pr;
9     do
10    :: (1) -> np = _nr_pr;
11               run process_run(ret_chan);
12               ret_chan ? tmp;
13               _nr_pr == np
14    od
15 }
```

Fig. 17. Promela model of main().

We use an active proctype to model the non-deterministic behavior of ISR. The built-in variable _last (Line 7 in Figure 18) is a predefined variable in Promela that holds the pid of the last active process. The condition in Line 7 in Figure 18 checks whether the ISR is the process that executed the last operation in the current sequence of operations. If the condition in Line 7 holds, then the ISR randomly selects a process and invokes process_poll() (see Lines 9–12 in Figure 18). Such random selection models the fact that the scheduler has no control over the occurrence of interrupts and the execution of the corresponding ISR. The objective behind the condition in Line 7 of Figure 18 is to prevent non-terminating loops and to ensure that other active proctypes get a chance for execution after each ISR termination.

5.3 Protothreads and Their Execution

This section describes how we model protothreads and how the scheduler activates/deactivates them when a process is scheduled for execution. Programs in Contiki are developed based on

```
1  active proctype ISR()
2  {
3    chan ret_chan = [0] of int;
4    int p;
5    run process_poll();
6    newInterrupt: if
7                  ::(_last != _pid) − >
8                    atomic{
9                      select(p : 0 .. nProcesses - 1);
10                     // poll the process
11                     poll_sync_chan ! p;
12                     poll_sync_chan ? NULL
13                   }
14                 fi
15   goto newInterrupt
16 }
```

Fig. 18. Promela model of interrupts and requesting a process poll.

the protothread model in which lightweight multi-threading is enabled for event-driven applications. Protothreads eliminate the need for heavyweight context switching as only the local continuation of a protothread is saved when it is taken off the CPU. A C function in Contiki applications can contain a protothread whose start and end are marked by the PT_BEGIN() and PT_END() macros. Other statements and macros should be between these two macros. For example, PT_INIT() and PT_WAIT_UNTIL() macros are used, respectively, for initialization and conditional blocking. PT_INIT() resets the local continuation of the protothread. PT_WAIT_UNTIL() stores the local continuation of the current protothread in the common stack, blocks the protothread (until a specified condition holds), and transfers the program control to the scheduler.

The modeling of protothreads comprises two phases: static and dynamic. In the static phase, we model the process thread that is assigned to a process. In this phase, since the thread field of each PCB points to a process thread that is a C function (see Figure 4 in Section 2.1.2), we map protothread to a proctype, called *pThread*. When creating a PCB related to a process, we instantiate pThread and assign its _pid to the PCB using an event identifier ASSIGN_PTHREAD. pThread is initially blocked on a channel called pThread_params_chan (S_0 in Figure 19) until it receives ASSIGN_PTHREAD. The kernel of Contiki uses the pointer pt to the process thread in the PCB structure (see Figure 4 in Section 2.1.2) to save the return point of the process thread. The Promela model (Figure 19) uses a channel to model pt where it is blocked on an event until it receives parameters (transitions S_3–S_4 and S_8–S_9 in Figure 19).

In the dynamic phase, we model the functionality of the protothread mechanism. In this phase, the scheduler invokes each process thread by sending its _pid and an event to pThread_params_chan. In PT_BEGIN, the process thread is blocked until the scheduler sends PROCESS_EVENT_INIT to the process when it starts. In PT_END, the process thread sends PT_ENDED or PT_EXITED via pThread_sync_chan to terminate the process thread. Since a process can be activated again, the process thread will be blocked on PT_BEGIN after termination. The body of the process thread includes Statements (S_5 in Figure 19) and PT_WAIT (S_7 in Figure 19). A process can post a synchronous or asynchronous event to another process (see transitions S_6–S_{11} in Figure 19). A process may also start another process (see transitions S_6–S_{11} in Figure 19). For simplicity, we show postRandomEvent and startRandomProcess (see transitions S_6–S_{11} in Figure 19) as C-style macros. In postRandomEvent the process thread may invoke process_post() or process_post_synch() to post an event to a random process. startRandomProcess invokes process_start() to start a random

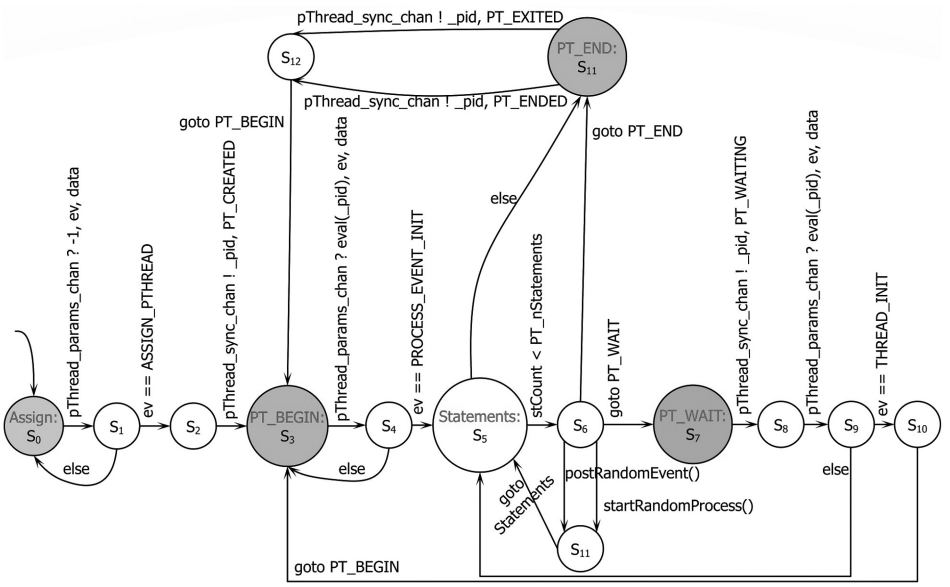


Fig. 19. The state diagram of the protothreads in Promela.

process. This randomness models the non-determinism and dynamism of the scheduler. If the received event is THREAD_INIT (see transitions S_8–S_{10} in Figure 19) then the process thread is blocked on PT_BEGIN until the process starts. Otherwise, the process thread continues its normal execution.

6 VERIFICATION

In this section, we specify some requirements of Contiki's scheduler and use the SPIN model checker [22] to verify the proposed Promela model of the scheduler with respect to these requirements. In this section, we first define the following macros in the extracted Promela model. In these macros, we use temporal operators such as [] (Always), <>(Eventually), X(next), !(Negation), →(Logical implication), &&(Logical and) and ||(Logical or) in the LTL formula.

```
#define isInNoneState(p) (processes[p].state == PROCESS_STATE_NONE)
#define isInRunningState(p) (processes[p].state==PROCESS_STATE_RUNNING)
#define isInCalledState(p) (processes[p].state==PROCESS_STATE_CALLED)
#define isInProcessList(p) (processes_valid[p] == 1)
#define isActive(p) (isInRunningState(p) || isInCalledState(p))
#define needspollProc(p) (processes[p].needspoll==1)
#define isTerminatedProc(p) (isTerminated[p] ==1)
#define ev_exit PROCESS_ EVENT_ EXIT
#define receivedExitEvent(p) (calledProcess_id == p && sent_ev == ev_exit )
#define pollRequested() (poll_requested == 1)
```

processes_valid and isTerminated are Boolean arrays indicating whether a process is in the process list and its process thread is terminated, respectively. callProcess_id and sent_ev are set when a process is called and sent a sync or an asynchronous event. Second, we consider an instance with five processes created randomly to verify some requirements.

Table 1. High-Level Requirements Extracted from Literature on Contiki (e.g., docs, in-line comments) and Classical Requirements

req.#	requirement
1	When a process starts, its state transits to RUNNING.
2	When a process is killed, its state transits to NONE.
3	When a process is called, its state transits to CALLED.
4	Process list contains all active processes (running or called process).
5	When a process is killed, it is removed from the process list.
6	A protothread is called when an event occurs and its corresponding process is running.
7	A process is killed when its protothread terminates or the process receives a kill/exit event (e.g., PROCESS_EVENT_EXIT).

Section 6.1 provides the discovery process of the requirements. Section 6.2 presents the requirements whose verification succeeded. Section 6.3 discusses the requirements whose verification failed and SPIN produced a counterexample. We validate these counterexamples in the source code (i.e., file process.c). To the best of our knowledge, we find these flaws in Contiki's scheduler for the first time.

6.1 Requirements Elicitation

This section presents the discovery process of the requirements. We follow two steps to extract requirements. First, we consider some classic requirements that any OS scheduler must meet and examine the literature to obtain more details about such requirements in Contiki's design. Specifically, we have studied Contiki's documents [2, 4] and the in-line comments in the scheduler's source code in order to extrapolate such classic requirements. For example, in any operating system, each process may be in different states in its life cycle (e.g., created, ready, execution, termination). We study Contiki's documents [2, 4] to find information related to the process life cycle (e.g., process_start(), call_process(), exit_process()). We inspect the source code to discover the process states NONE, RUNNING, and CALLED. For instance, we observe that when a process is killed (i.e., exit_process() is invoked), the process state is changed to NONE. During this process, we also discovered additional requirements. For example, a process is called *active* in Contiki if and only if it is in one of the states RUNNING or CALLED. Moreover, we discovered that there is a relation between the status of a process and its presence in the process list of Contiki's scheduler. Table 1 presents some of the high-level classic requirements that we have extracted based on the approach explained above.

Second, in some cases, we combine some of the high-level requirements to obtain new requirements. This composition decreases the abstraction level of the high-level requirements. Below, we present some of the requirements that are derived in this way:

- We extract Requirement #1 in Section 6.2 as follows. Requirement #3 in Table 1 states that when a protothread is called (e.g., an event occurs), its process state transits to CALLED. After the execution of the protothread, the related process will be killed if the protothread terminates or receives a kill/exit signal (see Requirement #7 in Table 1). When a process is killed, its state changes to NONE (see Requirement #2 in Table 1) and it should no longer be in the process list (see Requirement #4 in Table 1). Thus, we recognize that if a process transits into the NONE state from the CALLED state (it is killed), then that process must eventually be removed from the process list (see Requirement #1 in Section 6.2).

- To extract Requirement #7 in Section 6.3, we look at Requirements #2, #6, and #7 in Table 1 and realize that if the state of a process is RUNNING or CALLED and it receives an exit/kill event, then it will eventually exit and transit to the NONE state (see Requirement #7 in Section 6.3).
- To obtain Requirement #8 in Section 6.3, we consider Requirements #2 and #5 in Table 1 and understand that a process will be in the NONE state and removed from the process list if it is killed. Moreover, when a process starts, it will be in the RUNNING state and will be placed in the process list (see Requirements #1 and #4 in Table 1). Thus, if a process is in the CALLED or RUNNING state, then it must be in the process list (see Requirement #8 in Section 6.3).

6.2 Verified Requirements

This section introduces some of the requirements of Contiki's scheduler that we have mechanically shown to be met (using SPIN) by the proposed Promela model of the scheduler. Requirements #1 and #3 are liveness/progress, whereas #2 and #4 are safety.

Requirement #1. It is always the case that, if a process transits into the NONE state from the CALLED state, then the process will eventually be removed from the process list. This is a logical expectation that when a process is no longer active, it should be removed from the list of processes waiting to be executed.

```
#define p1(p) (isInCalledState(p) && X(isInNoneState(p)) → X(<>(!isInProcessList(p))))
```

We instantiate this macro for five processes and verify the following property:

```
[]( p1(0) && p1(1) && p1(2) && p1(3) && p1(4))
```

Requirement #2. It is always the case that, if a process is removed from the process list, then it must be in the NONE state.

```
#define p2(p) (isInProcessList(p) && X(!isInProcessList(p)) → isInNoneState(p))
```

We then verify the following property:

```
[]( p2(0) && p2(1) && p2(2) && p2(3) && p2(4))
```

Requirement #3. It is always the case that, if a process is added to the process list, then it will eventually transit to the RUNNING state. The rationale behind this requirement is to ensure that there are no orphan processes that occupy the process list.

```
#define p3(p) (!isInProcessList(p) && X(isInProcessList(p)) → X(<>(isInRunningState(p)) ))
```

We then verify the following property:

```
[]( p3(0) && p3(1) && p3(2) && p3(3) && p3(4))
```

Requirement #4. It is always the case that, if a process is polled, then it must be in the CALLED or RUNNING state. A process that is in the NONE state is inactive and cannot respond to polling.

#define p4(p) (!needspollProc(p) && X(needspollProc(p)) → isActive(p))
We then verify the following property:

[](p4(0) && p4(1) && p4(2) && p4(3) && p4(4))

6.3 Detected Flaws

This section discusses the requirements for which we found errors during the verification process. Requirements #6, #7, #10, and #11 are liveness requirements, whereas others are safety.

Requirement #5. It is always the case that, if a process transits into the RUNNING state from the NONE state, then its needspoll field is 0. The rationale behind this requirement is that a process in the NONE state is considered a killed or exited process, and if it is revived, its status must be the same as a newly initialized process.

#define p5(p) ((isInNoneState(p) && (X(isInRunningState(p)))) → !needspollProc(p))
We then verify the following property:

[](p5(0) && p5(1) && p5(2) && p5(3) && p5(4))

Counterexample. Consider the case in which a process is killed (its state transits to NONE and it is removed from the process list) while its needspoll field is 1. If the process receives an asynchronous event (PROCESS_EVENT_INIT) that might have been posted earlier when the process was active, then its state changes to RUNNING from NONE (Lines 276–297 in Figure 9) while its needspoll is 1. Though a process that is in the RUNNING state should be active and exist in the process list, when the kernel posts an asynchronous event (PROCESS_EVENT_INIT) to the process, the state of the process transits to RUNNING without checking the existence of the process in the process list (see Lines 276–297 in Figure 9).

Requirement #6. It is always the case that, if a process is called and its thread is terminated, then the process will eventually transit to the NONE state.

#define p6(p) ((isInCalledState(p) && X(isTerminatedProc(p))) → (<> isInNoneState(p)))
We then verify the following property:

[](p6(0) && p6(1) && p6(2) && p6(3) && p6(4))

Counterexample. Consider a process that moves from the NONE state to the RUNNING state after receiving an asynchronous event (PROCESS_ EVENT_ INIT) (Lines 276–297 in Figure 9). Then, the process is called and executes until it is terminated. However, after the thread terminates, the scheduler invokes exit_process() to kill the process, but since the process is not in the process list, it stays CALLED forever. The root cause of this flaw is that if an event is directed to an individual process, the scheduler fails to check the presence of the process in the process list before calling that process.

Requirement #7. It is always the case that, if the state of a process is RUNNING and that process receives an exit/kill event, then it will eventually exit and transit to the NONE state.

#define p7(p) ((isInRunningState(p) && receivedExitEvent(p)) → (<>(isInNoneState(p))))
We then verify the following property:

[] (p7(0) && p7(1) && p7(2,ev) && p7(3) && p7(4))

Counterexample. We find a scenario in which a process without any thread in the RUNNING state receives a PROCESS_EVENT_EXIT event. However, the process cannot be killed because it is a dummy process, and it stays in the RUNNING state forever.

Requirement #8. It is always the case that, if a process is in CALLED or RUNNING state, then it must be in the process list.

#define p8(p) (isActive(p) → isInProcessList(p))
We then verify the following property:

[](p8(0) && p8(1) && p8(2) && p8(3) && p8(4))

Counterexample. A counterexample confirms that if a process in the NONE state (and is absent in the process list) receives a PROCESS_EVENT_INIT event (which might have been posted by another process), its state transits to RUNNING without being in the process list (Lines 276–297 in Figure 9).

Requirement #9. It is always the case that, when a process is polled, it is in the process list.

#define p9(p) ((isActive(p) && needspollProc(p)) → isInProcessList(p))
We then verify the following property:

[](p9(0) && p9(1) && p9(2) && p9(3) && p9(4))

Counterexample. We find a counterexample in which a process that has been already terminated and is in the NONE state, transits into the RUNNING state by receiving an asynchronous PROCESS_EVENT_INIT event (Lines 276–297 in Figure 9). Then, an interrupt occurrence polls the process while it is not in the process list.

Requirement #10. It is always the case that if the scheduler invokes do_poll() (i.e., poll_requested be reset) and the state of a process is RUNNING or CALLED and its needspoll flag is 1, then the needspoll flag will eventually be reset.

#define p10(p) ((pollRequested() && X(!pollRequested()) && isActive(p) && needspollProc(p)) → <>(!needspollProc(p)))
We then verify the following property:

[](p10(0) && p10(1) && p10(2) && p10(3) && p10(4))

Counterexample. We find a counterexample in which the state of an inactive process changes to RUNNING (by receiving a PROCESS_EVENT_INIT in Lines 276–297 in Figure 9) without being added to the process list. Then, an interrupt requests the scheduler to poll the process. However,

due to not being in the process list, the process will never be polled and its needspoll remains 1 forever.

Requirement #11. It is always the case that, when a process is in the NONE state, then its needspoll field must eventually be 0.

```
#define p11(p) (isInNoneState(p) → <>(!needspollProc(p)))
```
We then verify the following property:

```
[]( p11(0) && p11(1) && p11(2) && p11(3) && p11(4) )
```

Counterexample. We find a counterexample in which the state of a terminated process transits to NONE in exit_process() while its needspoll field remains 1 forever. Although a process that is in the NONE state should be inactive, its needspoll field is not set to 0.

Requirement #12. It is always the case that, if a process is not in the process list, then its needspoll field is 0.

```
#define p12(p) (!isInProcessList(p) → (!needspollProc(p)))
```
We then verify the following property:

```
[]( p12(0) && p12(1) && p12(2) && p12(3) && p12(4) )
```

Counterexample. We find a counterexample in which the kernel removes a terminated process from the process list in exit_process() while its needspoll field remains 1 forever.

In this section, we discussed requirements whose verification results gave counterexamples. We validated these counterexamples based on the source code execution path manually and found the causes of the errors; then, we classified the causes into three groups. In the next section, we present the groups as well as the repair of the model.

7 REPAIR

In Section 6, we classify the counterexamples we observed in our verification activities and discuss some flaws we found in the Promela model of the scheduler with respect to some requirements. We validate these flaws with respect to the source code. Then, we classify the root causes of these errors into three groups.

- As discussed in Section 6.3, some counterexamples (Requirements 6 and 8–10) occur under the following scenario: an asynchronous event (PROCESS_EVENT_INIT) changes the state of an inactive process to RUNNING while the process is not in the process list. While one would expect that a process in the state of RUNNING must exist in the process list (based on Requirement 8), there is no explicit sanity check in the source code of the scheduler for this (see Lines 276–297 in Figure 9). To remedy this flaw, we add a condition to the do_event() function to ensure that any time a PROCESS_EVENT_INIT is sent to a process, the state of the receiver will change to RUNNING only if it is in the process list (see the green area in Figure 20). We use the processes_valid[] array in the Promela model to represent the presence of each process in the process list.
- Another category of counterexamples includes the case in which a process whose thread field is NULL remains in the RUNNING state forever despite receiving a kill/exit event. To fix

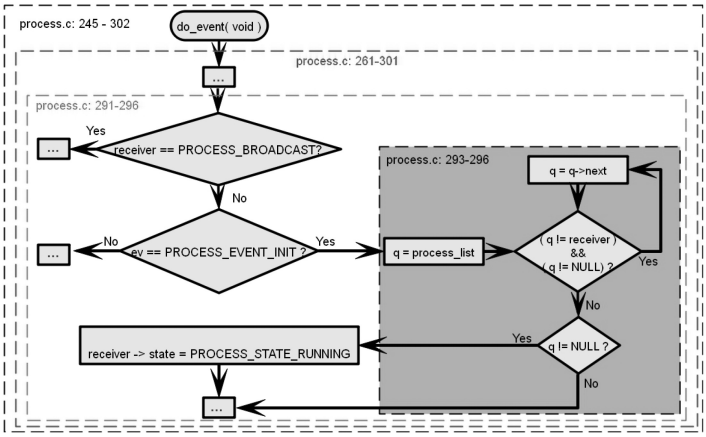


Fig. 20. Repairing do_event().

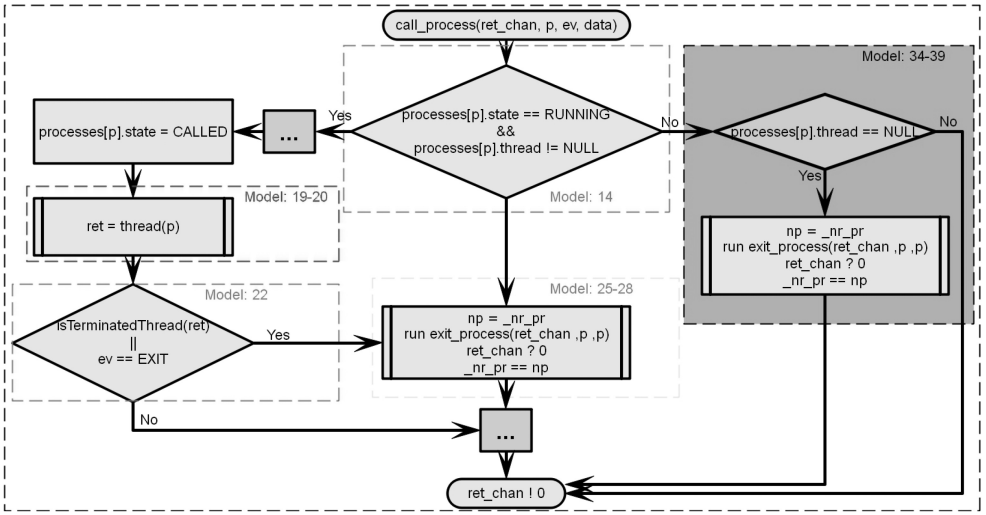


Fig. 21. Repairing call_process().

this issue, we revise the call_process() function in such a way that if a process is not assigned to any thread and it receives an event, then the scheduler must invoke exit_process() to kill it (see the green area in Figure 21).

- As explained in Section 6.3, the needspoll field of a process that is in NONE state must become 0. This property is violated when the scheduler kills a process without setting its needspoll field to 0. To address this issue, we simply include an assignment needspoll = 0 immediately when the scheduler changes the state of a process to NONE in the exist_process() function.

After making the aforementioned revisions, we have successfully re-verified the repaired Promela model of the scheduler with respect to the requirements in Section 6.3. We also have repaired the source code accordingly.

Table 2. Experimental Results of Verification

	Before repair						After repair			
Req.#	**Result**	**Time (Section)**	**States stored**	**States matched**	**Trans.**	**Error depth**	**Time (Section)**	**States stored**	**States matched**	**Trans.**
1 (L)	Ok	12.06	377,027	1,301,416	1,803,354	N/A	11.48	453,209	1,255,730	1,753,712
2 (S)	Ok	9.02	508,680	1,232,823	1,741,503	N/A	9.56	501,211	1,212,691	1,225,663
3 (L)	Ok	13.04	499,190	1,229,532	1,729,622	N/A	11.15	502,372	1,205,192	1,708,493
4 (S)	Ok	8.99	503,708	1,232,636	1,736,344	N/A	9.38	504,384	1,212,920	1,717,304
5 (S)	Error	8.21	273,801	747,589	1,021,390	3,249	9.55	482,986	1,209,739	1,692,725
6 (L)	Error	10.68	239,467	883,780	1,177,904	4,691	11.77	420,519	1,390,455	1,881,035
7 (L)	Error	14.61	399,812	1,033,022	1,433,454	16,426	11.67	485,853	1,216,039	1,701,892
8 (S)	Error	10.93	296,991	804,610	1,101,601	4,956	11.56	484,171	1,211,732	1,695,903
9 (S)	Error	9.08	366,357	969,389	1,335,746	4,956	13.47	484,825	1,214,362	1,669,187
10 (L)	Error	10.17	159,613	478,536	651,320	2,444	13.85	498,036	1,223,595	1,721,631
11 (L)	Error	7.02	9,340	24,030	33,372	1,182	11.31	479,024	1,209,899	1,691,844
12 (S)	Error	6.66	9,056	24,009	33,065	1,092	8.03	484,171	1,211,732	1,695,903

L denotes Liveness and S represents Safety.

8 EXPERIMENTAL RESULTS

This section presents the experimental results of verification. We performed verification on a PC with Intel(R) Core i7-6700HQ CPU (2.6 GHz) and 16 GB memory. We used SPIN version 6.5.0 and its graphical runtime environment iSpin 1.1.1 with bitstate/supertrace storage and partial order reduction search mode. We set advanced parameters maximum search depth to 50,000 and -DVECTORSZ to 2048. We verify liveness requirements without weak fairness. Table 2 illustrates the experimental results of verification before and after the repair of the extracted model. In Table 2, the "States stored" column contains the number of unique global states that were stored during verification; "States matched" captures the number of times the search revisited an already explored state; the "Transitions" column includes the number of explored transitions; and the "Error depth" column presents the depth at which an error was found. Looking at Table 2, we find a few oddities that we should explain. For example, we observe that errors are found at different depths in the "Error depth" column. This is in part due to the different number of processes that are created randomly and dynamically. Table 3 presents the number of processes dynamically created during verification as well as the number of remaining active processes when an error is found. Note that our nested function call mechanism actually kills a callee when its execution ends. Thus, processes may be dynamically created and then killed when we have nested function calls. For this reason, we observe that the failed verification attempt of Requirement #7 reaches maximum depth (among all properties) because a maximum number of dynamic processes are created during its verification (see Table 3). A similar observation can be made about the "States matched". Nonetheless, observe that after repair, "States stored" and "States matched" are roughly close for all requirements because no errors are found and all reachable states are explored. Despite the relatively large number of explored states/transitions, note that the verification times are acceptable. This is mainly due to the proposed method for modeling nested function calls, which makes our Promela model highly scalable for model checking.

9 RELATED WORK

This section discusses existing work on the formal specification and verification of Contiki and real time and embedded software. Several works focus on model checking and theorem proving. Others use static analysis, in which they analyze code without explicitly running it.

Table 3. Number of Processes Created During Failed Verification Attempts

Req.#	Number of Dynamically Created Processes	Number of Active Processes When Error Found
5 (S)	105	13
6 (L)	111	12
7 (L)	175	22
8 (S)	122	13
9 (S)	122	13
10 (L)	100	12
11 (L)	69	12
12 (S)	43	16

L denotes Liveness and S represents Safety.

Model checking of RTOS. Some researchers focus on the model checking of some modules in Real-Time Operating Systems (RTOSs) using extended finite automata in UPPAAL [7, 26]. Boukir et al. [14] apply a formal verification on the Trampoline GLOBAL Earliest Deadline First (G-EDF) scheduling algorithm, which is a multicore version of EDF (Earliest Deadline First). They detect two major errors in the scheduling policy: the first includes a running task that has a priority lower than a ready task and the second is about executing a task on a core other than one indicated by G-EDF. Tigori et al. [36] present a model-based approach for the design of application-specific RTOSs. Their approach configures the code of the RTOS according to the requirements of the application and removes unused services and dead code from the binary program. Li et al. [27] design a portable middleware layer for implementing scheduling algorithms over POSIX-compliant operating systems. They also present a formal model to verify some properties of their middleware. Béchennec et al. [10] present a formal model-based conformance verification of an OSEK/VDX-compliant RTOS. The main purpose of their approach is to optimize the RTOS based on application requirements. Their method impacts the memory footprint of the RTOS and performance by eliminating unused services and dead codes from the binary code. Waszniowski and Hanzálek [39] work on model extraction and verification of non-preemptive interrupt-driven multitasking applications running on an RTOS compliant with the OSEK/VDX standard. They verify logical and timing properties in such applications and their verification results indicate no errors. Tables 4 and 5 summarize methods on model extraction and verification of RTOS components using the SPIN model checker and UPPAAL, respectively. We classify these approaches (in Tables 4 and 5) based on the following criteria: (i) the target component of modeling (e.g., scheduler, the entire kernel, etc.); (ii) support for protothreads; (iii) event-driven nature of the OS; (iv) the granularity of abstraction; (iv) success of verification in finding errors, and (v) model repair.

Although the aforementioned methods study the model extraction and verification of real-time OS kernel modules, as indicated in Tables 4 and 5, none of them addresses model extraction and verification of an event-driven scheduler with a programming model based on protothreads. The major cause of the complexity of our approach stems from the high degree of concurrency due to the nested occurrence of interrupts and process-generated (synchronous and asynchronous) events in a protothread-based programming paradigm. Moreover, the high depth of the nested function calls of Contiki's scheduler makes the model extraction and verification even more

Table 4. Methods on Model Extraction and Verification of Real-Time OS Components Using the SPIN Model Checker

	[34]	[8]	[15]	[29]
Research objective	Verification of time partitioning	Using testing for effective verification	Safety analysis	Verification of task isolation
OS Name	DEOS	REL	Trampoline	Topsy
Module	Time partitioning	Scheduler	Core functions	Kernel
Using protothread	No	No	No	No
Event-driven kernel	No	No	No	No
Granularity	Function level	Module level	Function level	Module level
Verification result	Error detection	No error	Error detection	No error
Model/Code repair	No	No	No	No

Table 5. Methods on Model Extraction and Verification of Real-Time OS Components Using UPPAAL

	[14]	[36]	[27]	[10]	[39]
Research objective	Verification of G-EDF policy	Removing the dead code of RTOS according to the requirments of applications	Formal verification of multitasking RTOS application	Conformance verification of RTOS	Verification of multitasking applications
OS Name	Trampoline	Trampoline	QNX Neutrino	Trampoline	RTOS
Module	Scheduler	Scheduler	Multitasking application	Kernel	Multitasking application
Using protothread	No	No	No	No	No
Event-driven kernel	No	No	No	No	No
Granularity	Function level	Application level	Application level	Function level	Application level
Verification result	Error detection	No error	No error	No error	No error
Model/Code repair	No	No	No	No	No

complicated, whereas most existing methods extract just high-level models of some structures and APIs of the kernel. Also, the granularity of our model is fine-grained where the Promela code is very close to implementation. These features make our verification results without any superfluous counterexample. Moreover, many existing methods (e.g., [10, 27, 36, 39]) concentrate on the interactions of the kernel/scheduler with its environment rather than the internals of the scheduler. Our contributions are not limited to verification only as we also repair the model and the code of Contiki's scheduler.

Theorem proving. Several studies focus on applying theorem-proving techniques to model and design kernel components. Velykis [37] models separate kernel components: a process table, a process queue and a scheduler formally using the Z/Eves theorem prover. Gu et al. [20] and Boukir

et al. [13] develop a certified kernel for secure cloud computing and implement a global EDF in Trampoline. Klein et al. [25] apply a formal proof of the functional correctness of the seL4 microkernel using Isabelle/HOL. Nelson et al. [31] verify the implementation of Hyperkernel with the Z3 SMT solver. Yu et al. [40] use the Coq tool to prove the correctness of the μC/OS-II kernel. Von Tessin [38] formally specifies and verifies a multiprocessor OS kernel in Isabelle/HOL. Bevier et al. [11] use the Boyer-Moore theorem prover to mechanically verify the correctness of a multitasking operating system kernel, called KIT. Elphinstone et al. [18] apply Isabelle in the process of kernel development for high assurance. Heitmeyer et al. [21] formally specify and verify data separation behaviors in a kernel as Top-Level Specification (TLS). Then, they translate the TLS into timed automata and also use the PVS theorem prover [3, 33] to verify data separation. The usability of theorem provers is limited due to their semi-automated nature and their steep learning curve for mainstream engineers. As such, model checking approaches have an advantage in terms of practical use. Thus, it is desirable to develop a model whose size allows model checking of critical properties while having a realistic level of abstraction that permits code realizability. Our Promela model of Contiki's scheduler has these properties.

Static analysis. Three studies focus on the verification of Contiki using Frama-C. Peyrard et al. [35] present a case study on the deductive verification of the encryption-decryption module of Contiki (namely, AES–CCM*). They verify the absence of runtime errors, such as invalid memory accesses. Also, to ensure that the input parameters are compatible with each function's preconditions when called by Contiki's other modules, they have simulated and verified the use of AES–CCM* by generating several test cases. Their approach guarantees that the module is free of out-of-bounds memory accesses. Mangano et al. [28] verify the memory allocation module of Contiki, called *memb*. Their proof guarantees the absence of out-of-bound accesses to the block array in the *memb* module. Blanchard et al. [12] use Frama-C/WP to automatically verify the properties of linked lists in Contiki. It is difficult to apply static analysis methods for modeling and analyzing the dynamic behaviors of an event-driven scheduler with a high degree of concurrency.

10 CONCLUSIONS AND FUTURE WORK

In this article, we presented a novel approach for formal specification, verification, and repair of Contiki's scheduler. First, we manually inspected the source code of Contiki's scheduler and extracted a model (as an abstract state machine) representing how the scheduler changes its modes of operation upon the occurrence of interrupts and process-generated events. Then, we presented abstractions of the data structures as well as control flow abstractions of some critical functions of the scheduler (We accomplished the aforementioned tasks in the conference version [30].) We proposed a proper approach to model nested function calls in the Promela model of Contiki's scheduler. We extracted a Promela model of the scheduler as well as its environment. After specifying some of Contiki's critical requirements, we verified them using the SPIN model checker [22] and detected some important design flaws in the scheduler for the first time. We validated that these flaws are not superfluous errors and that they actually exist in the source code of the scheduler. We then analyzed the counterexamples and determined the root causes of the flaws. Accordingly, we then used the extracted Promela model to remedy these flaws and re-verified the revised Promela model with respect to the scheduler's requirements. Next, we revised the source code of Contiki's scheduler, thereby generating a more dependable version thereof. The most important contribution of our work originates from our fine-grained Promela model of the scheduler that captures its non-deterministic nature yet allows efficient model checking of its critical properties. Our model, along with our verification results, provide valuable assets for researchers, developers of Contiki, and educators.

We plan to extend this work in several directions. First, we would like to incorporate the optional yield() system call (for preemptive scheduling) in the proposed Promela model of the scheduler. Second, we plan to extend this work to capture the improvements in the scheduler of the next-generation Contiki, called Contiki-NG. Such improvements include synchronization primitives such as atomic compare-and-swap, mutexes, and memory barriers. Third, we will model faults and cyber attacks in the proposed Promela model and will analyze the behavior of the scheduler. Such analyses go beyond the traditional failure mode analysis as the contributions of this work enable us to simulate and verify temporal and non-terminating behaviors of the scheduler in the presence of faults and attacks. Finally, we plan to incorporate fault tolerance functionalities in a modular fashion as new components of Contiki's kernel and verify their correctness. The outcome of this extension will be a fault-tolerant scheduler for Contiki, which will tolerate different types of faults (e.g., transient soft errors, process crashes). Such a fault-tolerant scheduler will form a reliable computing base for Contiki.

REFERENCES

[1] 2011. Promela Language Reference. http://spinroot.com/spin/Man/promela.html. (2011).
[2] 2012. Contiki 2.6 docs. https://contiki.sourceforge.net/docs/2.6/. (2012).
[3] 2021. PVS. https://pvs.csl.sri.com. (2021).
[4] 2022. Contiki-NG Docs. https://docs.contiki-ng.org/en/develop/. (2022).
[5] 2022. Dependable Contiki Project. https://github.com/DependableContiki/DependableContiki. (2022).
[6] 2022. uC-OS2. https://github.com/weston-embedded/uC-OS2. (2022).
[7] 2022. UPPAAL. https://uppaal.org/. (2022).
[8] Toshiaki Aoki, Makoto Satoh, Mitsuhiro Tani, Kenro Yatake, and Tomoji Kishi. 2017. Combined model checking and testing create confidence–A case on commercial automotive operating system. In *Cyber-Physical System Design from an Architecture Analysis Viewpoint*. Springer, 109–132.
[9] Jean-Luc Bechennec, Mikael Briday, Sébastien Faucou, and Yvon Trinquet. 2006. Trampoline an open source implementation of the OSEK/VDX RTOS specification. In *2006 IEEE Conference on Emerging Technologies and Factory Automation*. IEEE, 62–69.
[10] Jean-Luc Béchennec, Olivier Henri Roux, and Toussaint Tigori. 2018. Formal model-based conformance verification of an OSEK/VDX compliant RTOS. In *2018 5th International Conference on Control, Decision and Information Technologies (CoDIT)*. IEEE, 628–634.
[11] William R. Bevier et al. 1987. *A Verified Operating System Kernel*. Ph.D. Dissertation. https://www.cs.utexas.edu
[12] Allan Blanchard, Nikolai Kosmatov, and Frédéric Loulergue. 2018. Ghosts for lists: A critical module of Contiki verified in Frama-C. In *NASA Formal Methods Symposium*. Springer, 37–53.
[13] Khaoula Boukir, Jean-Luc Béchennec, and Anne-Marie Déplanche. 2017. Reducing the gap between theory and practice: Towards a proven implementation of global EDF in trampoline. *JRWRTC 2017* (2017), 9.
[14] Khaoula Boukir, Jean-Luc Béchennec, and Anne-Marie Déplanche. 2020. Requirement specification and model-checking of a real-time scheduler implementation. In *Proceedings of the 28th International Conference on Real-Time Networks and Systems*. 89–99.
[15] Yunja Choi. 2014. Model checking Trampoline OS: A case study on safety analysis for automotive software. *Software Testing, Verification and Reliability* 24, 1 (2014), 38–60.
[16] Adam Dunkels, Oliver Schmidt, and Thiemo Voigt. 2005. Using protothreads for sensor node programming. In *Proceedings of the Real-World Wireless Sensor Networks (REALWSN)*, Vol. 5.
[17] Adam Dunkels, Oliver Schmidt, Thiemo Voigt, and Muneeb Ali. 2006. Protothreads: Simplifying event-driven programming of memory-constrained embedded systems. In *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*. 29–42.
[18] Kevin Elphinstone, Gerwin Klein, Philip Derrin, Timothy Roscoe, and Gernot Heiser. 2007. Kernel development for high assurance. In *Proceedings of the 11th Workshop on Hot Topics in Operating Systems*.
[19] George Fankhauser, Christian Conrad, Eckart Zitzler, and Bernhard Plattner. 2000. Topsy–a teachable operating system. *Computer Engineering and Networks Laboratory, ETH Zürich, Switzerland* (2000).
[20] Liang Gu, Alexander Vaynberg, Bryan Ford, Zhong Shao, and David Costanzo. 2011. CertiKOS: A certified kernel for secure cloud computing. In *Proceedings of the 2nd Asia-Pacific Workshop on Systems*. 1–5.
[21] Constance L. Heitmeyer, Myla Archer, Elizabeth I. Leonard, and John McLean. 2006. Formal specification and verification of data separation in a separation kernel for an embedded system. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*. 346–355.

[22] G. Holzmann. 1997. The model checker SPIN. *IEEE Transactions on Software Engineering* 23, 5 (1997), 279–295.
[23] Gerard J. Holzmann and Margaret H. Smith. 2002. An automated verification method for distributed systems software based on model extraction. *IEEE Transactions on Software Engineering* 28, 4 (2002), 364–377.
[24] Ke Jiang. 2009. Model checking C programs by translating C to Promela. (2009).
[25] Gerwin Klein, Kevin Elphinstone, Gernot Heiser, June Andronick, David Cock, Philip Derrin, Dhammika Elkaduwe, Kai Engelhardt, Rafal Kolanski, Michael Norrish, et al. 2009. seL4: Formal verification of an OS kernel. In *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles*. 207–220.
[26] Kim G. Larsen, Paul Pettersson, and Wang Yi. 1997. UPPAAL in a nutshell. *International Journal on Software Tools for Technology Transfer* 1, 1 (1997), 134–152.
[27] Peng Li, Binoy Ravindran, Syed Suhaib, and Shahrooz Feizabadi. 2004. A formally verified application-level framework for real-time scheduling on Posix real-time operating systems. *IEEE Transactions on Software Engineering* 30, 9 (2004), 613–629.
[28] Frédéric Mangano, Simon Duquennoy, and Nikolai Kosmatov. 2016. Formal verification of a memory allocation module of Contiki with Frama-C: A case study. In *International Conference on Risks and Security of Internet and Systems*. Springer, 114–120.
[29] Nicolas Marti, Reynald Affeldt, and Akinori Yonezawa. 2006. Model-checking of a multi-threaded operating system. In *23rd Workshop of the Japan Society for Software Science and Technology*.
[30] Hassan Mousavi, Elham Mahmoudzadeh, and Ali Ebnenasir. 2020. A Promela model for Contiki's scheduler. In *2020 CSI/CPSSI International Symposium on Real-Time and Embedded Systems and Technologies (RTEST)*. IEEE, 1–10.
[31] Luke Nelson, Helgi Sigurbjarnarson, Kaiyuan Zhang, Dylan Johnson, James Bornholt, Emina Torlak, and Xi Wang. 2017. Hyperkernel: Push-button verification of an OS kernel. In *Proceedings of the 26th Symposium on Operating Systems Principles*. 252–269.
[32] George Oikonomou, Simon Duquennoy, Atis Elsts, Joakim Eriksson, Yasuyuki Tanaka, and Nicolas Tsiftes. 2022. The Contiki-NG open source operating system for next generation IoT devices. *SoftwareX* 18 (2022), 101089. https://doi.org/10.1016/j.softx.2022.101089
[33] Sam Owre, John M. Rushby, and Natarajan Shankar. 1992. PVS: A prototype verification system. In *International Conference on Automated Deduction*. Springer, 748–752.
[34] John Penix, Willem Visser, Seungjoon Park, Corina Pasareanu, Eric Engstrom, Aaron Larson, and Nicholas Weininger. 2005. Verifying time partitioning in the DEOS scheduling kernel. *Formal Methods in System Design* 26, 2 (2005), 103–135.
[35] Alexandre Peyrard, Nikolai Kosmatov, Simon Duquennoy, Inria Lille, and Shahid Raza. 2018. Towards Formal Verification of Contiki: Analysis of the AES-CCM* modules with Frama-C. In *Proceedings of the 2018 International Conference on Embedded Wireless Systems and Networks*. 264–269.
[36] Kabland Toussaint Gautier Tigori, Jean-Luc Béchennec, Sébastien Faucou, and Olivier Henri Roux. 2017. Formal model-based synthesis of application-specific static RTOS. (2017).
[37] Andrius Velykis. 2009. Formal modelling of separation kernels. *Master's thesis, Department of Computer Science, University of York* (2009).
[38] Michael Von Tessin. 2012. The clustered multikernel: An approach to formal verification of multiprocessor OS kernels. In *Proceedings of the 2nd Workshop on Systems for Future Multi-core Architectures, Bern, Switzerland*.
[39] Libor Waszniowski and Zdeněk Hanzálek. 2008. Formal verification of multitasking applications based on timed automata model. *Real-Time Systems* 38, 1 (2008), 39–65.
[40] Zhang Yu, Dong Yunwei, Zhang Zhongqiu, Huo Hong, and Zhang Fan. 2011. The study on formal verification of OS kernel. *International Journal of Wireless and Microwave Technologies* 3 (2011), 62–69. DOI: 10.5815/ijwmt.2011.03.10

Received 18 June 2022; revised 3 May 2023; accepted 29 May 2023

A Reliable Wireless Protocol for Highway and Metered-Ramp CAV Collaborative Merging with Constant-Time-Headway Safety Guarantee

XUELI FAN and QIXIN WANG, Department of Computing, The Hong Kong Polytechnic University, China
JIE LIU, Harbin Institute of Technology (Shenzhen), China

To realize the grand vision of automated driving in smart vehicle *cyber-physical systems* (CPS), one important task is to support the merging of *connected automated vehicles* (CAVs) from a metered-ramp to highway. Certain safety rules must be guaranteed. However, this demand is complicated by the inherently unreliable wireless communications. In this article, we focus on the well adopted *constant-time-headway* (CTH) safety rule. We propose a highway and metered-ramp CAV collaborative merging protocol, and formally prove its guarantee of the CTH safety and liveness under *arbitrary wireless data packet losses*. These theoretical claims are further validated by our simulations. Furthermore, the simulation results also show significant improvements in the merging efficiency over other solution alternatives. Particularly, the merging success rates are more than 99% better in 11 out of 18 comparison pairs, and 0%(i.e., tied)~ 71% better in the remaining 7 comparison pairs.

CCS Concepts: • **Computer systems organization** → **Embedded systems**; Robotics; • **Networks** → Network reliability;

Additional Key Words and Phrases: CPS, wireless, reliability, hybrid automata, CAV

ACM Reference format:
Xueli Fan, Qixin Wang, and Jie Liu. 2023. A Reliable Wireless Protocol for Highway and Metered-Ramp CAV Collaborative Merging with Constant-Time-Headway Safety Guarantee. *ACM Trans. Cyber-Phys. Syst.* 7, 4, Article 23 (October 2023), 26 pages.
https://doi.org/10.1145/3609227

The research projects related to this article are supported in part by the Hong Kong RGC Theme-Based Research Scheme (TRS) under Grant T22-505/19-N (P0031331, RBCR; P0031259, RBCP), in part by RGC GRF under Grants PolyU 152002/18E (P0005550, Q67V) and PolyU 152164/14E (P0004750, Q44B), in part by RGC Germany/HK under Grant G-PolyU503/16 (P0001326, RAA8), in part by HKJCCT P0041424 (ZB5A), and in part by the Hong Kong Polytechnic University Fund under Grants P0045578 (CE1C), P0043884 (CD6R), P0043634 (TAB2), P0043647 (TABF), 49NZ, P0042701 (CE09), P0042699 (CE55), P0036469 (CDA8), P0042721 (ZVG0), P0013879 (BBWH), P0031950 (ZE1N), P0033695 (ZVRD), P0000157 (BBWC), P0009706 (YBXW), and 8B2V.
Authors' addresses: X. Fan and Q. Wang (corresponding author), Department of CompuEng, The Hong Kong Polytechnic University, 11 Yuk Choi Rd, Hung Hom, Hong Kong SAR, Kowloon, China; emails: xueli.fan@connect.polyu.hk, csqwang@connect.polyu.hk; J. Liu, Harbin InsEtute of Technology (Shenzhen), University Town of Shenzhen, Shenzhen, Guangdong, 518055, China; email: jieliu@hit.edu.cn.

2378-962X/2023/10-ART23
https://doi.org/10.1145/3609227

1 INTRODUCTION

To realize the grand vision of fully autonomous driving, one of the most promising directions is to first realize it in controlled environments, particularly in highways, where accesses are limited, and ***connected autonomous vehicles*** (**CAVs**) are collaborative [2, 13]. One important structure for such limited access highways is the *metered-ramp* [1, 15, 24, 41, 45], where local CAVs are stopped by a traffic signal (i.e., the *ramp meter*) before they enter the highway system via a ramp. Merging of CAVs from the metered-ramp to highway must be supported. The CAVs involved need to collaborate wirelessly to guarantee certain safety rules, as well as achieve good merging efficiency.

However, these demands are complicated by the inherently unreliable vehicular wireless communications. The solution will heavily depend on the targeted safety rule and the chosen wireless communication paradigm. A panacea solution is highly unlikely. In this article, we shall focus on a widely adopted safety rule, the ***constant time headway*** (**CTH**) safety [8, 11, 14, 30, 37, 39, 47].

Intuitively, CTH safety means at any time instance, any follower vehicle must maintain a constant temporal distance from its predecessor vehicle; i.e., the minimal spatial distance needed is proportional to the follower's current speed. For the wireless communications paradigm, there are two basic categories: ***vehicle to vehicle*** (**V2V**) and ***vehicle to infrastructure*** (**V2I**). Each has its pros and cons. Therefore, a mixed (i.e., V2I+V2V) approach, aka V2X, is gaining increasing attention recently [3, 12, 43, 46]. In this article, we shall also adopt the V2X approach: the design is centered on V2I, but V2V communications between line-of-sight neighboring CAVs along the highway lane are also exploited as an alternative for ranging.

In summary, this article shall focus on a wireless highway and metered-ramp CAV merging protocol, which guarantees the CTH safety under *arbitrary* wireless *data packet* (simplified as "*packet*" in the following) losses and achieves good merging efficiency.

Merging of vehicles is a hot research topic in smart vehicle ***cyber-physical systems*** (**CPS**). Besides the large volume of works based on the pure V2V or pure V2I wireless communications paradigm (which are to be elaborated in Section 2), V2X solutions are gaining increasing attention recently. Wang et al. [42] develop a merging algorithm using V2V and V2I communications to facilitate merging of CAVs. Virtual vehicles are mapped onto both the highway lane and the ramp to facilitate the merging of individual vehicles and platoons. Ntousakis et al. [32] propose a cooperative merging system model based on V2V and V2I communication which enables the effective handling of the available gaps between vehicles and evaluate its performance and impact on highway capacity by adopting a microscopic traffic simulator. Wang et al. [43] present a distributed consensus-based cooperative merging protocol, where ***road side unit*** (**RSU**) based infrastructure assigns sequence identifications to different vehicles based on their estimated arrival time (V2I communication), then vehicles apply distributed consensus protocol to adjust their velocity and positions in advance with V2V communications. Ahmed et al. [3] describe a freeway merge assistance system utilizing both V2V and V2I communication. The freeway merge assistance system uses an innovative three-way handshaking protocol and provides advisories to guide the merging sequence. However, the above works (including those based on pure V2V or pure V2I paradigm) do not discuss how to deal with arbitrary wireless packet losses.

There are also works focusing only on the application layer, and are independent of the underlying communication infrastructure (may it be V2V, V2I, or V2X—in another sense, this can be viewed as a more generic V2X) [6, 9, 22, 29, 31]. However, these works also assume the communication infrastructure is reliable, hence do not deal with arbitrary wireless packet losses.

Aoki et al. [5] present a safe highway and ramp merging protocol, which provides safety by using V2V communications and perception systems cooperatively and accommodates losses of wireless packets. In the protocol, packet losses can decrease traffic throughput, but cannot cause

vehicle collisions. However, how to adapt this protocol to guarantee the CTH safety rule remains an open problem, as the protocol is not designed for the CTH safety rule to begin with.

In order to guarantee CTH safety rule under arbitrary wireless packet losses, we shall deploy a timeout (aka "lease" [18]) based approach. The basic idea is to properly configure certain timeout deadlines, so that if the corresponding wireless packets cannot arrive before the timeout deadlines, the distributed entities will independently reset themselves, hence implicitly reset the holistic system. Specifically, we make the following contributions.

(1) We propose a timeout-based CAV collaboration protocol for automatic highway and metered-ramp merging. We formally prove the safety (i.e., guarantee of the CTH safety rule) and liveness of our proposed protocol, even if there are arbitrary wireless packet losses.
(2) We carry out extensive simulations to further verify our proposed protocol. The results show that our protocol can always fulfill the CTH safety rule and liveness despite of arbitrary wireless packet losses.
(3) Furthermore, the simulation results also show significant improvements in the merging efficiency over other solution alternatives. Particularly, the merging success rates are more than 99% better in 11 out of 18 comparison pairs, and 0%(i.e., tied)$\sim$ 71% better in the remaining 7 comparison pairs.

In the following, Section 2 presents related work. Section 3 formulates the problem. Section 4 proposes the protocol and formally proves its properties. Section 5 gives some important observations. Section 6 evaluates our protocol. Section 7 concludes the article and discusses future work.

2 RELATED WORK

Despite the V2X merging solutions listed in Section 1, there is a large volume of literature on purely V2V or purely V2I-based solutions.

In V2V-based approaches, Lu et al. [27, 28] propose a virtual vehicle-based approach to ensure sufficient distance for vehicles to merge into highway via a ramp. Hidas [20] classifies the merging maneuvers into "free", "forced", and "cooperative;" and studies their impact on the traffic flow, showing that "cooperative" merging, followed by "forced" merging, provides the greatest impact on the traffic flow. Xie et al. [44] develop an optimization-based ramp control strategy and a simulation platform to assess the potential safety and mobility benefits of V2V cooperative merging. Kazerooni et al. [23] and Heim et al. [19] present interactive protocol for merging, in which vehicles use both V2V communications and sensing for cooperation and safety guarantee.

In V2I-based approaches, Jiang et al. [21] use a V2I-based dynamic merge assistance method, to improve merging efficiency and safety. Letter et al. [26] present a longitudinal freeway merging control algorithm for maximizing the average travel velocity of CAVs. Raravi et al. [35] propose an approach for automatic merge control system, where an infrastructure node plans the merging sequences. Pueboobpaphan et al. [33] discuss an algorithm, where trajectories are planned with a safety zone around the ramp CAV. Adjustments based on the planning are continually relayed to the highway CAVs to accommodate the ramp CAV.

All of the above works, however, do not deal with arbitrary wireless packet losses; and how to adapt them to guarantee CTH safety for all vehicles at all time are still open problems.

There are various timeout (aka "lease") based distributed protocols [18, 38]. However, these protocols are not designed for highway and ramp merging, and neither for the CTH safety guarantee.

A less-than-two-full-page ***Work-in-Progress*** (**WiP**) abstract (not an article) of this work is published in a conference's WiP session [17]. Permission to reuse the contents of [17] are obtained (©2023 IEEE. Reprinted, with permission, from [17]). Compared to this article, the WiP abstract does not provide any intuitive explanation, narrative definition, theoretical proof, or evaluation of

the proposed solution. The introduction, related work, and problem formulation are covered but very brief (about one page total, with only three references). Prototypes of Figures 1, 3–5, Definition 1, a conjecture of Theorem 1, as well as Ineq. (1), (6) have appeared in [17]. That is, this sentence occupies a unique paragraph by itself. The content of this article is also included in the PhD dissertation of the first author [16].

3 PROBLEM FORMULATION

In this section, we present the assumptions on CAV driving dynamics, describe the highway and metered-ramp merging scenario, and specify the demanded CTH safety.

3.1 Assumptions on Driving Dynamics

Vehicular driving dynamics modeling is nontrivial (interested readers can refer to [34]). Fortunately, we can make the following assumptions.

3.1.1 CAV Acceleration. We assume the CAV acceleration strategy along a straight lane is fixed. Specifically, given the initial speed (for straight lanes, we only need to discuss speed, instead of velocity) $v_{\mathrm{a}}^{\mathrm{low}}$ and the target speed $v_{\mathrm{a}}^{\mathrm{high}}$ (where $0 \leqslant v_{\mathrm{a}}^{\mathrm{low}} < v_{\mathrm{a}}^{\mathrm{high}}$, and note in this article, we assume vehicles cannot move backward), suppose currently the acceleration process has been going on for τ_{a} seconds ($\tau_{\mathrm{a}} \geqslant 0$) and has not yet finished, then the CAV's current acceleration value is fixed, and is a function of $v_{\mathrm{a}}^{\mathrm{low}}$, $v_{\mathrm{a}}^{\mathrm{high}}$, and τ_{a}. Denote this function as $\mathrm{acc}(v_{\mathrm{a}}^{\mathrm{low}}, v_{\mathrm{a}}^{\mathrm{high}}, \tau_{\mathrm{a}})$. This function implies that the current *speed* of the CAV is also a function of $v_{\mathrm{a}}^{\mathrm{low}}$, $v_{\mathrm{a}}^{\mathrm{high}}$, and τ_{a}, which can be denoted as $v_{\mathrm{a}}(v_{\mathrm{a}}^{\mathrm{low}}, v_{\mathrm{a}}^{\mathrm{high}}, \tau_{\mathrm{a}})$. This in turn implies that the total duration and distance needed to accelerate from $v_{\mathrm{a}}^{\mathrm{low}}$ to $v_{\mathrm{a}}^{\mathrm{high}}$ is a function of $v_{\mathrm{a}}^{\mathrm{low}}$ and $v_{\mathrm{a}}^{\mathrm{high}}$. We denote this duration and this distance to be respectively $\delta_{\mathrm{a}}(v_{\mathrm{a}}^{\mathrm{low}}, v_{\mathrm{a}}^{\mathrm{high}})$ and $d_{\mathrm{a}}(v_{\mathrm{a}}^{\mathrm{low}}, v_{\mathrm{a}}^{\mathrm{high}})$. Furthermore, we have

Assumption 1. *We assume the acceleration process as per* $\mathrm{acc}(v_{\mathrm{a}}^{\mathrm{low}}, v_{\mathrm{a}}^{\mathrm{high}}, \tau_{\mathrm{a}})$ *(where* $0 \leqslant v_{\mathrm{a}}^{\mathrm{low}} < v_{\mathrm{a}}^{\mathrm{high}}$*) is nonzeno (i.e.,* $\delta_{\mathrm{a}}(v_{\mathrm{a}}^{\mathrm{low}}, v_{\mathrm{a}}^{\mathrm{high}}) > 0$*) and strictly monotonic (i.e., speed will strictly monotonically increase from* $v_{\mathrm{a}}^{\mathrm{low}}$ *to* $v_{\mathrm{a}}^{\mathrm{high}}$ *over time).*

3.1.2 CAV Deceleration. Similar to the acceleration case, in this article, we assume the CAV deceleration strategy along a straight lane is also fixed. Specifically, given the initial speed $v_{\mathrm{d}}^{\mathrm{high}}$ and the target speed $v_{\mathrm{d}}^{\mathrm{low}}$ (where $v_{\mathrm{d}}^{\mathrm{high}} > v_{\mathrm{d}}^{\mathrm{low}} \geqslant 0$), suppose currently the deceleration process has been going on for τ_{d} seconds ($\tau_{\mathrm{d}} \geqslant 0$) and has not yet finished, then the CAV's current acceleration value is fixed, and is a function of $v_{\mathrm{d}}^{\mathrm{high}}$, $v_{\mathrm{d}}^{\mathrm{low}}$, and τ_{d}. Denote this function as $\mathrm{dec}(v_{\mathrm{d}}^{\mathrm{high}}, v_{\mathrm{d}}^{\mathrm{low}}, \tau_{\mathrm{d}})$. This function implies that the current *speed* of the CAV is also a function of $v_{\mathrm{d}}^{\mathrm{high}}$, $v_{\mathrm{d}}^{\mathrm{low}}$, and τ_{d}, which can be denoted as $v_{\mathrm{d}}(v_{\mathrm{d}}^{\mathrm{high}}, v_{\mathrm{d}}^{\mathrm{low}}, \tau_{\mathrm{d}})$. This in turn implies that the total duration and distance needed to decelerate from $v_{\mathrm{d}}^{\mathrm{high}}$ to $v_{\mathrm{d}}^{\mathrm{low}}$ is a function of $v_{\mathrm{d}}^{\mathrm{high}}$ and $v_{\mathrm{d}}^{\mathrm{low}}$. We denote this duration and this distance to be respectively $\delta_{\mathrm{d}}(v_{\mathrm{d}}^{\mathrm{high}}, v_{\mathrm{d}}^{\mathrm{low}})$ and $d_{\mathrm{d}}(v_{\mathrm{d}}^{\mathrm{high}}, v_{\mathrm{d}}^{\mathrm{low}})$. Furthermore, we have

Assumption 2. *We assume the deceleration process as per* $\mathrm{dec}(v_{\mathrm{d}}^{\mathrm{high}}, v_{\mathrm{d}}^{\mathrm{low}}, \tau_{\mathrm{d}})$ *(where* $v_{\mathrm{d}}^{\mathrm{high}} > v_{\mathrm{d}}^{\mathrm{low}} \geqslant 0$ *) is nonzeno (i.e.,* $\delta_{\mathrm{d}}(v_{\mathrm{d}}^{\mathrm{high}}, v_{\mathrm{d}}^{\mathrm{low}}) > 0$*) and strictly monotonic (i.e., speed will strictly monotonically decrease from* $v_{\mathrm{d}}^{\mathrm{high}}$ *to* $v_{\mathrm{d}}^{\mathrm{low}}$ *over time).*

3.2 Merging Scenario and CTH Safety Rule

Figure 1 shows the highway and metered-ramp merging scenario in a bird's-eye view. We assume a metered-ramp leads to a straight highway lane. Mathematically, the highway lane is modeled as a real number axis. The metered-ramp is modeled as a half line. The highway lane and the

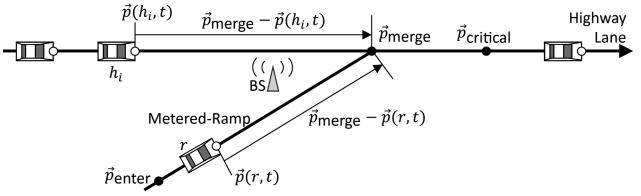


Fig. 1. The highway and metered-ramp merging scenario. $\vec{p}(x,t)$ is CAV x's location at wall clock time t. Drawn based on our intellectual property. Early version appeared in [17] Figure 1.

metered-ramp intersect at point $\vec{p}_{\text{merge}}$, which cuts the highway lane into two halves: the segment $(-\infty, \vec{p}_{\text{merge}}]$ and the segment $(\vec{p}_{\text{merge}}, +\infty)$. The metered-ramp, on the other hand, has a fixed entrance point $\vec{p}_{\text{enter}}$, which is D_r away from $\vec{p}_{\text{merge}}$ (where $D_r \stackrel{\text{def}}{=} |\vec{p}_{\text{merge}} - \vec{p}_{\text{enter}}|$ is a given configuration constant). Any CAV merging into the highway lane via this metered-ramp must first stop at $\vec{p}_{\text{enter}}$ to wait for permission to start. Typically, $\vec{p}_{\text{enter}}$ is where a physical infrastructural ramp meter (such as a red/green traffic light) is installed. But the ramp meter can also be virtual: the CAV simply stops at $\vec{p}_{\text{enter}}$ (e.g., assisted by GPS, or simple visual marks painted on the ramp at $\vec{p}_{\text{enter}}$) and waits for a wireless permission message (from certain participants of the collaborative merging) to start.

For the time being, we abstract every CAV as a point mathematically (see the $\circ$ dots in Figure 1), and let $\vec{p}(x,t)$ denote the location of CAV x at wall clock time t. Considerations on vehicle body length are discussed in the end of this subsection. Suppose the whole system starts at wall clock time t_0, when there are n ($n < +\infty$) CAVs driving at the speed limit v_{lim} (a given configuration constant, the maximum allowed speed on a highway lane, see discussions before Assumption 3) along the highway lane. Without loss of generality, denote the leading CAV to be h_1, which is followed by h_2, so on and so forth, till the last CAV h_n. We call h_is ($i = 1, 2, \ldots, n$) the "*highway CAVs*".

Also, at t_0, a CAV r is stopping at $\vec{p}_{\text{enter}}$ on the metered-ramp waiting for permission to start merging onto the highway lane. We call r the "*ramp CAV*". Once started, r should first accelerate as per $\text{acc}(0, v_{\text{rm}}, \tau_{\text{a}})$ (acc is defined in Section 3.1.1) to the speed of v_{rm} (a given configuration constant, the minimum speed allowed on a highway lane, see discussions before Assumption 3) and then maintains this speed to reach $\vec{p}_{\text{merge}}$. Correspondingly, Ineq. (1) is the configuration prerequisite to make this feasible:

$$d_{\text{a}}(0, v_{\text{rm}}) < D_r \stackrel{\text{def}}{=} |\vec{p}_{\text{merge}} - \vec{p}_{\text{enter}}|, \tag{1}$$

where $d_{\text{a}}(0, v_{\text{rm}})$ is the distance needed to accelerate from speed 0 to v_{rm} (see Section 3.1.1, note the corresponding time cost is $\delta_{\text{a}}(0, v_{\text{rm}})$). The duration cost for r from the start of acceleration to reaching $\vec{p}_{\text{merge}}$ hence is

$$\Delta_r \stackrel{\text{def}}{=} \delta_{\text{a}}(0, v_{\text{rm}}) + \frac{D_r - d_{\text{a}}(0, v_{\text{rm}})}{v_{\text{rm}}}. \tag{2}$$

We also give the following configuration prerequisite:

$$0 < v_{\text{rm}} < v_{\text{lim}}. \tag{3}$$

Correspondingly, once r reaches $\vec{p}_{\text{merge}}$, it will accelerate again according to $\text{acc}(v_{\text{rm}}, v_{\text{lim}}, \tau_{\text{a}})$ to the speed of v_{lim}. The location on the highway lane where r first reaches v_{lim} hence is fixed. Denote it as $\vec{p}_{\text{critical}}$ (see Figure 1).

Note we assume when the merging is completed, all CAVs on the highway shall drive at a same constant speed (specifically, v_{lim}). This is a popular practice adopted by many collaborative CAV driving schemes [10, 25, 36, 40], particularly in the large volume of literature on ***cooperative***

adaptive cruise control **(CACC)** [7, 14]. This practice prevails not only for its simplicity but also for its safety and energy efficiency [7, 10, 14, 25]. On the other hand, the ramp CAV r reaching $\vec{p}_{\mathsf{merge}}$ with v_{rm}, the so-called minimum speed allowed on a highway lane, is a design out of caution. It covers the special case where $v_{\mathsf{rm}} = v_{\mathsf{lim}} - \varepsilon$, where $\varepsilon > 0$ is an arbitrarily small number.

We also assume the following about the CAVs and the road system for the time being.

Assumption 3. *The road system is equipped with V2I infrastructure. Particularly, a base station* BS *resides near $\vec{p}_{\mathsf{merge}}$, which can coordinate the merging between r and $h_1, h_2, \ldots, h_n$. For the time being, we assume* BS *and the highway/metered-ramp lanes are equipped with sufficient wired infrastructure sensors, so that upon* BS*'s request, it can instantly know the distance (from $\vec{p}_{\mathsf{merge}}$) and speed of any CAV (this assumption will be relaxed in Section 5).*

Assumption 4. *Each CAV is equipped with redundant ranging sensors (e.g., laser, radar, ultrasonic, computer vision, V2V communications, and human driver as the last resort), so that for any two consecutive CAVs along the highway lane, the follower CAV can instantly detect the predecessor CAV's speed (e.g., based on the follower's own speed and the relative velocity to the predecessor detected by the ranging sensors). Particularly, due to the redundancy, even if the V2V communications fail (so that the predecessor CAV cannot inform its speed via wireless packets to the follower CAV), the ranging sensing can still function correctly.*

For the above highway and metered-ramp merging scenario, we aim at guaranteeing the CTH safety [8, 11, 14, 30, 37, 39, 47] as specified in the following.

Definition 1 (CTH Safety). Suppose two vehicles (in math point abstraction) x and y are driving in the same direction along a same lane. Suppose x precedes y at time t. Denote the distance between x and y at t as $d(t)$, and y's speed at t as $v_y(t)$. We call $\delta(t) \stackrel{\text{def}}{=} d(t)/v_y(t)$ the *time headway* of y (relative to x) at t. If $\delta(t)$ is no less than a given constant $\Delta^* > 0$, aka the *desired time headway*, then we say the ordered tuple (x, y) is CTH-Δ^* safe at t. In other words, if $d(t) \geqslant v_y(t)\Delta^*$, then we say (x, y) is CTH-Δ^* safe at t.

Assumption 5. *$\forall i \in \{1, 2, \ldots, n-1\}$, (h_i, h_{i+1}) is CTH-Δ^* safe at t_0.*

Intuitively, suppose a lane has both a minimum speed limit v^{min} and a maximum speed limit v^{max}, Δ^* should be set to $\Delta^{**} + \frac{D_0}{v^{\mathsf{min}}}$, where Δ^{**} is the maximum duration needed to stop a vehicle at any speed $v \leqslant v^{\mathsf{max}}$ using emergency braking (which could be different from the normal deceleration dec, but should be monotonic), and D_0 is the maximum vehicle body length. This way, CTH-Δ^* safety rule guarantees y will never hit x, even if x can abruptly stop at any time on the lane.

4 SOLUTION

In this section, we propose a protocol to realize the aforementioned highway and metered-ramp merging (see Section 3.2) and prove its guarantee of the CTH safety and liveness, even under arbitrary wireless packet losses.

4.1 Heuristics

The heuristics of our proposed protocol are illustrated by the automata sketches in Figure 2.

Initially, the base station BS, the ramp CAV r, and the highway CAVs h_i ($i = 1, 2, \ldots, n$) all dwell in their respective "Init" mode. Then the ramp CAV r requests permission to start the merging by sending a "MergeReq" wireless packet to BS (see event "SendMergeReq" in Figure 2(b)). If BS receives this wireless packet, it triggers the "GotMergeReq" event (see event "GotMergeReq" in Figure 2(a)). As the *action* (i.e., the handling routine) is carried out by this event, BS finds the approaching highway CAV closet to $\vec{p}_{\mathsf{merge}}$, and names it coop (for "Cooperator"). BS then enters the transient mode of "L_0" to take further actions based on coop's distance to $\vec{p}_{\mathsf{merge}}$. Specifically,

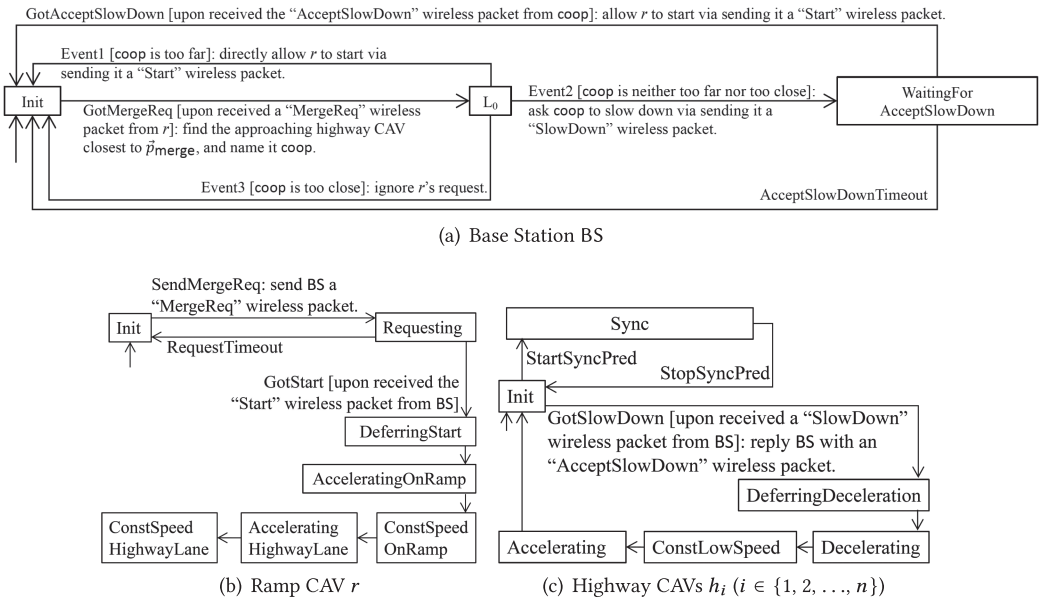


Fig. 2. Automata Sketches. Rectangles are modes, arrows between modes are events, and the arrow without source mode indicates the initial mode in the respective automata sketches. Texts in "[]" are the triggering conditions (aka *guards*) for the corresponding events; texts after the ":" are the *actions* to be carried out once the corresponding events happen.

(1) If coop is too far away from $\vec{p}_{\mathsf{merge}}$, BS will directly allow r to start by sending it a "Start" wireless packet (see "Event1" in Figure 2(a)).
(2) If coop is too close to $\vec{p}_{\mathsf{merge}}$, r's merge request is ignored and r has to request again in the future (see "Event3" in Figure 2(a)).
(3) If coop is neither too far away nor too close to $\vec{p}_{\mathsf{merge}}$ (see "Event2" in Figure 2(a)), BS first sends a "SlowDown" wireless packet to coop to request it to decelerate (i.e., to yield). If coop receives this packet, it will acknowledge BS with an "AcceptSlowDown" wireless packet and start a deceleration routine (see "GotSlowDown" event in Figure 2(c)). Upon reception of the "AcceptSlowDown" wireless packet, BS will send a "Start" wireless packet to r to start its merging routine (see "GotAcceptSlowDown" event in Figure 2(a)). Upon reception of the "Start" wireless packet (see "GotStart" event in Figure 2(b)), r will start and accelerate. Once r reaches $\vec{p}_{\mathsf{merge}}$, it will accelerate to v_{lim}, and later coop will also accelerate to v_{lim}. In addition, when a highway CAV sees its close (current distance is within a certain threshold) predecessor CAV decelerates or accelerates, it will do the same (i.e., "*synchronize*" with the predecessor, see mode "Sync" in Figure 2(c)).

For the above cases, how "far" is "too far," how "close" is "too close," and how to configure the parameters to achieve the CTH-Δ^* safety are non-trivial problems. We will clarify them in the detailed protocol design and analysis (see Sections 4.2 and 4.3).

Another challenge is the possibility of arbitrary wireless packet losses. What if the "MergeReq," "SlowDown," "AcceptSlowDown," and/or "Start" wireless packets are lost? Can the CTH-Δ^* safety still sustain? Can the CAVs still reset themselves, instead of being stuck in a mode forever? Can the CAVs still merge efficiently?

To address these concerns, we propose to deploy the "*lease*" design philosophy for distributed systems [18, 38]. A "*lease*" is an agreement on timeout, contracted since the early stage of a

distributed collaboration. After the lease is contracted, if wireless packets are lost, the affected entities can reset themselves when the agreed timeout is reached (by looking at their respective local clocks, hence need no more communications). In Figure 2, nearly every mode has its timeout configuration. The exact configurations to choose are also non-trivial problems that affect the CTH-Δ^* safety, system liveness, and efficiency. The details and analysis are also elaborated in Sections 4.2 and 4.3. The efficiency is evaluated in Section 6.

4.2 Proposed Protocol

We propose our detailed protocol by expanding the automata sketches of Figure 2 with the heuristics described in Section 4.1. The resulting full-fledged hybrid automata [4] A_{BS} (see Figure 3), A_r (see Figure 4), and A_i (see Figure 5), respectively, define the protocol behaviors of the base station BS, the ramp CAV r, and the highway CAV h_i ($i = 1, 2, \ldots, n$). These behaviors are explained as follows; a symbol list is also provided in Appendix A for the reader's convenience.

Base Station BS *protocol behaviors (illustrated by hybrid automaton A_{BS} in Figure 3)*:

(1) At any time instance, the base station BS dwells in one of the following modes: "Init," "L_0," and "WaitingForAcceptSlowDown."

(2) Initially, BS dwells in the "Init" mode, and has its local clock τ's initial value set randomly from $[0, \Delta_{\mathrm{BS}}^{\mathrm{min}}]$ (e.g., as per uniform distribution), where $\Delta_{\mathrm{BS}}^{\mathrm{min}} > 0$ is a configuration constant.

(3) When dwelling in mode "Init", if a "MergeReq" wireless packet is received from the ramp CAV r, and BS has been continuously dwelling in "Init" for at least $\Delta_{\mathrm{BS}}^{\mathrm{min}}$ seconds (i.e., $\tau > \Delta_{\mathrm{BS}}^{\mathrm{min}}$), then BS triggers the "GotMergeReq" event. This event carries out the following action (see event "GotMergeReq" in Figure 3):

Step1 **IF** currently there is no highway CAV approaching BS (i.e., if $\nexists$ vehicle on highway lane segment $(-\infty, \vec{p}_{\mathrm{merge}}]$) **THEN** set $\hat{\delta}_{\mathrm{coop}}$ to $+\infty$.

Step2 **ELSE**

Step2.1 **IF** coop is undefined, **THEN** set coop as the current closest highway CAV approaching BS (i.e., the current vehicle closest to $\vec{p}_{\mathrm{merge}}$ on the highway lane segment $(-\infty, \vec{p}_{\mathrm{merge}}]$);

Step2.2 set $\hat{\delta}_{\mathrm{coop}}$ to $|\vec{p}_{\mathrm{merge}} - \vec{p}(\mathrm{coop}, t_1)|/v_{\mathrm{lim}}$, where t_1 is the current wall clock time (i.e., $|\vec{p}_{\mathrm{merge}} - \vec{p}(\mathrm{coop}, t_1)|$ is the current distance between $\vec{p}_{\mathrm{merge}}$ and the coop).

After the above action, BS enters the transient mode "L_0."

(4) Mode "L_0" is a transient mode that BS cannot stay. Upon entrance to "L_0," BS immediately triggers one of the following events (see "Event1," "Event2," and "Event3," respectively, in Figure 3):

Case1 (Event1) If the highway CAV coop is too far from the merging point $\vec{p}_{\mathrm{merge}}$, specifically, if $\hat{\delta}_{\mathrm{coop}} \geqslant \Delta_r + \Delta^* + \Delta_1$, where

$$\Delta_1 \stackrel{\mathrm{def}}{=} \delta_{\mathrm{a}}(v_{\mathrm{rm}}, v_{\mathrm{lim}}) - \frac{d_{\mathrm{a}}(v_{\mathrm{rm}}, v_{\mathrm{lim}})}{v_{\mathrm{lim}}}, \quad \text{(note Ineq. (3) implies } \Delta_1 > 0\text{)}, \tag{4}$$

then BS triggers "Event1." This event carries out the following sequential action: send a "Start" wireless packet (with the data payload of 0) to the ramp CAV r, telling r to start immediately (i.e., with 0 delay); set the local clock τ to 0; undefine coop.
After the above action, BS returns to mode "Init."

Case2 (Event2) If coop is neither too far nor too close to $\vec{p}_{\mathrm{merge}}$, specifically, if $\Delta_r + \Delta^* + \Delta_1 > \hat{\delta}_{\mathrm{coop}} > \Delta_2$, where

$$\Delta_2 \stackrel{\mathrm{def}}{=} (d_{\mathrm{d}}(v_{\mathrm{lim}}, v_{\mathrm{rm}}) + v_{\mathrm{rm}}(\Delta_r + \Delta^* - \delta_{\mathrm{d}}(v_{\mathrm{lim}}, v_{\mathrm{rm}})))/v_{\mathrm{lim}}, \tag{5}$$

then BS triggers "Event2." This event carries out the following sequential action:
set δ_{defer} to $\hat{\delta}_{\mathsf{coop}} - \Delta_2$; send a "SlowDown" wireless packet (with the data payload of δ_{defer}) to coop, telling it to slow down in δ_{defer} seconds; set the local clock τ to 0.
After the above action, BS enters mode "WaitingForAcceptSlowDown" for coop's reply.
Note we enforce the following configuration prerequisite:

$$\Delta^* < \delta_{\mathsf{d}}(v_{\mathsf{lim}}, v_{\mathsf{rm}}) < \Delta_r, \tag{6}$$

which implies $\Delta_2 > 0$, and also implies

$$\Delta_r + \Delta^* + \Delta_1 > \Delta_2, \tag{7}$$

because (6)

$$\begin{aligned}
\Rightarrow \quad & (v_{\mathsf{lim}} - v_{\mathsf{rm}})(\Delta_r + \Delta^*) + v_{\mathsf{lim}}\delta_{\mathsf{a}}(v_{\mathsf{rm}}, v_{\mathsf{lim}}) + v_{\mathsf{rm}}\delta_{\mathsf{d}}(v_{\mathsf{lim}}, v_{\mathsf{rm}}) \\
& > v_{\mathsf{lim}}\delta_{\mathsf{d}}(v_{\mathsf{lim}}, v_{\mathsf{rm}}) + v_{\mathsf{lim}}\delta_{\mathsf{a}}(v_{\mathsf{rm}}, v_{\mathsf{lim}}) > d_{\mathsf{d}}(v_{\mathsf{lim}}, v_{\mathsf{rm}}) + d_{\mathsf{a}}(v_{\mathsf{rm}}, v_{\mathsf{lim}}) \\
\Rightarrow \quad & \Delta_r + \Delta^* + \delta_{\mathsf{a}}(v_{\mathsf{rm}}, v_{\mathsf{lim}}) - d_{\mathsf{a}}(v_{\mathsf{rm}}, v_{\mathsf{lim}})/v_{\mathsf{lim}} = \Delta_r + \Delta^* + \Delta_1 \\
& > (v_{\mathsf{rm}}(\Delta_r + \Delta^* - \delta_{\mathsf{d}}(v_{\mathsf{lim}}, v_{\mathsf{rm}})) + d_{\mathsf{d}}(v_{\mathsf{lim}}, v_{\mathsf{rm}}))/v_{\mathsf{lim}} = \Delta_2.
\end{aligned}$$

Ineq. (7) ensures the guards for "Event1" and "Event2" (see Figure 3) are valid and non-overlapping.

Case3 (Event3) Otherwise, i.e., if coop is too close to $\vec{p}_{\mathsf{merge}}$, specifically, $\hat{\delta}_{\mathsf{coop}} \leqslant \Delta_2$, then BS triggers "Event3." This event carries out the following sequential action:
set the local clock τ to 0; undefine coop.
After the above action, BS returns to mode "Init."

(5) When dwelling in mode "WaitingForAcceptSlowDown," the local clock τ grows continuously (i.e., $\dot{\tau} = 1$), and must not exceed its range constraint of $[0, \max\{\Delta_{\mathsf{nonzeno}}, \delta_{\mathsf{defer}}\}]$, where $\Delta_{\mathsf{nonzeno}} > 0$ is a configuration constant, and δ_{defer} is set by "Event2." In this mode, BS may trigger one of the following two events (see "GotAcceptSlowDown" and "AcceptSlowDownTimeout" events respectively in Figure 3):

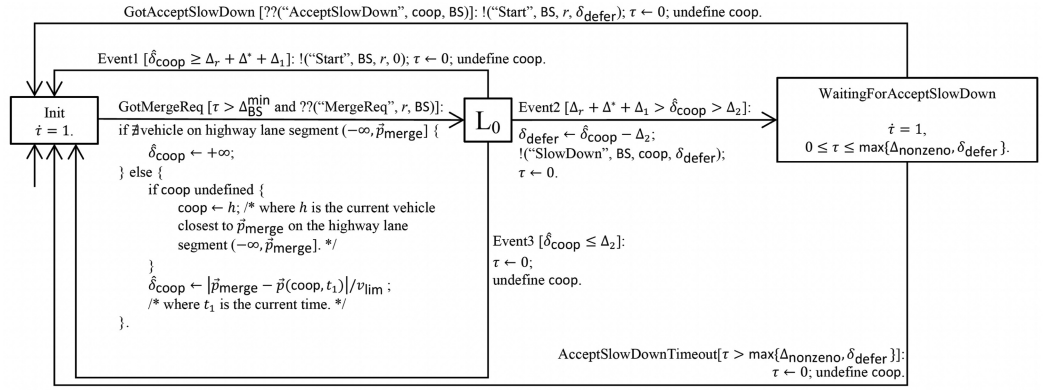
Case1 (GotAcceptSlowDown) If before τ exceeds $\max\{\Delta_{\mathsf{nonzeno}}, \delta_{\mathsf{defer}}\}$, an "AcceptSlowDown" wireless packet is received from coop, then BS triggers the "GotAcceptSlowDown" event. This event carries out the following sequential action:
send a "Start" wireless packet to r, telling r to start in δ_{defer} seconds (with the packet data payload of δ_{defer}); set the local clock τ to 0; undefine coop.
After the above action, BS returns to mode "Init."

Case2 (AcceptSlowDownTimeout) If local clock τ exceeds $\max\{\Delta_{\mathsf{nonzeno}}, \delta_{\mathsf{defer}}\}$, then BS gives up waiting for the "AcceptSlowDown" wireless packet from coop, and triggers the timeout event "AcceptSlowDownTimeout." This event carries out the following sequential action:
set the local clock τ to 0; undefine coop.
After the above action, BS returns to mode "Init."

Ramp CAV r protocol behaviors (illustrated by hybrid automaton A_r in Figure 4):

(1) At any time instance, the ramp CAV r dwells in one of the following modes: "Init," "Requesting," "DeferringStart," "AcceleratingOnRamp," "ConstSpeedOnRamp," "AcceleratingHighwayLane," and "ConstSpeedHighwayLane."

(2) Initially, r dwells in the "Init" mode, stops at $\vec{p}_{\mathsf{enter}}$ (i.e., $\vec{p}(r, t) = \vec{p}_{\mathsf{enter}}$), and has its local clock τ's initial value set to 0.



Legends: Each rectangle box indicates a *hybrid automaton mode* (simplified as "*mode*" in the following). Inside a mode, the top line is the mode's name (it is local to the respective hybrid automata), the rest describes the constraints (e.g., a dwelling duration constraint like $0 \leq \tau \leq \Delta_{\mathsf{nonzeno}}$) and continuous domain dynamics (typically specified with differential equations, e.g., $\dot{\tau} = 1$) related to the mode.
"L_0" is a transient mode, whose maximum dwelling duration constraint is 0 seconds, i.e., when the execution enters "L_0", it must exit "L_0" immediately (via a qualified event).
The arrow without source mode indicates the starting mode of execution (τ's initial value is uniformly sampled from $[0, \Delta^{\mathsf{min}}_{\mathsf{BS}}]$). Other arrows represent discrete *events* for the system. Annotations to each event arrow have the following meanings. Before the ":" is the optional event name and the *guard* (quoted by the brackets "[]"), i.e., the triggering condition for the event. Particularly, "??(x)" means the event is triggered upon the reception of a wireless packet "(x)" (a wireless packet (x) is a tuple of three or four elements, respectively the type, sender, intended receiver, and optional data payload of the packet). Note a sent wireless packet is not always received: the packet could be lost arbitrarily. After the ":" is the action carried out by the event. Particularly, "!(y)" means a wireless packet (y) is sent; and "←" means value assignment. Same legends also apply to Figures 4 and 5.

Fig. 3. Hybrid automaton A_{BS} for the base station BS. Drawn based on our intellectual property. Early version appeared in [17] Figure 2.

(3) When dwelling in mode "Init," r is stopping (i.e., $|\dot{\vec{p}}(r,t)| = 0$) and the local clock τ grows continuously (i.e., $\dot{\tau} = 1$). But when τ exceeds $\Delta_{\mathsf{nonzeno}}$, r triggers the "SendMergeReq" event. This event carries out the following sequential action: send a "MergeReq" wireless packet to BS; reset τ to 0.
After the above action, r enters mode "Requesting" to wait for BS's reply.

(4) When dwelling in mode "Requesting," r is stopping (i.e., $|\dot{\vec{p}}(r,t)| = 0$) and τ grows continuously. If before τ exceeds $\Delta_{\mathsf{nonzeno}}$, the reply from BS, i.e., a "Start" wireless packet (with the data payload of value σ_{defer}), is received, then r triggers the "GotStart" event, resets τ to 0, and enters the "DeferringStart" mode. Otherwise, if no reply from BS is received till τ exceeds $\Delta_{\mathsf{nonzeno}}$, then r triggers the "RequestTimeout" event, resets τ to 0, and returns to mode "Init," giving up waiting for BS's reply.

(5) Once r enters the "DeferringStart" mode, r will first wait for σ_{defer} seconds, then (enter "AcceleratingOnRamp" mode) accelerate as per $\mathrm{acc}(0, v_{\mathsf{rm}}, \tau)$ ($\tau \in [0, \delta_{\mathsf{a}}(0, v_{\mathsf{rm}})]$) to v_{rm}, (enter "ConstSpeedOnRamp" mode) maintain this speed till passed $\vec{p}_{\mathsf{merge}}$, (enter "AcceleratingHighwayLane" mode) accelerate as per $\mathrm{acc}(v_{\mathsf{rm}}, v_{\mathsf{lim}}, \tau)$ ($\tau \in [0, \delta_{\mathsf{a}}(v_{\mathsf{rm}}, v_{\mathsf{lim}})]$)

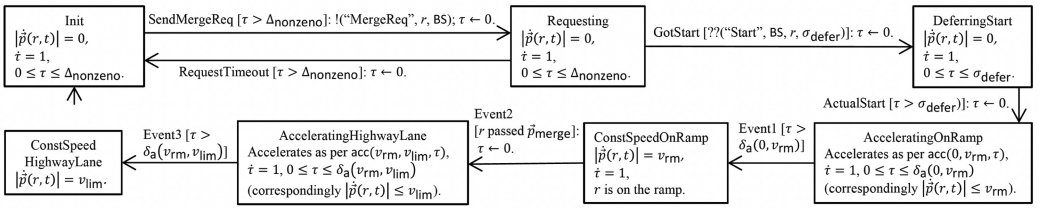


Fig. 4. Hybrid automaton A_r for the ramp CAV r (τ's initial value is set to 0). Drawn based on our intellectual property. Early version appeared in [17] Figure 3(a).

to v_{lim}, and (enter "ConstSpeed" mode) maintain v_{lim} on the highway lane, finishing the merging.

Highway CAV h_i ($i \in \{1, \ldots, n\}$) protocol behaviors (illustrated by hybrid automaton A_i in Figure 5):

(1) At any time instance, highway CAV h_i ($i \in \{1, \ldots, n\}$) dwells in one of the following modes: "Init," "DeferringDeceleration," "Decelerating," "ConstLowSpeed," "Accelerating," and "Sync."
(2) Initially, h_i dwells in mode "Init," drives at speed v_{lim}, and the state local variable is set to "Init."
(3) When dwelling in mode "Init," h_i may trigger one of the following two events (see "GotSlow-Down" and "StartSyncPred" events, respectively, in Figure 5):
   Case1 (GotSlowDown) If a "SlowDown" wireless packet is received from BS (with the data payload of value δ_{defer}), then h_i triggers the "GotSlowDown" event. This event carries out the following sequential action (see event "GotSlowDown" in Figure 5): send the "AcceptSlowDown" wireless packet to BS; set state to "Coop"; set local clock τ to 0. After the above action, h_i enters mode "DeferringDeceleration."
   Case2 (StartSyncPred, only applicable for $i > 1$) If h_{i-1} is no more than

$$D_1 \stackrel{\text{def}}{=} v_{\text{lim}}(\Delta_r + 2\Delta^* + \Delta_1 - \Delta_2) \qquad \text{(note Ineq. (7) implies } D_1 > 0) \tag{8}$$

   distance ahead of h_i and starts to decelerate from speed v_{lim}, then h_i triggers the "StartSyncPred" event, sets state to "Sync," and enters mode "Sync."
(4) Once h_i enters the "DeferringDeceleration" mode, h_i will first wait for δ_{defer} seconds, then (enter "Decelerating" mode with local clock τ reset to 0) decelerate as per $\text{dec}(v_{\text{lim}}, v_{\text{rm}}, \tau)$ ($\tau \in [0, \delta_{\text{d}}(v_{\text{lim}}, v_{\text{rm}}]$) to v_{rm},
(enter "ConstLowSpeed" mode without changing τ) maintain this speed till local clock τ exceeds $\Delta_r + \Delta^*$ seconds (note Ineq. (6) implies $\Delta_r + \Delta^* > \delta_{\text{d}}(v_{\text{lim}}, v_{\text{rm}})$),
(enter "Accelerating" mode with local clock τ reset to 0) accelerate as per $\text{acc}(v_{\text{rm}}, v_{\text{lim}}, \tau)$ ($\tau \in [0, \delta_{\text{a}}(v_{\text{rm}}, v_{\text{lim}}]$) to v_{lim},
and return to mode "Init" (with state reset to "Init").
(5) Once h_i enters the "Sync" mode, h_i keeps its speed the same as h_{i-1}'s, until h_{i-1} recovers its speed of v_{lim}. At that moment, h_i triggers the "StopSyncPred" event, sets state to "Init," and returns to mode "Init."

We claim the above protocol for BS, r, and h_i ($i \in \{1, \ldots, n\}$) guarantees CTH safety and liveness (i.e., entities will not stuck in any mode), even under arbitrary wireless packet losses. In the next subsection, we shall rigorously describe and prove these properties.

4.3 Analysis

We claim the following theorem on sufficient conditions for safety and liveness.

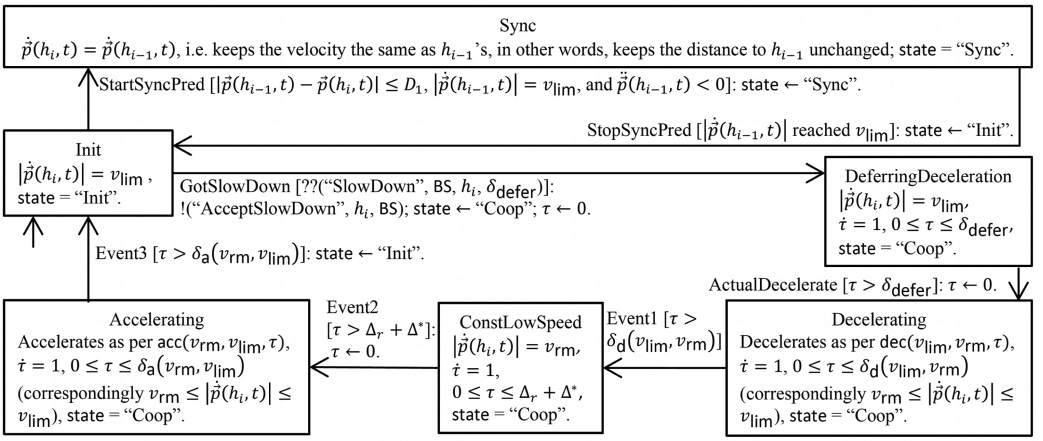


Fig. 5. Hybrid automaton A_i for the highway CAV h_i ($i \in \{1, 2, \ldots, n\}$); note as h_0 does not exist, for h_1, the event "StartSyncPred" can never happen. Drawn based on our intellectual property. Early version appeared in [17] Figure 3(b).

Theorem 1. *Suppose configuration constants of A_{BS}, A_r, and A_i ($i = 1, 2, \ldots, n$) comply with the following constraints:*

(c1) *aforementioned constraints: Ineq.* (1), (3), (6), $\Delta^* > 0$, *and* $\Delta_{\mathsf{nonzeno}} > 0$;
(c2) $\Delta^{\min}_{\mathsf{BS}} > \Delta^{\max}_{\mathsf{coop}} + \Delta_{\mathsf{nonzeno}}$, *where*

$$\Delta^{\max}_{\mathsf{coop}} \stackrel{\text{def}}{=} \delta^{\max}_{\mathsf{defer}} + \Delta_r + \Delta^* + \delta_{\mathsf{a}}(v_{\mathsf{rm}}, v_{\mathsf{lim}}),\ \delta^{\max}_{\mathsf{defer}} \stackrel{\text{def}}{=} \hat{\delta}^{\max}_{\mathsf{coop}} - \Delta_2,\ \text{and}\ \hat{\delta}^{\max}_{\mathsf{coop}} \stackrel{\text{def}}{=} \Delta_r + \Delta^* + \Delta_1;$$

(c3) $v_{\mathsf{rm}}\Delta_r \geqslant v_{\mathsf{lim}}\Delta^*$;
(c4) $\Delta_{\mathsf{nonzeno}} < \Delta_r + \Delta^* + \delta_{\mathsf{a}}(v_{\mathsf{rm}}, v_{\mathsf{lim}})$.

Then we have the following claims.

Claim 1 (Safety). $\forall t \in [t_0, +\infty)$, *for any two CAVs x and y on the highway lane, one and only one of the following sustains: (x, y) is CTH-Δ^* safe at t, or (y, x) is CTH-Δ^* safe at t.*

Claim 2 (Liveness (Automatic Reset)). *suppose at $t_1 \in [t_0, +\infty)$, the base station* BS *leaves hybrid automaton A_{BS} mode "Init", while highway CAV h_is ($i = 1, 2, \ldots, n$) are all dwelling in respective A_i mode "Init", let*

$$\Delta^{\max}_{\mathsf{reset}} \stackrel{\text{def}}{=} \Delta^{\max}_{\mathsf{coop}} + \Delta_{\mathsf{nonzeno}} + \delta_{\mathsf{a}}(v_{\mathsf{rm}}, v_{\mathsf{lim}}), \tag{9}$$

*then $\exists t_2 \in (t_1, t_1 + \Delta^{\max}_{\mathsf{reset}}]$ s.t. either (**Stable State 1**) at t_2, h_1, $h_2, \ldots, h_n$,* BS*, and the ramp CAV r are in respective hybrid automata mode "Init"; or (**Stable State 2**) at t_2, h_1, $h_2, \ldots, h_n$, and* BS *are in respective hybrid automata mode "Init" and r is in A_r's mode "ConstSpeedHighwayLane".*

In order to prove Theorem 1, we need to first propose/prove several definitions and lemmas.

Definition 2 (Coop-duration). For a highway CAV h_i ($i \in \{1, 2, \ldots, n\}$), suppose its hybrid automaton variable, state, changes from "Init" to "Coop" at $t_1 \in [t_0, +\infty)$, then as per Figure 5, the state must change back to "Init" at some finite t_2 (where $t_1 < t_2 \leqslant t_1 + \Delta^{\max}_{\mathsf{coop}}$, see **(c2)** for the definition of $\Delta^{\max}_{\mathsf{coop}}$). That is, $\forall t \in (t_1, t_2]$, state ="Coop";[1] and at t_2^+, state = "Init". We call $(t_1, t_2]$

[1]Note, if we regard hybrid automaton discrete variables' values are left continuous along the time axis, then at t_1, we regard state ="Init".

a "*coop-duration*". Note as per Figures 3 and 5, it is easy to see that $\Delta_{\text{coop}}^{\max}$ is the maximum possible time length for a coop-duration.

Lemma 1. *Any two coop-durations* $(t_1, t_2]$ *and* $(t_3, t_4]$ *respectively belonging to two different CAVs can never overlap nor connect, i.e.,* $[t_1, t_2] \cap [t_3, t_4] = \varnothing$.

Proof. Suppose $[t_1, t_2] \cap [t_3, t_4] \neq \varnothing$ and suppose $t_5 \in [t_1, t_2] \cap [t_3, t_4]$. Then $t_1 \in [t_5 - \Delta_{\text{coop}}^{\max}, t_5]$ and $t_3 \in [t_5 - \Delta_{\text{coop}}^{\max}, t_5]$, therefore $|t_1 - t_3| \leqslant \Delta_{\text{coop}}^{\max}$. This means BS sends two different "SlowDown" packets within $\Delta_{\text{coop}}^{\max}$. This contradicts (**c2**), where $\Delta_{\text{BS}}^{\min} > \Delta_{\text{coop}}^{\max}$. □

Lemma 2. *Any two coop-durations* $(t_1, t_2]$ *and* $(t_3, t_4]$ *can never overlap nor connect, i.e.,* $[t_1, t_2] \cap [t_3, t_4] = \varnothing$.

Proof. In addition to Lemma 1, applying similar reasonings, we can prove coop-durations of a same highway CAV cannot overlap nor connect. □

Lemma 3. $\forall t \in [t_0, +\infty)$*, if no highway CAV is in coop-duration at* t*, then all highway CAVs (i.e.,* $h_1, h_2, \ldots, h_n$*) are in "Init" mode at* t.

Proof. According to Figure 5, if $\exists h_i$, whose state = "Sync" at t, then there must be an h_j in a coop-duration at t. □

Lemma 4. *Suppose* $(t_1, t_2] \subseteq [t_0, +\infty)$ *is the first ever happened coop-duration, then* $\forall t \in [t_0, t_2]$, $\forall i \in \{1, 2, \ldots, n-1\}$, (h_i, h_{i+1}) *is CTH-*Δ^* *safe at* t.

Proof. See Appendix B for details. □

Lemma 5. $\forall t \in [t_0, +\infty)$, $\forall i \in \{1, 2, \ldots, n-1\}$, (h_i, h_{i+1}) *is CTH-*Δ^* *safe at* t.

Proof. See Appendix C for details. □

Corollary 1. *Throughout* $[t_0, +\infty)$*, there is no spatial swapping between* h_i *and* h_j *(*$\forall i, j \in \{1, 2, \ldots, n\}$, $i \neq j$*) along the highway lane.*

Proof. Due to Lemma 5, the first swapping never happens. □

Lemma 6. *Suppose ramp CAV* r *reaches* $\vec{p}_{\text{merge}}$ *at* $t_1 \in [t_0, +\infty)$*, then for each* $i \in \{1, 2, \ldots, n\}$*, one and only one of the following claims sustain: (**Claim 1**)* (h_i, r) *is CTH-*Δ^* *safe throughout* $[t_1, +\infty)$*; (**Claim 2**)* (r, h_i) *is CTH-*Δ^* *safe throughout* $[t_1, +\infty)$.

Proof. See Appendix D for details. □

Now we are ready to prove Theorem 1.

Proof of Theorem 1 **Claim 1**:

In case $x, y \in \{h_1, h_2, \ldots, h_n\}$, the claim sustains due to Lemma 5 and Corollary 1 (in case x and y are not consecutive, e.g., $x = h_i$ and $y = h_{i+k}$, where $k > 1$, then due to Corollary 1, the distance between x and y is no less than the distance between h_{i+k-1} and y, hence the CTH-Δ^* safety rule still sustains for (x, y)).

In case $x \in \{h_1, h_2, \ldots, h_n\}$ and $y = r$, or the reverse, the claim sustains due to Lemma 6.

Combining the above two cases, the claim sustains. □ (‡)

Proof of Theorem 1 **Claim 2**:

Case 1: "Event1" happens at t_1. Then at t_1^+, BS returns to "Init" and remains there till at least $t_1 + \Delta_{\text{BS}}^{\min}$.

Case 1.1: If r receives the "Start" packet at t_1, then it will be in "ConstSpeedHighwayLane" by $t_1 + \Delta_r + \delta_a(v_{rm}, v_{lim}) < t_1 + \Delta_{BS}^{min}$ (due to (**c2**)). Meanwhile, all $h_1 \sim h_n$ remain in "Init" from t_1 to $t_1 + \Delta_r + \delta_a(v_{rm}, v_{lim})$. Therefore, $t_3 \stackrel{def}{=} t_1 + \Delta_r + \delta_a(v_{rm}, v_{lim})$ is a time instance that matches the claim's description (we call such a time instance a "*valid time instance*" in the following).
Case 1.2: If r did not receive the "Start" packet at t_1. Then, as r sent the "MergeReq" packet at t_1, it will be at "Init" at $t_1 + \Delta_{nonzeno} < t_1 + \Delta_{BS}^{min}$ (due to (**c2**)). Meanwhile, $h_1 \sim h_n$ remains in "Init" at $t_1 + \Delta_{nonzeno}$. Hence $t_4 \stackrel{def}{=} t_1 + \Delta_{nonzeno}$ is a valid time instance.
Case 2: "Event2" happens at t_1. Then by $t_1 + \max\{\Delta_{nonzeno}, \delta_{defer}\}$, BS should have returned to "Init" and remain there till at least $t_1 + \Delta_{BS}^{min}$.

Meanwhile, it will not send another "SlowDown" packet during $(t_1, t_1 + \Delta_{BS}^{min}]$ at least. (♣)
Case 2.1: h_{coop} receives the "SlowDown" packet at t_1. Then the coop-duration starts at t_1 and ends at $t_5 \stackrel{def}{=} t_1 + \delta_{defer} + \Delta_r + \Delta^* + \delta_a(v_{rm}, v_{lim})$.

Meanwhile, as per (**c2**), $\exists \varepsilon \in (0, \Delta_{BS}^{min} - \Delta_{coop}^{max} - \Delta_{nonzeno})$, s.t. $\varepsilon < \delta_a(v_{rm}, v_{lim})$. Let $t_6 \stackrel{def}{=} t_5 + \varepsilon$, and $t_7 \stackrel{def}{=} t_6 + \Delta_{nonzeno}$. Then we have $t_1 + \max\{\Delta_{nonzeno}, \delta_{defer}\} < t_5 < t_6 < t_7 < t_1 + \Delta_{BS}^{min}$ (due to (**c2**), (**c4**)). Hence BS is in "Init" at t_6 and t_7.

Due to (♣), a second coop-duration will not start till after $t_1 + \Delta_{BS}^{min}$. Hence due to Lemmas 2 and 3, we know $h_1 \sim h_n$ are all in "Init" at t_6 and at t_7.
Case 2.1.1 BS receives "AcceptSlowDown" at t_1^+, it sends ("Start", BS, r, δ_{defer}) at t_1^+.

(**a**) r receives the "Start" packet at t_1^+. Then it reaches "ConstSpeedHighwayLane" at $t_1 + \delta_{defer} + \Delta_r + \delta_a(v_{rm}, v_{lim}) < t_6$. Hence t_6 is a valid time instance.

(**b**) r did not receive the "Start" packet at t_1^+. Then at t_6, it must be in "Init" or "Requesting". In this case, if r is in "Init" at t_6. Then t_6 is a valid time instance; If r is in "Requesting" at t_6. Then r must have switched to "Init" at t_7. Then t_7 is a valid time instance.

Combining **a** and **b**, **Case 2.1.1** complies with the claim.
Case 2.1.2 BS does not receive "AcceptSlowDown" at t_1^+. Then it returns to "Init" at $t_1 + \max\{\Delta_{nonzeno}, \delta_{defer}\}$ and remains there till $t_1 + \max\{\Delta_{nonzeno}, \delta_{defer}\} + \Delta_{BS}^{min}$. No "Start" packet was sent.

Then similar to the analysis of item (**b**), if r is in "Init" at t_6. Then t_6 is a valid time instance; If r is in "Requesting" at t_6. Then t_7 is a valid time instance.

Combining **Case 2.1.1** and **Case 2.1.2**, **Case 2.1** complies with the claim.
Case 2.2 h_{coop} does not receive "SlowDown" at t_1. Then nothing happens to $h_1 \sim h_n$ during $[t_1, t_1 + \Delta_{BS}^{min}]$.

Let $t_8 \stackrel{def}{=} t_1 + \max\{\Delta_{nonzeno}, \delta_{defer}\}$, $t_9 \stackrel{def}{=} t_8 + \varepsilon$, $t_{10} \stackrel{def}{=} t_9 + \Delta_{nonzeno}$, where ε is the same ε chosen for **Case 2.1**. Then (**c2**) and (**c4**) imply $0 < t_8 < t_9 < t_{10} < t_1 + \Delta_{BS}^{min}$. Hence at t_9 and t_{10}, BS and $h_1 \sim h_n$ are in "Init". Considering r, we have the following two cases.
Case 2.2.1 r is in "Init" at t_9. Then t_9 is a valid time instance.
Case 2.2.2 r is in "Requesting" at t_9. Then t_{10} is a valid time instance.

Combining **Case 2.2.1** and **Case 2.2.2**, **Case 2.2** complies with the claim.

Combining **Case 2.1** and **Case 2.2**, **Case 2** complies with the claim.
Case 3 "Event3" happens at t_1. Then **BS** returns to "Init" at t_1^+. Nothing happens to $h_1 \sim h_n$ till $t_1 + \Delta_{BS}^{min}$.

Let $t_{11} \stackrel{def}{=} t_1 + \varepsilon$, $t_{12} \stackrel{def}{=} t_{11} + \Delta_{nonzeno}$, where ε is the same ε chosen for **Case 2.1**. Then (**c2**) and (**c4**) imply $t_1 < t_{11} < t_{12} < t_1 + \Delta_{BS}^{min}$. Hence at t_{11} and t_{12}, BS and $h_1 \sim h_n$ are in "Init". Considering r, we have the following two cases.
Case 3.1 r is in "Init" at t_{11}. Then t_{11} is a valid time instance.

Case 3.2 r is in "Requesting" at t_{11}. Then t_{12} is a valid time instance.

Combining **Case 3.1** and **Case 3.2**, Case **Case 3** complies with the claim.

Combining **Case 1**, **Case 2**, and **Case 3**, the claim sustains. □ (‡‡)

Due to (‡) and (‡‡), the theorem sustains.

5 IMPORTANT OBSERVATIONS

We have two important observations regarding Theorem 1's validity based on the design of the proposed protocol and the proof of the theorem.

Relaxation on Assumption 3. BS only needs to be able to instantly know (upon reception of a "MergeReq" packet, see Figure 3) which highway CAV is currently closest to $\vec{p}_{\text{merge}}$ on the segment $[\vec{p}_{\text{merge}} - v_{\text{lim}}(\Delta_r + \Delta^* + \Delta_1), \vec{p}_{\text{merge}}]$, and (if it exists) whether its current distance to $\vec{p}_{\text{merge}}$ is no less than $v_{\text{lim}}(\Delta_r + \Delta^* + \Delta_1)$, or no greater than $v_{\text{lim}}\Delta_2$, or otherwise.

V2V Communication Failures are Irrelevant. V2V communications (if used) are only used in the "Sync" mode of the highway CAV hybrid automaton (see Figure 5), and are only used between two consecutive highway CAVs (h_i and h_{i+1}, where $i = 1, 2, \ldots, n-1$) for three possible cases: to trigger the "StartSyncPred" event, to let h_i inform h_{i+1} of the former's current ranging/velocity/acceleration or to trigger the "StopSyncPred" event. For all these three cases, the V2V communications can be replaced by h_{i+1}'s local ranging sensors (see Assumption 4). Hence V2V communications failures are irrelevant. In case the ranging sensors need line-of-sight, we have the following observations. All the highway CAVs that should be in "Sync" at any time instance t must be following a unique highway CAV h_ι ($\iota \in \{1, 2, \ldots, n\}$) that is in a coop-duration. This implies h_ι must be behind r, if r is after all on the highway lane at t. Therefore, it is impossible that r resides between two *speed synchronizing* highway CAVs (i.e., the predecessor highway CAV is in a coop-duration, while the follower highway CAV is in "Sync"; or both are in "Sync") at t. Therefore, the line-of-sight between two speed synchronizing highway CAVs is available at t.

6 EVALUATION

We carry out simulations to verify the proposed protocol, particularly on the CTH-Δ^* safety guarantee, the liveness (automatic reset) guarantee, and the success rates and time costs of merging.

We also compare the proposed protocol with two other protocols: the *priority-based protocol* adapted from Aoki et al. [5], and the *consensus-based protocol* from Wang et al. [43]. We choose these two protocols because their focus problem contexts are the most similar to ours.

Specifically, Aoki et al. [5] focus on the design of a safe highway metered-ramp merging protocol, with collision avoidance guarantee under arbitrary wireless packet losses. As mentioned in Section 1, how to adapt their protocol to guarantee *CTH safety* under arbitrary wireless packet losses is still an open problem. Fortunately, Aoki et al. [5] mentioned a "baseline priority-based protocol" for comparison purposes in their article's evaluation section. We found a way to adapt this "baseline priority-based protocol" to guarantee CTH safety under arbitrary wireless packet losses. Specifically, the adapted protocol (referred to as the "*priority-based protocol*" in the following) looks exactly the same as our proposed protocol of Section 4.2 (referred to as "*the proposed protocol*" in the following), except that the base station no longer requests highway CAVs to yield. Formally, this means to adapt the hybrid automaton A_{BS} of Figure 3 as follows:

(1) Expand Event3's guard to cover all cases where $\hat{\delta}_{\text{coop}} < \Delta_r + \Delta^* + \Delta_1$;
(2) Delete mode "WaitingForAcceptSlowDown" and event "Event2," "GotAcceptSlowDown," "AcceptSlowDownTimeout."

The proof of CTH guarantee under arbitrary wireless packet losses of the above priority-based protocol follows the corresponding proof for the proposed protocol, as the priority-based protocol is basically a subset of the proposed protocol.

The other comparison alternative, Wang et al. [43]'s consensus-based protocol, is a highway and ramp merging protocol using V2X communications. The protocol can achieve good CTH safety statistically, but it does not focus on CTH *guarantee* under arbitrary wireless packet losses. We choose to compare with this protocol because it covers V2X communications, highway and ramp merging, and CTH safety. Similar to Aoki et al. [5]'s work, the focus problem context does not exactly match ours but is among the closest.

Next, we shall discuss the simulator configurations and the evaluation results.

6.1 Simulation Configuration

We follow the recommendations by the seminal textbook of [34] to configure our simulator. Specifically, CTH safety desired time headway $\Delta^* = 3$s; $\Delta_{\mathsf{BS}}^{\mathsf{min}} = 39.61$s; $\Delta_{\mathsf{nonzeno}} = 0.1$s; $D_r = 300$m; $v_{\mathsf{lim}} = 33.333$m/s; $v_{\mathsf{rm}} = 25$m/s; acceleration and deceleration strategy are set as per [34], which decides $\delta_{\mathsf{a}}(0, v_{\mathsf{rm}}) = 13.01$s, $d_{\mathsf{a}}(0, v_{\mathsf{rm}}) = 200.6840$m, $\delta_{\mathsf{a}}(v_{\mathsf{rm}}, v_{\mathsf{lim}}) = 12.20$s, $d_{\mathsf{a}}(v_{\mathsf{rm}}, v_{\mathsf{lim}}) = 362.3613$m, $\delta_{\mathsf{d}}(v_{\mathsf{lim}}, v_{\mathsf{rm}}) = 3.08$s, and $d_{\mathsf{d}}(v_{\mathsf{lim}}, v_{\mathsf{rm}}) = 90.9735$m. The above configuration further decides other parameters, specifically, Δ_r (see Equation (2)), Δ_1 (see Equation (4)), Δ_2 (see Equation (5)), D_1 (see Equation (8))), and $\Delta_{\mathsf{reset}}^{\mathsf{max}}$ (see Equation(9)). Particularly, $\Delta_{\mathsf{reset}}^{\mathsf{max}} = 50.4$s, which is used in Section 6.3 and Table 2.

Note the above configurations comply with the constraints demanded by Theorem 1, as well as the recommendations of the consensus-based protocol [43].

At the beginning of each simulation trial, our simulator generates n ($n = 120$, 180, or 240, respectively for light, mild, and heavy traffic; n's value is fixed for each individual simulation trial) highway CAVs along the highway lane segment $[-50,000\text{m}, 0\text{m}]$, where the location at 0m is $\vec{p}_{\mathsf{merge}}$. The exact initial locations of the n highway CAVs are randomly chosen as per a pseudo-uniform distribution, which takes into consideration of Assumption 5. Specifically, the pseudo-code is as follows:

Step1 initialize H to empty set;
Step2 **IF** ($|H| \geqslant n$) **THEN** terminate; **ELSE**
  Step2.1 randomly choose a point p on the highway lane segment $[-50,000\text{m}, 0\text{m}]$ as per uniform distribution;
  Step2.2 **IF** p does not violate CTH-Δ^* safety rule with the points already in H **THEN** add p into H; **ELSE** ignore p;
  Step2.3 go back to Step2.

The generated H is the initial location for the highway CAVs for the trial.

Our simulator also adopts a wireless packet loss rate parameter P, whose value is set to 0.1 (i.e., 10%), 0.5 (i.e., 50%), or 0.9 (i.e., 90%) to evaluate the proposed protocol under mild, moderate, and severe wireless packet losses (P's value is fixed for each individual simulation trial).

For each given n and P values, we run 25 simulation trials. Each trial simulates 10 minutes (unless in some exception cases, see the last paragraph of Section 6.3) of a highway and metered-ramp merging scenario.

6.2 Safety

Theorem 1 **Claim 1** is on the CTH-Δ^* safety guarantee. To validate this claim, Table 1 shows the statistics of sampled time headways (relative to the respective immediate predecessor vehicles, see Definition 1) of all vehicles in all simulation trials (for each vehicle simulated, its time headway

Table 1. Simulation Results: Time Headway

Protocols	n	P	Time headway statistics (s)				
			min	median	max	average	std
The Proposed Protocol	120	0.1	**3.0**	9.4	70.3	12.4	9.1
		0.5	**3.0**	9.5	73.1	12.4	9.4
		0.9	**3.0**	9.7	81.1	12.5	9.3
Priority-based Protocol		0.1	**3.0**	9.6	81.9	12.4	9.0
		0.5	**3.0**	9.7	80.5	12.4	9.2
		0.9	**3.0**	9.6	72.8	12.4	9.2
Consensus-based Protocol		0.1	**1.9**	9.7	97.0	12.5	9.3
		0.5	**1.1**	9.7	73.2	12.5	9.1
		0.9	**0.4**	9.9	73.1	12.4	9.0
The Proposed Protocol	180	0.1	**3.0**	6.8	49.9	8.3	5.2
		0.5	**3.0**	6.8	58.0	8.3	5.1
		0.9	**3.0**	6.9	42.2	8.3	5.1
Priority-based Protocol		0.1	**3.0**	6.8	45.3	8.3	5.0
		0.5	**3.0**	6.8	44.3	8.3	5.1
		0.9	**3.0**	6.8	43.0	8.3	5.0
Consensus-based Protocol		0.1	**2.6**	6.8	58.1	8.3	5.0
		0.5	**3.0**	6.8	54.8	8.3	5.1
		0.9	**3.0**	6.9	47.5	8.3	5.1
The Proposed Protocol	240	0.1	**3.0**	5.5	31.2	6.2	2.9
		0.5	**3.0**	5.4	33.8	6.2	2.9
		0.9	**3.0**	5.5	33.7	6.2	3.0
Priority-based Protocol		0.1	**3.0**	5.5	41.3	6.3	3.0
		0.5	**3.0**	5.5	35.3	6.3	3.0
		0.9	**3.0**	5.5	40.2	6.3	3.0
Consensus-based Protocol		0.1	**3.0**	5.5	27.4	6.3	3.0
		0.5	**3.0**	5.5	27.7	6.2	2.9
		0.9	**3.0**	5.5	34.6	6.2	3.0

n: initial number of highway CAVs on the highway segment [−50km, 0km]; P: wireless packet loss rate; and $\Delta^* = 3$s.

is sampled every 0.4s). According to Table 1, for the proposed protocol, the time headways are always no less than 3.0s, which means the CTH-Δ^* safety (remember Δ^* is set to 3s, see Section 6.1) holds.[2] For the priority-based protocol, which basically is a subset of the proposed protocol, the CTH-Δ^* safety also holds. For the consensus-based protocol, the time headways cannot always satisfy CTH-Δ^* safety. Corresponding failures are highlighted in light gray in Table 1.

6.3 Liveness (Automatic Reset)

Theorem 1 **Claim 2** is on liveness guarantee, particularly in the sense of automatic reset. It proves the boundedness of reset time. This is confirmed by our simulations. According to Table 2, for the proposed protocol, all reset time costs are within the theoretical bound of $\Delta^{\max}_{\text{reset}} = 50.4$s.

Note for all protocols, for given n, as wireless packet loss rate P rises, more resets return to **Stable State 1** instead of **Stable State 2** (see Theorem 1-**Claim 2** for definitions). The former can happen as fast as a sub-second software reset (though not always); while the latter must involve physical movement, hence usually costs tens of seconds.

[2]Note our computer simulation's time granularity is 0.01s, hence our minimum time headway value is rounded to one digit after the floating point.

Table 2. Simulation Results: Reset Time Cost

Protocols	n	P	Reset time cost statistics (s)				
			min	median	max	average	std
The Proposed Protocol	120	0.1	0.1	0.1	**37.9**	5.6	11.8
		0.5	0.1	0.1	**37.7**	3.4	9.5
		0.9	0.1	0.1	**35.3**	0.7	3.8
Priority-based Protocol		0.1	0.1	0.1	29.2	2.9	8.5
		0.5	0.1	0.1	29.2	1.6	6.5
		0.9	0.1	0.1	29.2	0.3	2.2
Consensus-based Protocol		0.1	0.1	0.1	56.5	2.8	7.8
		0.5	0.1	0.1	45.4	1.9	6.3
		0.9	0.1	0.1	24.3	0.4	2.3
The Proposed Protocol	180	0.1	0.1	0.1	**37.6**	2.1	7.9
		0.5	0.1	0.1	**37.6**	1.6	6.9
		0.9	0.1	0.1	**33.8**	0.4	2.4
Priority-based Protocol		0.1	0.1	0.1	29.2	0.7	4.2
		0.5	0.1	0.1	29.2	0.5	3.5
		0.9	0.1	0.1	0.1	0.1	0
Consensus-based Protocol		0.1	0.1	0.1	30.8	0.4	2.7
		0.5	0.1	0.1	20.7	0.3	1.9
		0.9	0.1	0.1	22.8	0.2	1.2
The Proposed Protocol	240	0.1	0.1	0.1	**36.0**	0.4	3.2
		0.5	0.1	0.1	**34.9**	0.4	3.0
		0.9	0.1	0.1	**0.1**	0.1	0
Priority-based Protocol		0.1	0.1	0.1	29.2	0.2	1.5
		0.5	0.1	0.1	0.1	0.1	0
		0.9	0.1	0.1	0.1	0.1	0
Consensus-based Protocol		0.1	0.1	0.1	0.1	0.1	0
		0.5	0.1	0.1	0.1	0.1	0
		0.9	0.1	0.1	0.1	0.1	0

See Table 1 for definitions of n and P. Note according to Theorem 1-**Claim 2**, the reset time costs of the proposed protocol shall be upper bounded by $\Delta^{\max}_{\text{reset}} = 50.4$s.

Also, note that normally each simulation trial lasts 10 minutes (in the simulated universe). But in case by the end of the 10th minute, the system is still waiting for a reset to happen, the simulation will go on till the reset happens.

6.4 Merging Success Rate and Time Cost

Besides safety and liveness guarantees, we are also concerned about the merging success rates and time costs. Merging success means before the end of the simulation trial, the ramp CAV is merged into the highway lane, all vehicles on the highway lane reach speed of v_{lim}, *and* the CTH-Δ^* safety is maintained at all times. Merging time cost is the total time cost from the start of the merging scenario to the first time instance when merging success is achieved. For a simulation trial where merging success is never achieved, merging time cost is not applicable.

Table 3 shows the merging success rates and merging time cost statistics. According to the table, for any given n and P (referred to as "(n, P) *combination*" or simply "*combination*" in the following), we have 2 comparison pairs: the proposed protocol versus the priority-based protocol, and the proposed protocol versus the consensus-based protocol. Hence for all the 9 combinations of n and P (light, mild, and heavy traffic versus low, mild, and high wireless packet loss rates), we have $9 \times 2 = 18$ comparison pairs.

Table 3. Simulation Results: Merging Success Rate and Time Cost

Protocols	n	P	succ. rate	Merging time cost statistics (s)				
				min	median	max	avg	std
The Proposed Protocol	120	0.1	**24**/25	39.5	200.7	462.1	215.5	117.6
		0.5	**17**/25	42.9	235.5	516.5	282.9	134.8
		0.9	**3**/25	73.7	277.9	485.1	278.9	168.0
Priority-based Protocol		0.1	**19**/25	37.0	213.6	569.2	239.4	154.7
		0.5	**14**/25	48.6	325.1	535.6	300.0	164.9
		0.9	**2**/25	186.2	293.8	401.4	293.8	107.6
Consensus-based Protocol		0.1	**14**/25	23.6	199.0	548.8	219.2	138.8
		0.5	**7**/25	215.8	278.2	326.6	271.5	42.1
		0.9	**0**/25	n.a.	n.a.	n.a.	n.a.	n.a.
The Proposed Protocol	180	0.1	**14**/25	36.7	146.5	580.1	205.4	159.8
		0.5	**5**/25	48.2	374.6	479.8	282.3	185.8
		0.9	**1**/25	431.9	431.9	431.9	431.9	0
Priority-based Protocol		0.1	**7**/25	46.4	269.0	551.4	315.5	165.3
		0.5	**5**/25	81.8	352.6	561.2	310.8	182.8
		0.9	**0**/25	n.a.	n.a.	n.a.	n.a.	n.a.
Consensus-based Protocol		0.1	**3**/25	93.6	152.0	539.9	261.8	198.1
		0.5	**0**/25	n.a.	n.a.	n.a.	n.a.	n.a.
		0.9	**0**/25	n.a.	n.a.	n.a.	n.a.	n.a.
The Proposed Protocol	240	0.1	**3**/25	90.5	159.0	197.0	148.9	44.0
		0.5	**2**/25	140.0	352.4	564.8	352.4	212.4
		0.9	**0**/25	n.a.	n.a.	n.a.	n.a.	n.a.
Priority-based Protocol		0.1	**1**/25	155.6	155.6	155.6	155.6	0
		0.5	**0**/25	n.a.	n.a.	n.a.	n.a.	n.a.
		0.9	**0**/25	n.a.	n.a.	n.a.	n.a.	n.a.
Consensus-based Protocol		0.1	**0**/25	n.a.	n.a.	n.a.	n.a.	n.a.
		0.5	**0**/25	n.a.	n.a.	n.a.	n.a.	n.a.
		0.9	**0**/25	n.a.	n.a.	n.a.	n.a.	n.a.

See Table 1 for definitions of n and P; n.a.: not applicable.

Out of these 18 comparison pairs, there are 11 of them, where the proposed protocol's merging success rates are more than 99% better than the comparison counterpart's.[3]

This improvement is mainly because the proposed protocol focuses on two aspects simultaneously. It not only guarantees CTH-Δ^* safety under arbitrary wireless packet losses but also proactively coordinates the highway CAVs and the ramp CAV: when traffic is heavy, it asks the highway CAVs to yield to the ramp CAV. In comparison, neither of the other two protocols focuses on both of the aforementioned aspects.

More specifically, for all the 9 combinations of n and P, the consensus-based protocol fails all the 25 trials (i.e., success rate = 0) for 6 combinations; the priority-based protocol fails all the 25 trials (i.e., success rate = 0) for 3 combinations; while the proposed protocol only fails all the 25 trials (i.e., success rate = 0) for 1 combination, which corresponds to the heaviest traffic and highest wireless packet loss rate (i.e., $(n = 240, P = 0.9)$).

[3]In case the proposed protocol's success rate is positive, while the comparison counterpart's is 0, we also count the case as "the proposed protocol is more than 99% better".

Also, for (n, P) combinations where the consensus-based protocol succeeds for some trials (i.e., success rate > 0), the proposed protocol's merging time cost statistics are all comparable with (and usually better than) those of the consensus-based protocol's. Same is for the priority-based protocol.

7 CONCLUSION AND DISCUSSION

In this article, we propose a protocol to realize the safe merging of CAVs on highway and metered-ramp. We formally prove that the protocol can always guarantee the CTH safety and liveness, even under arbitrary wireless packet losses. These theoretical claims are verified by our simulations, which also show significant performance improvements over other alternatives.

This article also exemplifies the importance of introducing *cyber-physical transactions* and hybrid automata in the design and analysis of CPS. Particularly, Lemma 2 isolates possible combinations of discrete events and continuous maneuvres into mutually exclusive coop-durations (i.e., cyber-physical transactions). This greatly simplifies the design and analysis. Meanwhile, the specifications of the protocol and the formal proof of the CTH safety guarantee would be difficult (if not impossible) without the help of hybrid automata.

In future work, we will do the following.

(**FW1**) Take into consideration of wireless transmission and propagation delay. This delay shall be in the order of magnitude of 10 μs, which corresponds to distance errors in the order of magnitude of millimeters.

(**FW2**) Carry out *sensitivity study*: allow more variations in the various physical parameters, such as CAV velocities. As demonstrated by this article, the analyses are expected to be nontrivial and deserve multiple articles. Fortunately, the continuous nature of the physical world will bind the deviations from this article's formal models. This can help us to speculate the sensitivity. For example, suppose the CAV velocity error is bounded by V m/sec; as a coop-duration is bounded by $\Delta^{\max}_{\text{reset}}$, (suppose CTH is satisfied at the start of the coop-duration) then we can expect that the inter-CAV distance error from CTH safety is bounded by $2V\Delta^{\max}_{\text{reset}}$ when the coop-duration ends.

(**FW3**) Analyze the impacts when the number of packet losses is bounded. For example, whether there can be a time bound on the success of the merging.

(**FW4**) Derive necessary conditions for safety and liveness. We may start by negating the sufficient conditions listed in Theorem 1.

APPENDICES

A SYMBOL LIST

Symbols used in the article are listed alphabetically (Greek before Latin, and upper case before lower case) in the following.

(1) Δ_1, Δ_2, $\Delta^{\min}_{\text{BS}}$, Δ_{nonzeno}, Δ_r, Δ^* are all configuration constants with positive values, see Equations (4), (5), Section 4.2-"Base Station BS protocol behaviors"-(2) (5), Equation (2), Definition 1, respectively.
(2) $\Delta^{\max}_{\text{coop}}$, see Theorem 1 (**c2**).
(3) $\delta_{\text{a}}(v_1, v_2)$ is the total time needed to accelerate from v_1 to v_2 (where $0 \leqslant v_1 < v_2$), see Section 3.1.1.
(4) $\hat{\delta}_{\text{coop}}$ is a runtime variable local to A_{BS}. It is used to estimate the time distance of the current coop CAV to reach $\vec{p}_{\text{merge}}$.
(5) $\delta_{\text{d}}(v_2, v_1)$ is the total time needed to decelerate from v_2 to v_1 (where $v_2 > v_1 \geqslant 0$), see Section 3.1.2.

(6) δ_{defer} is a runtime variable created by A_{BS} at "Event2" (note Ineq. (7) ensures $\delta_{\text{defer}} > 0$), but may be sent to A_i ($i \in \{1, 2, \ldots, n\}$) in case h_i is chosen as the coop. It basically requests h_i to start deceleration (i.e., to yield) in δ_{defer} seconds.
(7) σ_{defer} is a runtime variable for A_r (see Figure 4). It is the data payload parameter received via the "Start" packet. In case the packet is sent by BS via "Event1" (see Figure 3), $\sigma_{\text{defer}} = 0$. In case the packet is sent by BS via the "GotAcceptSlowDown" event, $\sigma_{\text{defer}} = \delta_{\text{defer}}$. Upon reception of a "Start" packet, r will defer σ_{defer} seconds before actually starting the acceleration (i.e., entering the "AcceleratingOnRamp" mode of A_r).
(8) τ represents a runtime timer; it is a local variable to each hybrid automaton. Note for A_{BS}, the initial value of τ (when the system starts, i.e., at t_0) can be any value in $[0, \Delta_{\text{BS}}^{\text{min}}]$ (e.g., randomly chosen as per uniform distribution from this range); for A_r and A_i ($i = 1, 2, \ldots, n$), the initial value of τ is 0.
(9) D_1, D_r are both configuration constants with positive values, see Equation (8), Ineq. (1), respectively.
(10) acc is the predefined acceleration strategy, see Section 3.1.1.
(11) coop is a runtime variable for hybrid automaton A_{BS} only, whose value can only be "undefined" or h_i ($i \in \{1, 2, \ldots, n\}$). Intuitively, it refers to the closest approaching highway CAV toward the base station BS.
(12) $d_{\text{a}}(v_1, v_2)$ is the total distance needed to accelerate from v_1 to v_2 (where $0 \leqslant v_1 < v_2$), see Section 3.1.1.
(13) $d_{\text{d}}(v_2, v_1)$ is the total distance needed to decelerate from v_2 to v_1 (where $v_2 > v_1 \geqslant 0$), see Section 3.1.2.
(14) dec is the predefined deceleration strategy, see Section 3.1.2.
(15) $\vec{p}(x, t)$ is the location of vehicle x at wall clock time t. Correspondingly, $|\dot{\vec{p}}(x, t)|$ is the speed of vehicle x at t, and $\ddot{\vec{p}}(x, t)$ is the acceleration (deceleration) of x at t.
(16) state is a runtime variable for A_i ($i = 1, 2, \ldots, n$) only, whose value can only be "Init", "Coop", or "Sync". Note the initial value of state is set to "Init".
(17) t is the current wall clock time; it is a global variable.
(18) v_{lim} and v_{rm} are configuration constants related to CAV speed. They are, respectively, the maximum and minimum allowed speed on the highway lane. See Section 3.2 and Ineq. (3) for more information.

B PROOF OF LEMMA 4

Proof: First, as $(t_1, t_2]$ is the first ever coop-duration, due to Assumption 5 and Lemma 3, h_1, $h_2, \ldots, h_n$ all reside in hybrid automata mode "Init" throughout $[t_0, t_1]$, hence $\forall t \in [t_0, t_1]$, (h_i, h_{i+1}) ($i = 1, 2, \ldots, n-1$) is CTH-Δ^* safe. ($\star$)

Suppose the coop-duration $(t_1, t_2]$ belongs to h_ι ($\iota \in \{1, 2, \ldots, n\}$). Then we have the following cases.

Case 1: First we discuss h_j, where $j < \iota$. As coop-durations cannot overlap nor connect, $\forall j \in \{1, 2, \ldots, \iota - 1\}$, as per A_j (see Figure 5), throughout $(t_1, t_2]$, h_j must remain in mode "Init". That is, for any h_j and h_{j+1} ($j \in \{1, 2, \ldots, \iota - 2\}$), throughout $(t_1, t_2]$, both retain the speed of v_{lim}. hence (h_j, h_{j+1}) is CTH-Δ^* safe throughout $(t_1, t_2]$. For $h_{\iota-1}$ and h_ι, as $h_{\iota-1}$ retains the maximum allowed speed, v_{lim}, throughout $(t_1, t_2]$, hence $(h_{\iota-1}, h_\iota)$ is CTH-Δ^* safe throughout $(t_1, t_2]$.

Case 2: Now we discuss h_j, where $j > \iota$.

Case 2.1: Suppose at t_1, h_k ($k > \iota$) is the first highway CAV after h_ι s.t. $|\vec{p}(h_{k-1}, t_1) - \vec{p}(h_k, t_1)| > D_1$. Then we have the following cases.

Case 2.1.1: $\forall j \in \{\iota + 1, \iota + 2, \ldots, k - 1\}$, we have (h_{j-1}, h_j) is CTH-Δ^* safe throughout $(t_1, t_2]$. ($\star\star$)

This can be proved iteratively.

For $h_{\iota+1}$, throughout $(t_1, t_1 + \delta_{\text{defer}}]$, both h_ι and $h_{\iota+1}$ remain at v_{lim}; throughout $(t_1 + \delta_{\text{defer}}, t_2]$, $h_{\iota+1}$ synchronizes its speed with h_ι according to mode "Sync" (see Figure 5); Hence throughout $(t_1, t_2]$, $(h_\iota, h_{\iota+1})$ remains CTH-Δ^* safe.

Same reasoning can be applied to $h_{\iota+2}, h_{\iota+3}, \ldots, h_{k-1}$. Hence ($\star\star$) sustains.

Case 2.1.2: For h_k, we have (h_{k-1}, h_k) is CTH-Δ^* safe throughout $(t_1, t_2]$. ($\star\star\star$)

We prove this step by step.

(i) $\forall t \in (t_1, t_1 + \delta_{\text{defer}}]$, both h_{k-1} and h_k retain the speed of v_{lim}, hence h_k remains in "Init" and $|\vec{p}(h_{k-1}, t) - \vec{p}(h_k, t)|$ remains unchanged.

(ii) $\forall t \in (t_1 + \delta_{\text{defer}}, t_1 + \delta_{\text{defer}} + \delta_{\text{d}}(v_{\text{lim}}, v_{\text{rm}})]$, h_{k-1} synchronizes its speed with $h_{k-2}, h_{k-3}, \ldots, h_\iota$, hence keeps decelerating from v_{lim} to v_{rm}; while h_k remains in "Init" (as $|\vec{p}(h_{k-1}, t_1 + \delta_{\text{defer}}) - \vec{p}(h_k, t_1 + \delta_{\text{defer}})| > D_1$, event "StartSyncPred" will not happen at $t_1 + \delta_{\text{defer}}$ to h_k, and during $(t_1 + \delta_{\text{defer}}, t_1 + \delta_{\text{defer}} + \delta_{\text{d}}(v_{\text{lim}}, v_{\text{rm}})]$ the event will neither happen to h_k as h_{k-1}'s speed is below v_{lim}). Meanwhile, for the entire deceleration process, $|\vec{p}(h_{k-1}, t) - \vec{p}(h_k, t)| > D_1 + d_{\text{d}}(v_{\text{lim}}, v_{\text{rm}}) - v_{\text{lim}}\delta_{\text{d}}(v_{\text{lim}}, v_{\text{rm}}) > v_{\text{lim}}\Delta^*$ (see D_1's definition in Equation (8)). This means (h_{k-1}, h_k) is CTH-Δ^* safe at t.

(iii) $\forall t \in (t_1 + \delta_{\text{defer}} + \delta_{\text{d}}(v_{\text{lim}}, v_{\text{rm}}), t_1 + \delta_{\text{defer}} + \Delta_r + \Delta^*]$ (note according to (6), $\delta_{\text{d}}(v_{\text{lim}}, v_{\text{rm}}) < \Delta_r + \Delta^*$), h_{k-1} remains synchronizing its speed with $h_{k-2}, h_{k-3}, \ldots, h_\iota$, hence keeps the speed of v_{rm}, while h_k remains in mode "Init" (as $|\dot{\vec{p}}(h_{k-1}, t)| < v_{\text{lim}}$, event "StartSyncPred" will not happen to h_k). Meanwhile for this entire constant speed process, $|\vec{p}(h_{k-1}, t) - \vec{p}(h_k, t)| > D_1 + d_{\text{d}}(v_{\text{lim}}, v_{\text{rm}}) - v_{\text{lim}}\delta_{\text{d}}(v_{\text{lim}}, v_{\text{rm}}) - (v_{\text{lim}} - v_{\text{rm}})(\Delta_r + \Delta^* - \delta_{\text{d}}(v_{\text{lim}}, v_{\text{rm}})) > v_{\text{lim}}\Delta^*$ (see D_1's definition in Equation (8)). This means (h_{k-1}, h_k) is CTH-Δ^* safe at t.

(iv) $\forall t \in (t_1 + \delta_{\text{defer}} + \Delta_r + \Delta^*, t_1 + \delta_{\text{defer}} + \Delta_r + \Delta^* + \delta_{\text{a}}(v_{\text{rm}}, v_{\text{lim}})]$ (note $t_1 + \delta_{\text{defer}} + \Delta_r + \Delta^* + \delta_{\text{a}}(v_{\text{rm}}, v_{\text{lim}})$ is when the coop-duration ends, i.e., it equals to t_2), h_{k-1} remains synchronizing its speed with $h_{k-2}, h_{k-3}, \ldots, h_\iota$, hence keeps accelerating from v_{rm} to v_{lim}; while h_k remains in "Init" (as h_{k-1} is accelerating, event "StartSyncPred" will not happen to h_k). Meanwhile for this entire acceleration process, $|\vec{p}(h_{k-1}, t) - \vec{p}(h_k, t)| > D_1 + d_{\text{d}}(v_{\text{lim}}, v_{\text{rm}}) - v_{\text{lim}}\delta_{\text{d}}(v_{\text{lim}}, v_{\text{rm}}) - (v_{\text{lim}} - v_{\text{rm}})(\Delta_r + \Delta^* - \delta_{\text{d}}(v_{\text{lim}}, v_{\text{rm}})) - (v_{\text{lim}}\delta_{\text{a}}(v_{\text{rm}}, v_{\text{lim}}) - d_{\text{a}}(v_{\text{rm}}, v_{\text{lim}})) \geqslant v_{\text{lim}}\Delta^*$ (see Δ_1 and D_1's definition in Equations (4) and (8)). This means (h_{k-1}, h_k) is CTH-Δ^* safe at t.

Combining (i)~(iv), we see ($\star\star\star$) sustains.

Case 2.1.3: For h_j ($j = k+1, k+2, \ldots, n$), as h_k remains in mode "Init" throughout $(t_1, t_2]$, h_{k+1} remains in mode "Init" throughout $(t_1, t_2]$, so on and so forth.

Combining **Case 2.1.1**~**Case 2.1.3**, we see in **Case 2.1**, $\forall j \in \{\iota+1, \iota+2, \ldots, n\}$, $\forall t \in (t_1, t_2]$, (h_{j-1}, h_j) is CTH-Δ^* safe at t.

Case 2.2 Suppose at t_1, $\forall j \in \{\iota+1, \iota+2, \ldots, n\}$, $|\vec{p}(h_{j-1}, t_1) - \vec{p}(h_j, t_1)| \leqslant D_1$, then follow the same proving method for **Case 2.1.1**, we can prove $\forall t \in (t_1, t_2]$, (h_{j-1}, h_j) is CTH-Δ^* safe at t.

Combining **Case 2.1** and **Case 2.2**, we see in **Case 2**, $\forall j \in \{\iota+1, \iota+2, \ldots, n\}$, $\forall t \in (t_1, t_2]$, (h_{j-1}, h_j) is CTH-Δ^* safe at t.

Combining **Case 1** and **Case 2**, together with the claim ($\star$) proven at the very beginning, the lemma sustains. □

C PROOF OF LEMMA 5

Proof: **Case 1**: If coop-duration never happens, then all highway CAVs always remain in hybrid automata mode "Init". The lemma trivially sustain.

Case 2: If infinite coop-duration(s) happen. Suppose $(t_1, t_2] \subseteq [t_0, +\infty)$ is the first coop-duration ever happens. Then due to Lemma 4, this lemma trivially sustains for the duration $[t_0, t_2]$. At t_2^+, due to Lemma 2 and Lemma 3, all highway CAVs have returned to mode "Init", and $\forall i \in \{1, 2, \ldots, n-1\}$, (h_i, h_{i+1}) is CTH-Δ^* safe at t_2. Regard t_2 as the new t_0, and apply the same technique

to prove Lemma 4, we can prove this lemma sustains to the end of the second coop-duration, so on and so forth, until we cover time instance t. The lemma shall sustain.

Case 3: If finite coop-duration(s) happen. Then we can apply the proving technique of **Case 2**, and (if needed) after the last coop-duration ends, we can apply the proving technique of **Case 1**, until we cover time instance t. The lemma shall sustain.

Combining **Case 1** to **Case 3**, the lemma sustains. □

D PROOF OF LEMMA 6

Proof: According to A_r (see Figure 4), if r reaches $\vec{p}_{\text{merge}}$ at t_1, then it must have received the "ActualStart" event at $t_2 \stackrel{\text{def}}{=} t_1 - \Delta_r$, which is caused by a "Start" packet from the BS. There can be two cases.

Case 1: The "Start" packet is sent by BS via "Event1" in A_{BS} (see Figure 3) at t_2.

Then first, this means the most recent "Event2" of A_{BS}, the only event that can trigger a coop-duration, (if it ever happened) must be before $t_2 - \Delta_{\text{BS}}^{\min}$ (note there can be no more "Event2" of A_{BS} after t_2, as r has received "Start"). Due to (**c2**), $\Delta_{\text{BS}}^{\min} > \Delta_{\text{coop}}^{\max} > \Delta_r$, there is no coop-duration overlapping or connecting with $[t_2, +\infty)$. Due to Lemma 3, all highway CAVs hence should remain in "Init" throughout $[t_2, +\infty)$. (†)

Second, the "Event1" of A_{BS} at t_2 could be due to two cases at t_2^-, when BS receives a ("MergeReq", r, BS) packet.

Case 1.1: At t_2^-, there is no CAV on the highway lane segment of $(-\infty, \vec{p}_{\text{merge}}]$. This means at t_2^-, h_n is at highway lane segment of $(\vec{p}_{\text{merge}}, +\infty)$. So by t_1, h_n is at least $v_{\text{rm}}\Delta_r \geqslant v_{\text{lim}}\Delta^*$ (due to (**c3**)) ahead of r. Due to Corollary 1, this implies all highway CAVs are at least $v_{\text{rm}}\Delta_r \geqslant v_{\text{lim}}\Delta^*$ ahead of r at t_1. (††)

Conclusion (†) and (††) imply that $\forall t \in [t_1, +\infty)$, (h_i, r) $(i = 1, 2, \ldots, n)$ is CTH-Δ^* safe at t.

Case 1.2: At t_2^-, there is/are highway CAVs on the highway lane segment $(-\infty, \vec{p}_{\text{merge}}]$. Suppose the one closest to $\vec{p}_{\text{merge}}$ is h_i $(i \in \{1, 2, \ldots, n\})$. Then because BS sends "Start" packet via "Event1", we know

$$\hat{\delta}_{\text{coop}} = |\vec{p}_{\text{merge}} - \vec{p}(h_i, t_2^-)|/v_{\text{lim}} \geqslant \Delta_r + \Delta^* + \Delta_1 \tag{10}$$

Meanwhile, as per A_r, r shall reach $\vec{p}_{\text{critical}}$ (the location where r first reaches speed v_{lim}, see Figure 1) at $t_3 \stackrel{\text{def}}{=} t_1 + \delta_{\text{a}}(v_{\text{rm}}, v_{\text{lim}})$, and $|\vec{p}_{\text{critical}} - \vec{p}_{\text{merge}}| = d_{\text{a}}(v_{\text{rm}}, v_{\text{lim}})$.

Due to (†), h_i reaches $\vec{p}_{\text{merge}}$ at $t_2 + \hat{\delta}_{\text{coop}} \geqslant t_2 + \Delta_r + \Delta^* + \Delta_1$ (due to (10)) $= t_1 + \Delta^* + \Delta_1$. This means (r, h_i) is CTH-Δ^* safe at t_1.

Furthermore, h_i reaches $\vec{p}_{\text{critical}}$ at $t_2 + \hat{\delta}_{\text{coop}} + d_{\text{a}}(v_{\text{rm}}, v_{\text{lim}})/v_{\text{lim}} \geqslant t_2 + \Delta_r + \Delta^* + \Delta_1 + d_{\text{a}}(v_{\text{rm}}, v_{\text{lim}})/v_{\text{lim}} = t_1 + \Delta^* + \delta_{\text{a}}(v_{\text{rm}}, v_{\text{lim}})$ (see the definition of Δ_1 in (4)) $= t_3 + \Delta^*$. Hence (r, h_i) is CTH-Δ^* safe at t_3.

As r reaches v_{lim} after t_3, we hence conclude (r, h_i) is CTH-Δ^* safe throughout $[t_1, +\infty)$.

Furthermore, due to Lemma 5 and Corollary 1, we can conclude $\forall j \in \{i, i+1, \ldots, n\}$, (r, h_j) is CTH-Δ^* safe throughout $[t_1, +\infty)$.

Another important CAV is h_{i-1}. As it is on segment $(\vec{p}_{\text{merge}}, +\infty)$ at t_2^-, using the same reasoning for **Case 1.1**, we know (h_{i-1}, r) is CTH-Δ^* safe throughout $[t_1, +\infty)$.

Due to Corollary 1, we can conclude $\forall j \in \{1, 2, \ldots, i-1\}$, (h_j, r) is CTH-Δ^* safe throughout $[t_1, +\infty)$.

Case 2: The "Start" packet is sent by BS via "Event2" (followed by "GotAcceptSlowDown") in A_{BS} (see Figure 3) at $t_2 - \delta_{\text{defer}}$. Immediately before it, BS must have sent ("SlowDown", BS, h_{coop}, δ_{defer}) packet to h_{coop} at $t_2 - \delta_{\text{defer}}$ and received h_{coop}'s "AcceptSlowDown" packet, where coop $\in \{1, 2, \ldots, n\}$. Without loss of generality, suppose coop $= i$.

Then during $[t_2 - \delta_{\text{defer}}, t_2]$, h_i remains at v_{lim} and drives $v_{\text{lim}}\delta_{\text{defer}} = \hat{\delta}_{\text{coop}}v_{\text{lim}} - d_{\text{d}}(v_{\text{lim}}, v_{\text{rm}}) - v_{\text{rm}}(\Delta_r + \Delta^* - \delta_{\text{d}}(v_{\text{lim}}, v_{\text{rm}}))$ distance since $t_2 - \delta_{\text{defer}}$.

During $(t_2, t_2 + \delta_{\text{d}}(v_{\text{lim}}, v_{\text{rm}})]$, h_i decelerates from v_{lim} to v_{rm} (note due to (6), $t_2 + \delta_{\text{d}}(v_{\text{lim}}, v_{\text{rm}}) < t_2 + \Delta_r = t_1$) and drives $v_{\text{lim}}\delta_{\text{defer}} + d_{\text{d}}(v_{\text{lim}}, v_{\text{rm}}) = \hat{\delta}_{\text{coop}}v_{\text{lim}} - v_{\text{rm}}(\Delta_r + \Delta^* - \delta_{\text{d}}(v_{\text{lim}}, v_{\text{rm}}))$ distance since $t_2 - \delta_{\text{defer}}$.

During $(t_2+\delta_{\text{d}}(v_{\text{lim}}, v_{\text{rm}}), t_2+\Delta_r+\Delta^*]$, h_i remains at v_{rm}. Note $t_1 = t_2+\Delta_r \in (t_2+\delta_{\text{d}}(v_{\text{lim}}, v_{\text{rm}}), t_2+\Delta_r + \Delta^*)$. This means, at t_1, h_i is in the "ConstLowSpeed" mode, maintaining the speed of v_{rm}. Therefore, at t_1, $\vec{p}(r, t_1) - \vec{p}(h_i, t_1) = \vec{p}_{\text{merge}} - \vec{p}(h_i, t_1) = \hat{\delta}_{\text{coop}}v_{\text{lim}} - (\hat{\delta}_{\text{coop}}v_{\text{lim}} - v_{\text{rm}}(\Delta_r + \Delta^* - \delta_{\text{d}}(v_{\text{lim}}, v_{\text{rm}})) + v_{\text{rm}}(t_1 - t_2 - \delta_{\text{d}}(v_{\text{lim}}, v_{\text{rm}}))) = v_{\text{rm}}\Delta^* > 0$. This means, at t_1, r is ahead of h_i by $v_{\text{rm}}\Delta^*$; and as h_i's speed at t_1 is v_{rm}, the above means (r, h_i) is CTH-Δ^* safe at t_1.

After t_1, r accelerates from v_{rm} to v_{lim}, while h_i remains at v_{rm} till $t_1 + \Delta^*$, when it reaches $\vec{p}_{\text{merge}}$. Then h_i carry out the same acceleration process as that of r to reach v_{lim}. Therefore, the two time-location curves (time as the x-axis, and location as the y-axis) of r and h_i above the location of $\vec{p}_{\text{merge}}$ are parallel and Δ^* away shifted along the time axis. Note the acceleration process is monotonic (the speed keeps monotonically increasing until the target speed is reached, see Assumption 1), and finally both CAVs stabilize at v_{lim}. By observing the time-location curves, we can see that during $[t_1, t_1 + \Delta^*]$, (r, h_i) is CTH-Δ^* safe; and during $[t_1 + \Delta^*, +\infty)$, (r, h_i) is also CTH-Δ^* safe. So in summary, (r, h_i) is CTH-Δ^* safe throughout $[t_1, +\infty)$.

Furthermore, due to Lemma 5 and Corollary 1, we can conclude $\forall j \in \{i, i+1, \ldots, n\}$, (r, h_j) is CTH-Δ^* safe throughout $[t_1, +\infty)$.

Another important CAV is h_{i-1} (if $i > 1$). At $t_2 - \delta_{\text{defer}}$, when BS sends "SlowDown" packet to h_i, h_{i-1} must be on segment $(\vec{p}_{\text{merge}}, +\infty)$. Also, notice as coop-durations cannot overlap nor connect, and BS sends no more "SlowDown" packet after $t_2 - \delta_{\text{defer}}$. This means throughout $[t_2 - \delta_{\text{defer}}, +\infty)$, h_{i-1} is in "Init". Then using the same reasoning for **Case 1.1** for h_n, we can prove (h_{i-1}, r) is CTH-Δ^* safe throughout $[t_1, +\infty)$.

Due to Corollary 1, we conclude $\forall j \in \{1, \ldots, i-1\}$, (h_j, r) is CTH-Δ^* safe throughout $[t_1, +\infty)$.

Combining **Case 1** and **Case 2**, we conclude the lemma sustains. □

ACKNOWLEDGMENTS

We also thank Dr. Shaheer Muhammad, Mr. Shiyu Zhang, and Mr. Zhihao Zhao anonymous reviewers for their comments to improve this article.

REFERENCES

[1] Federal Highway Administration. 2014. *Ramp Metering: A Proven, Cost-Effective Operational Strategy - A Primer*. US Department of Transportation.

[2] Michael Aeberhard, SebasEan Rauch, Mohammad Bahram, Georg Tanzmeister, Julian Thomas, Yves Pilat, Florian Homm, Werner Huber, and Nico Kaempchen. 2015. Experience, results and lessons learned from automated driving on Germany's highways. *IEEE Intelligent Transportation Systems Magazine* 7, 1 (2015), 42–57.

[3] Md Salman Ahmed, Mohammad A. Hoque, Jackeline Rios-Torres, and Asad Khauak. 2018. A cooperative freeway merge assistance system using connected vehicles. arXiv:1805.00508. Retrieved from https://arxiv.org/abs/1805.00508

[4] Rajeev Alur, Costas Courcoubetis, Thomas A. Henzinger, and Pei-Hsin Ho. 1992. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. *Hybrid Systems* LNCS v736 (1992), 209–229.

[5] Shunsuke Aoki and Ragunathan Rajkumar. 2017. A merging protocol for self-driving vehicles. In *Proceedings of the 2017 ACM/IEEE 8th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 219–228.

[6] Tanveer Awal, Lars Kulik, and Kotagiri Ramamohanrao. 2013. Optimal traffic merging strategy for communication- and sensor-enabled vehicles. In *Proceedings of the 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. IEEE, 1468–1474.

[7] Jakob Axelsson. 2017. Safety in vehicle platooning: A systematic literature review. *IEEE Transactions on Intelligent Transportation Systems* 18, 5 (2017), 1033–1045.

[8] Mario di Bernardo, Alessandro Salvi, and Stefania SanEni. 2014. Distributed consensus strategy for platooning of vehicles in the presence of time-varying heterogeneous communication delays. *IEEE Transactions on Intelligent Transportation Systems* 16, 1 (2014), 102–112.

[9] Wenjing Cao, Masakazu Mukai, Taketoshi Kawabe, Hikaru Nishira, and Noriaki Fujiki. 2015. Cooperative vehicle path generation during merging using model predictive control with real-time optimization. *Control Engineering Practice* 34, January (2015), 98–105.

[10] David J. Chang and Edward K. Moriok. 2005. Vehicle speed profiles to minimize work and fuel consumption. *Journal of Transportation Engineering* 131, 3 (2005), 173–182.

[11] Hossein Chehardoli and Ali Ghasemi. 2018. Adaptive centralized/decentralized control and identification of 1-D heterogeneous vehicular platoons based on constant time headway policy. *IEEE Transactions on Intelligent Transportation Systems* 19, 10 (2018), 3376–3386.

[12] Tianyi Chen, Meng Wang, Siyuan Gong, Yang Zhou, and Bin Ran. 2021. Connected and automated vehicle distributed control for on-ramp merging scenario: A virtual rotation approach. arXiv:2103.15047. Retrieved from https://arxiv.org/abs/2103.15047

[13] Laurene Claussmann, Marc Revilloud, Dominique Gruyer, and Sebastien Glaser. 2020. A review of motion planning for highway autonomous driving. *IEEE Transactions on Intelligent Transportation Systems* 21, 5 (2020), 1826–1848.

[14] Kakan C. Dey, Li Yan, Xujie Wang, Yue Wang, Haiying Shen, Mashrur Chowdhury, Lei Yu, Chenxi Qiu, and Vivekgautham Soundararaj. 2016. A review of communication, driver characteristics, and controls aspects of cooperative adaptive cruise control (CACC). *IEEE Transactions on Intelligent Transportation Systems* 17, 2 (2016), 491–509.

[15] Jianbang Du, Qing Li, Fengxiang Qiao, and Lei Yu. 2018. Estimation of vehicle emission on mainline freeway under isolated and integrated ramp metering strategies. *Environmental Engineering and Management Journal* 17, 5 (2018), 1237–1248.

[16] Xueli Fan. 2022. *Cooperative Driving for Connected and Automated Vehicles on Dedicated Lanes*. Ph. D. Dissertation. The Hong Kong Polytechnic University.

[17] Xueli Fan, Qixin Wang, and Jie Liu. 2020. Work-in-progress abstract: A reliable wireless smart vehicle highway on-ramp merging protocol with constant time headway safety guarantee. In *Proceedings of the 2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS) WiP Session*. 184–185.

[18] Cary Gray and David Cheriton. 1989. Leases: An efficient fault-tolerant mechanism for distributed file cache consistency. In *Proceedings of the 20th ACM Symposium on Operating Systems Principles*. 202–210.

[19] Scou Heim, Rory Alston, Daniel Pierce, Patrick Smith, Robert Lyle, and David Bevly. 2019. *Automated Platoon Manipulation in Merging Scenarios Using Trajectory Estimation of Connected Vehicles*. Technical Report. US ARMY CCDC GVSC (FORMERLY TARDEC) WARREN United States.

[20] Peter Hidas. 2005. Modelling vehicle interactions in microscopic simulation of merging and weaving. *Transportation Research Part C: Emerging Technologies* 13, 1 (2005), 37–62.

[21] Xiaowen Jiang, Peter J. Jin, Xia Wan, and Yizhou Wang. 2017. A V2I (vehicle-to-infrastructure) based dynamic merge assistance method based on instantaneous virtual trajectories: A microscopic implementation of gap metering. In *Proceedings of the Transportation Research Board 96th Annual Meeting*.

[22] Igor Kabashkin. 2017. Reliable v2x communications for safety-critical intelligent transport systems. In *Proceedings of the 2017 Advances in Wireless and Optical Communications (RTUWO)*. IEEE, 251–255.

[23] Elham Semsar Kazerooni and Jeroen Ploeg. 2015. Interaction protocols for cooperative merging and lane reduction scenarios. In *Proceedings of the 2015 IEEE 18th International Conference on Intelligent Transportation Systems*. IEEE, 1964–1970.

[24] Stefan R. Klomp, Victor L. Knoop, Henk Taale, and Serge P. Hoogendoorn. 2021. Ramp metering with microscopic gap detection algorithm design and empirical acceleration verification. *Transportation Research Record* 2676, 1 (2021), 91–104.

[25] Stephanie Lefevre, Chao Sun, Ruzena Bajcsy, and Christian Laugier. 2014. Comparison of parametric and non-parametric approaches for vehicle speed prediction. In *Proceedings of the American Control Conference (ACC)*. 3494–3499.

[26] Clark Letter and Lily Elefteriadou. 2017. Efficient control of fully automated connected vehicles at freeway merge segments. *Transportation Research Part C: Emerging Technologies* 80, July (2017), 190–205.

[27] Xiao-Yun Lu and J. Karl Hedrick. 2003. Longitudinal control algorithm for automated vehicle merging. *International Journal of Control* 76, 2 (2003), 193–202.

[28] Xiao-Yun Lu, Han-Shue Tan, Steven E. Shladover, and J. Karl Hedrick. 2004. Automated vehicle merging maneuver implementation for AHS. *Vehicle System Dynamics* 41, 2 (2004), 85–107.

[29] Haigen Min, Yukun Fang, Xia Wu, Guoyuan Wu, and Xiangmo Zhao. 2021. On-ramp merging strategy for connected and automated vehicles based on complete information static game. *Journal of Traffic and Transportation Engineering (English Edition)* 8, 4 (2021), 582–595.

[30] Fernando V. Monteiro and Petros Ioannou. 2021. Safe lane change and merging gaps in connected environments. *IFAC-PapersOnLine* 54, 2 (2021), 69–74.
[31] América Morales and Henk Nijmeijer. 2016. Merging strategy for vehicles by applying cooperative tracking control. *IEEE Transactions on Intelligent Transportation Systems* 17, 12 (2016), 3423–3433.
[32] Ioannis A. Ntousakis, Kallirroi Porfyri, Ioannis K. Nikolos, and Markos Papageorgiou. 2014. Assessing the impact of a cooperative merging system on highway traffic using a microscopic flow simulator. In *Proceedings of the ASME 2014 International Mechanical Engineering Congress and Exposition*. American Society of Mechanical Engineers, V012T15A024–V012T15A024.
[33] Rattaphol Pueboobpaphan, Fei Liu, and Bart van Arem. 2010. The impacts of a communication based merging assistant on traffic flows of manual and equipped vehicles at an on-ramp using traffic flow simulation. In *Proceedings of the 13th International IEEE Conference on Intelligent Transportation Systems*. IEEE, 1468–1473.
[34] Rajesh Rajamani. 2012. *Vehicle Dynamics and Control*. Springer.
[35] Gurulingesh Raravi, Vipul Shingde, Krithi Ramamritham, and JaEn Bharadia. 2007. Merge algorithms for intelligent vehicles. In *Proceedings of the Next Generation Design and Verification Methodologies for Distributed Embedded Control Systems*. Springer, 51–65.
[36] Marwan K. Shalaby, Amr Farag, Omar M. AbdelAziz, Dalia M. Mahfouz, Omar M. Shehata, and Elsayed I. Morgan. 2019. Design of various dynamical-based trajectory tracking control strategies for multi-vehicle platooning problem. In *Proceedings of the IEEE Intelligent Transportation Systems Conference (ITSC)*. 1631–1637.
[37] Darbha Swaroop and K. R. Rajagopal. 2001. A review of constant time headway policy for automatic vehicle following. In *Proceedings of the 2001 IEEE Intelligent Transportation Systems.* IEEE, 65–69.
[38] Feng Tan, Yufei Wang, Qixin Wang, Lei Bu, and Neeraj Suri. 2015. A lease based hybrid design pattern for proper-temporal-embedding of wireless cps interlocking. *IEEE Transactions on Parallel and Distributed System* 26, 10 (2015), 2630–2642.
[39] Christophe Viel, Ulysse Vautier, Jian Wan, and Luc Jaulin. 2019. Platooning control for heterogeneous sailboats based on constant time headway. *IEEE Transactions on Intelligent Transportation Systems* 21, 5 (2019), 2078–2089.
[40] Meng Wang. 2018. Infrastructure assisted adaptive driving to stabilise heterogeneous vehicle strings. *Transportation Research Part C* 91, June (2018), 276–295.
[41] Yibing Wang, Elias B. Kosmatopoulos, Markos Papageorgiou, and Ioannis Papamichail. 2014. Local ramp metering in the presence of a distant downstream bottleneck: Theoretical analysis and simulation study. *IEEE Transactions on Intelligent Transportation Systems* 15, 5 (2014), 2024–2039.
[42] Yunpeng Wang, E. Wenjuan, Wenzhong Tang, Daxin Tian, Guangquan Lu, and Guizhen Yu. 2013. Automated on-ramp merging control algorithm based on internet-connected vehicles. *IET Intelligent Transport Systems* 7, 4 (2013), 371–379.
[43] Ziran Wang, Guoyuan Wu, and Matthew Barth. 2018. *Distributed Consensus-Based Cooperative Highway On-Ramp Merging Using V2X Communications*. Technical Report. SAE Technical Paper.
[44] Yuanchang Xie, Huixing Zhang, Nathan H. Gartner, and Tugba Arsava. 2017. Collaborative merging strategy for freeway ramp operations in a connected and autonomous vehicles environment. *Journal of Intelligent Transportation Systems* 21, 2 (2017), 136–147.
[45] Guangchuan Yang and Zong Tian. 2019. An exploratory investigation of the impact of ramp metering on driver acceleration behavior. *IATSS Research* 42, 4 (2019), 277–285.
[46] Masashi Yoshida, Hiromu Asahina, Hiroshi Shigeno, and Iwao Sasase. 2020. A scheduling scheme for cooperative merging at a highway on-ramp with maximizing average speed of automated vehicles. In *Proceedings of the 2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall)*. IEEE, 1–5.
[47] Xiaohai Yu, Ge Guo, and Hongbo Lei. 2018. Longitudinal cooperative control for a bidirectional platoon of vehicles with constant time headway policy. In *Proceedings of the 2018 Chinese Control And Decision Conference (CCDC)*. IEEE, 2427–2432.

Received 4 October 2021; revised 13 May 2023; accepted 19 June 2023

Message Authentication and Provenance Verification for Industrial Control Systems

ERTEM ESINER, UTKU TEFEK, and DAISUKE MASHIMA, Advanced Digital Sciences Center, Singapore
BINBIN CHEN, Singapore University of Technology and Design, Singapore
ZBIGNIEW KALBARCZYK and DAVID M. NICOL, University of Illinois Urbana Champaign, USA

Successful attacks against industrial control systems (ICSs) often exploit insufficient checking mechanisms. While firewalls, intrusion detection systems, and similar appliances introduce essential checks, their efficacy depends on the attackers' ability to bypass such middleboxes. We propose a provenance solution to enable the verification of an end-to-end message delivery path and the actions performed on a message. Fast and flexible provenance verification (F2-Pro) provides cryptographically verifiable evidence that a message has originated from a legitimate source and gone through the necessary checks before reaching its destination. F2-Pro relies on lightweight cryptographic primitives and flexibly supports various communication settings and protocols encountered in ICS thanks to its transparent, bump-in-the-wire design. We provide formal definitions and cryptographically prove F2-Pro's security. For human interaction with ICS via a field service device, F2-Pro features a multi-factor authentication mechanism that starts the provenance chain from a human user issuing commands. We compatibility tested F2-Pro on a smart power grid testbed and reported a sub-millisecond latency overhead per communication hop using a modest ARM Cortex-A15 processor.

CCS Concepts: • **Security and privacy** → **Hash functions and message authentication codes**; **Authentication**; • **Hardware** → *Smart grid*; • **Computer systems organization** → *Embedded software*; • **Networks** → **Security protocols;**

Additional Key Words and Phrases: Message authentication, verifiable, provenance, digital signature, hash chain, signature aggregation, cyberphysical system, Bump-in-the-wire, multi-factor authentication

ACM Reference format:
Ertem Esiner, Utku Tefek, Daisuke Mashima, Binbin Chen, Zbigniew Kalbarczyk, and David M. Nicol. 2023. Message Authentication and Provenance Verification for Industrial Control Systems. *ACM Trans. Cyber-Phys. Syst.* 7, 4, Article 24 (October 2023), 28 pages.
https://doi.org/10.1145/3607194

An earlier version of this article was presented at the IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm) in 2019 [1]. This article extends [1] with comprehensive system and threat models, formal definitions and cryptographic security proof (and the corresponding protocol improvements), a multi-factor authentication solution for remote access to ICS, and discussions on practical security implications.
Authors' addresses: E. Esiner, U. Tefek, and D. Mashima, Advanced Digital Sciences Center, Singapore, Singapore; emails: {e.esiner, u.tefek, daisuke.m}@adsc-create.edu.sg; B. Chen, Singapore University of Technology and Design, Singapore, Singapore; email: binbin_chen@sutd.edu.sg; Z. Kalbarczyk and D. M. Nicol, University of Illinois Urbana, Champaign, IL; emails: {kalbarcz, dmnicol}@illinois.edu.

2378-962X/2023/10-ART24 $15.00
https://doi.org/10.1145/3607194

1 INTRODUCTION

Increasing levels of automated control in **industrial control systems (ICSs)** call for stringent security checks and controls. An essential security measure is the verification of message source and integrity. Message authentication solutions provide such protection through the implementation of message authentication codes or digital signatures. However, message authentication alone is insufficient if a legitimate device in the system is compromised and controlled by an adversary for malicious command injection. Adversaries can exploit vulnerabilities through **a virtual private network (VPN)** [2] or user interfaces [3] or employ malware to gain access to less protected devices, through which they can publish malicious control commands. These malicious commands circumvent message authentication checks because their source is supposedly a legitimate device in the system. Against such threats, the state-of-the-art ICSs are equipped with security appliances or virtual network functions such as firewalls and intrusion detection systems on the message delivery paths. Examples of multi-hop communication in ICSs are shown in Table 1. For the security middle-boxes to be effective, the destinations must ensure that the messages are indeed transmitted via the right network path and hence undergo all the necessary checks and mediations. Such information is provided by message provenance, which summarizes the delivery path of a message and the actions performed on it.

The provenance of electronic data is, in general, defined as "the derivation from a particular source to a specific state of an item" in [4]. In this article, we propose a provenance verification solution called F2-Pro to enable the verification of the message delivery path and actions performed on the message en route. F2-Pro provides the devices along a message delivery path with a mechanism to generate and attach to the message a piece of cryptographically verifiable evidence, which can then be verified at the destination. We provide formal definitions and the cryptographic security proof for F2-Pro.

The cost of generating, communicating, and processing verifiable provenance information must always be weighed against the security threats that can be prevented by using provenance as a security measure. F2-Pro features a lightweight design, enabling real-time processing and flexible deployment options to facilitate adoption. Certain packets in time-critical ICSs, such as those in IEC 61850 for electric substations, need to be delivered within 2 milliseconds, rendering public-key cryptography infeasible [5]. F2-Pro is based on symmetric key cryptography and lightweight aggregate message authentication codes [6] for real-time performance.

A number of communication models need to be supported for various ICSs, their protocols, and devices. Certain communication modes involve hops (e.g., firewall, substation gateway, etc.) between the control center and **intelligent electronic devices (IEDs)** of a substation automation system, whereas others require remote access by a field service engineer via a field service device, or an ICS operator device in general. Thus, it is desirable to have security schemes flexible enough to fit into various systems and communication settings. F2-Pro features a **bump-in-the-wire (BITW)** deployment option to maintain flexibility and compatibility with legacy devices.

ICSs may require configuration and control by field service engineers and technicians by means of physically connecting an ICS operator device (such as a laptop computer) to the system. F2-Pro accommodates a multi-factor user authentication mechanism whereby the provenance chain is started from the human user operating an ICS operator device for issuing commands, and verified at the destination device for user credentials, device fingerprints, and delivery path. The multi-factor authentication extension of F2-Pro mitigates the risks introduced by the human factor and commercial-off-the-shelf ICS operator devices.

This article's contributions are summarized as follows:

- Our provenance verification solution features a lightweight design based on symmetric key cryptography and aggregate message authentication codes.

- The communication overhead of F2-Pro is constant, and the complexity of the verification algorithm scales linearly with the number of nodes on the message path.
- Our solution accommodates a multi-factor user authentication mechanism for the human user, such as the field service engineer, where the provenance chain starts from the field service device.
- We present formal definitions, the correctness proof, and the cryptographic security proof for F2-Pro.
- Our prototype implementation on a low-cost embedded device meets the requirements of time-stringent ICS communication settings.

The rest of this article is organized as follows. Section 2 presents the threats against substation automation and remote control systems and the design goals against such threats. Section 3 introduces the provenance verification model, algorithms, and construction, followed by the algorithm definitions, formal correctness, and security proofs in Section 4. Section 5 describes the multi-factor authentication solution to start the F2-Pro provenance chain from a human user. Section 6 discusses practical security implications and implementation considerations for F2-Pro followed by the evaluation of F2-Pro on a power grid testbed. Section 7 presents relevant literature before the conclusion in Section 8.

2 BACKGROUND, THREATS, AND GOALS

2.1 Overview of Industrial Control Systems

In this section, we begin with a general description of ICSs and discuss security challenges. We then provide the details of a smart grid system, as a representative ICS.

Based on the Purdue Model [7], recent ICSs, in general, are organized as shown in Figure 1. Level 3 and below are the **operation technology (OT)** systems. Level 0 includes a number of sensors and actuators physically connected to the plant equipment, and intelligent devices at Level 1 with computation and communication capabilities collect data from sensors and operate on actuators. Intelligent devices communicate with the local **human-machine interface (HMI)** on each outstation and/or systems in the control center (Level 3) using ICS protocols. The specific protocols utilized differ depending on the types of systems and plants. In the recent OT systems, typically VPN interfaces are set up for the sake of remote access by employees as well as system/device vendors for remote maintenance.

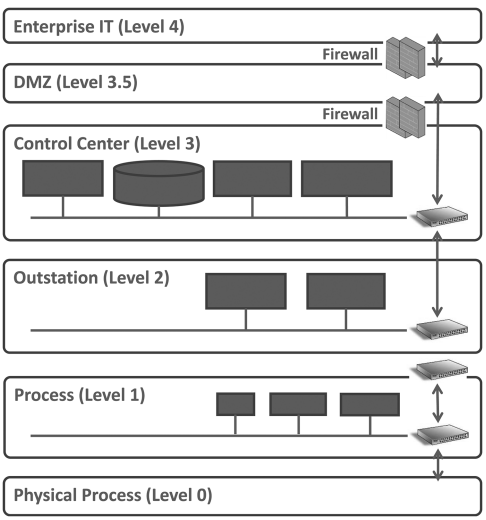


Fig. 1. Overview of industrial control systems.

2.2 Cybersecurity Threats and Attack Model against Industrial Control Systems

In this section, we discuss cyber attacks that could risk the operation of ICSs by referring to the system model discussed in the previous section and high-profile attack incidents.

Malicious Command Injection: Attackers can mount **man-in-the-middle (MITM)** attacks on the **wide-area network (WAN)**, which connects a control center (Level 3) to outstations in the

field (Level 2), or within the control center **local area network (LAN)** [8, 9], to replay, forge, or tamper with ICS messages.

Malicious Firmware or Configuration: This can be seen as a special case of malicious command injection. An engineering workstation at Level 3 would be sending updated firmware, ICS device configuration, and automated control logic to intelligent devices at Level 1. Once these are tampered with or any unauthorized update of these is made, it would cause significant damage to the physical plant as well as the stability of the plant operation.

False Data Injection: Similarly to false command injection, attackers could inject fake data into the **supervisory control and data acquisition (SCADA)** monitoring communication (from Level 1 to Level 2 and 3 as well as within Level 1) to confuse and reduce the situational awareness of the control center system [10] or into **programmable logic controllers (PLCs)** whose logic depends on power grid measurements. False data can also be injected via compromised field devices, e.g., by using a malware [11].

Compromised or Rogue Field Device: Attackers or malware may first target devices at Level 1 that are often less protected owing to the lack of strong cybersecurity protections. As demonstrated by CrashOverride [12], advanced malware could be used for publishing malicious control commands, executing features beyond its scope, and mounting MITM attacks.

Remote Attack via VPN: Attackers may remotely access the system via the VPN interface at Level 3 or Level 2. For instance, by exploiting vulnerabilities like *Shellshock* [2], an attacker can virtually become a part of the control center or outstation local area network and execute arbitrary shell commands on the VPN server to inject malicious commands through it. A malicious insider with knowledge of the VPN credential can mount an attack in a similar way.

Attacks via User Interface: Insider attacks against the SCADA master or HMI are also a serious concern [11]. As seen in the Ukraine incident [13], the SCADA master system can be remotely manipulated by an external attacker. Compromised field service (maintenance) devices or abuse of such devices would also cause similar outcomes [11].

2.3 Substation Automation and Remote Control in Smart Grid

The smart grid, an electrical power grid system empowered by information and communication technologies, is one of the most critical infrastructures for our lives as well as one of the most latency stringent examples of industrial control systems. In this section, we elaborate on the detail of it to provide a context for the rest of this article.

A typical smart grid system includes a control center and multiple (possibly thousands of) substations in the field. Figure 2 shows one substation connected to a control center via a WAN. The WAN can be a wired, private network or a private channel over a public network (e.g., the Internet) or through a wireless (e.g., cellular) network. Modern substations use standardized technologies like IEC 60870-5-104 or DNP3.0 for tele-control and IEC 61850 for substation automation [14].

Within the substation, as discussed in the reference model published by IEC TC57 [14], IEDs serve as the communication endpoints on the cyber side. They are responsible for operating on physical power system devices, e.g., circuit breakers and transformers. Real-time communication among IEDs is crucial for the automated protection of components. For example, a device or system fault event detected by one IED needs to trigger other IEDs to take immediate actions [15]. PLCs are also common devices in charge of automated control based on various power grid measurements. In addition, according to the reference model in [14], typically there is a substation gateway at the entrance of a substation [14]. The substation gateway often performs protocol translation (an example of message transformation en route), e.g., between IEC 60870-5-104 and IEC 61850 [14]. Remote control and monitoring of substations may involve **remote terminal units (RTUs)**, which receive commands from the control center and then interact with IEDs accordingly. The HMI for

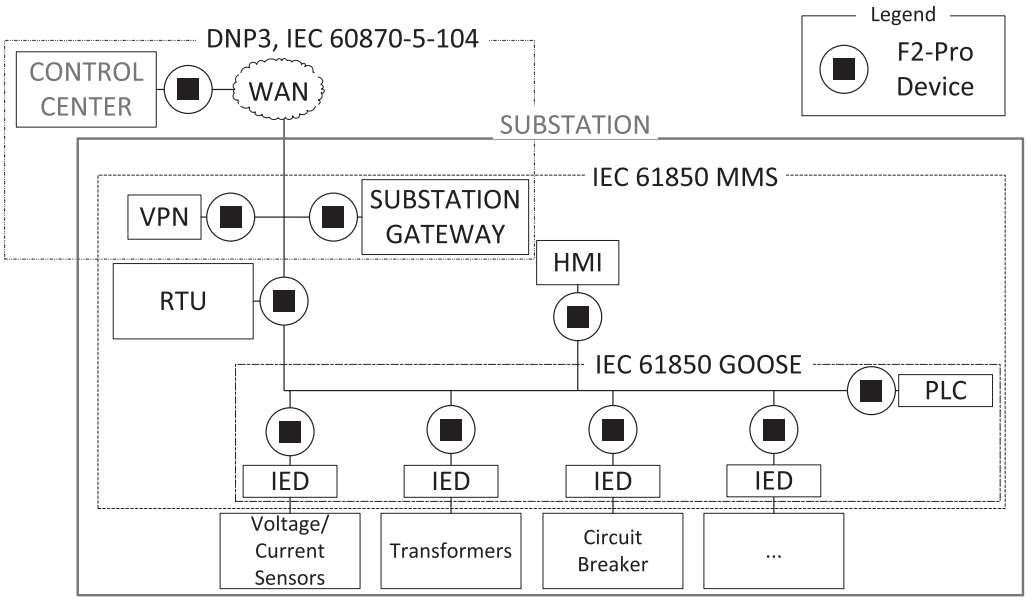


Fig. 2. Substation automation and remote control: a conceptual architecture, key components, and protocols and protection with F2-Pro devices.

Table 1. Key Communication Models in Substation Automation and Remote Control

Use Case	SCADA Master	FSD	High-volt. Substation	Substation VPN	Substation Gateway	Substation HMI	RTU	PLC	IED	Acceptable Delay
SCADA Control/Monitoring	S		M	M	M, S		M, S	D	D	< 100 ms
Operation on Substation HMI						S	M	D	D	>100 ms
Reporting by Field Device	D				M, S	D	M, S	S	S	>100 ms
Automated Control in Substation								S	D	10–100 ms
Status Update in Substation								D	S, D	2–10 ms
Protection to Switchgear								D	S, D	< 2 ms
Remote Maintenance		S		M	D, M			D	D	–
Local Maintenance		S				D		D	D	–

S: Source, D: Destination, M: Intermediary.

local control and monitoring by human operators is often found in large substations, and the HMI may be connected to RTUs for exercising control and monitoring. Lastly, in order to enable remote maintenance by grid operators or device vendors, VPN devices connected to the public network are increasingly deployed. The VPN may also be used as a backup SCADA.

Table 1 summarizes the typical communication patterns observed in the smart grid system. As shown in the table, some communication settings involve only a source and a destination. In IEC 61850-compliant substations, *Status Update in Substation*, *Automated Control in Substation*, and *Protection to Switchgear* are done using **Generic Object Oriented Substation Events (GOOSE)**, which is a publisher-subscriber-type multicast communication protocol.

In other communication settings, multiple intermediaries (hops) are involved. For instance, in *SCADA Control/Monitoring*, commands from a control center to a field device in a low-voltage substation may be mediated by devices in a high-voltage substation. Note that, for such communication, when protocol translation is involved, the substation gateway may work as a source of the translated messages. Besides, commands from a control center are often sent to a gateway

device or RTU of a substation. The substation gateway may perform translation of the protocol (e.g., from IEC 60870-5-104 to IEC 61850). Likewise, an RTU may have the autonomy to modify command semantics (e.g., a single incoming control command from the control center may be subdivided into multiple commands and sent to multiple IEDs and/or PLCs). In such a case, the substation gateway or RTU may also work as a source of communication. Another complicated communication model is the *Reporting by Field Devices*, where measurements from PLCs or IEDs are first sent to the substation gateway or RTU, which may perform protocol translation and/or message aggregation before forwarding them. In this case, again, the substation gateway behaves as a secondary source.

The *Remote Maintenance* use case corresponds to a situation where a grid operator or device vendor performs firmware or configuration updates remotely. In such a situation the VPN interface may be involved in the communication path. The same may apply to the *SCADA Control/Monitoring* case when the SCADA master connects to substations via VPN. *Operation on Substation HMI* may be direct or involve single or multiple hops such as an RTU, which either operates on physical power system devices or interacts with IEDs according to the operation performed on the HMI.

The last column of Table 1 shows the message delivery latency requirements found in the public guidelines [16, 17]. Two use cases corresponding to maintenance are not sensitive to latency and therefore are left blank. As seen in the table, the latency requirements vary depending on use cases, and communication within substations has very stringent delivery latency requirements. In particular, *Protection to Switchgear* requires very short latency (below 2 ms [16]). An example is the control of circuit breakers for system protection, as any delay of such communication would result in damage to transmission lines and, in the worst situation, a power blackout. The latency requirement with entities outside a substation is less stringent (i.e., around 100 ms or more). However, operations related to distribution automation still require a sub-100-millisecond delay, even over a WAN [17].

2.4 Design Goals

To counter injection by malicious parties, it is crucial to verify the source and integrity of messages. Protection of this type can be typically achieved by message authentication, using message authentication codes or digital signatures. Against injection by compromised field devices, the verification of the message source (and integrity) is insufficient because the message source is already a legitimate device or possesses the credentials of a legitimate device in the system.

To counter injection via compromised or rogue field devices, verification of the message delivery path is imperative. If an attacker steals an authorized source's credentials and inserts a message using an alternative entry point or path (e.g., from the malware on a field device), the path verification enables the destination to check whether the message has been *witnessed* by the expected set of nodes. For example, downstream nodes can regard the messages as trusted if a message has gone through a well-protected high-voltage substation system. As discussed in Section 2.3, the substation gateway performs protocol translation. In such a case, verifying the identity of the protocol translator allows the destination to check if the conversion is performed by a legitimate protocol translator, i.e., the substation gateway.

Based on these observations, we introduce the concept of *provenance* for addressing security concerns. Specifically, we focus on checking the authenticity of the message source and verifying the end-to-end message delivery path (i.e., which nodes are involved and in what order) as well as message transformation en route.

Systematically verifiable provenance information based on cryptographic proofs can be utilized for a variety of policy checks and decision-making regarding the legitimacy of a message.

In summary, our design goals are:

(1) providing ICSs with verifiable provenance information for message authentication, including the source of messages, message delivery path, and message transformation; (2) developing a low-latency cryptographic mechanism for generating and verifying provenance information within the stringent latency requirements for conveying verifiable provenance information; (3) developing a flexible solution for supporting various ICS settings and communication models (e.g., those listed in Table 1 in the smart grid context); and (4) enabling seamless multi-factor authentication for field service devices to issue commands to the protected systems and introduce command scope for authentication. Therefore, the generated codes cannot be used to execute features beyond their original scope as in CrashOverride.

3 PROVENANCE VERIFICATION

This section presents the system model and the core construction to achieve the above design goals. The notation used in the rest of this article is described in Table 2.

3.1 Model and Algorithms

In our message provenance verification model, there is a message source, a message destination, and intermediaries on the message delivery path. The source wants to prove its identity and the integrity of its messages to the destination. The intermediaries witness the message and may or may not extend it. The intermediaries that witness or extend want to prove to the destination that they have witnessed or extended the message. The destination wants to verify the provenance of the message. We provide below the definitions of the Initiate algorithm for the source, Witness and Extend algorithms for intermediaries, and Verify algorithm for the destination.

- Setup: The Setup is reserved for key generation and distribution.
- Initiate: The Initiate algorithm is run at the source to generate evidence to prove source identity and message integrity to the destination. The output evidence can be used as an input for the Witness and Extend algorithms.
- Extend: An intermediary runs the Extend algorithm with the received message(s), their evidence, and a new (or transformed) message as inputs to prove that the message passed through the intermediary and that the new message is added by this intermediary.
- Witness: An intermediary runs the Witness algorithm to perform Extend without adding a new message.
- Verify: The destination runs the Verify algorithm to verify the message provenance.

The correctness requirement is that, if the Initiate, Extend, Witness, and Verify algorithms are implemented correctly and the message is correctly sent from a source node to a destination node through a set of nodes as intermediaries, then given the correct inputs, the Verify algorithm should always accept the provenance of the message. The formal correctness definition is in Section 4.2.

3.2 Construction Overview

A practical way to introduce additional security into existing ICSs is to deploy transparent, BITWdevices [18–21]. Namely, the host ICS devices send and receive messages in an as-is manner, while the added BITW devices intercept messages and provide protection through additional checks without affecting the endpoints. For legacy compliance, F2-Pro is intended for such BITW devices. F2-Pro is also implemented as a software library that can be called by the products if altering the source code is an option, e.g., pre-production or via a software update.

As illustrated in Figure 2, to provide comprehensive coverage, an F2-Pro-enabled BITW device (or F2-Pro device for short) can be introduced to each key communication node (e.g., a SCADA

Table 2. Notations

Notation	Description
SK_i	A secret key that belongs to a node i.
$\mathcal{K}$	The security parameter.
AT	The verification key. It is subscripted to denote its "from" and "to" (s.a., AT_{SD} to denote it is the verification key that serves between source S destination D).
ev	The cryptographic evidence every node prepares when they *initiate*, *witness*, or *extend* a message. ev_i is the cryptographic evidence generated by the i^{th} node on the path of the message. To denote a cryptographic evidence "from" and "to," we use the same notation with AT.
S, W, D, U	The source, witness, destination, and human user node IDs, respectively.
$\mathcal{S}, \mathcal{D}$	The number of source and destination nodes, respectively.
MSG	A set of messages. When a message is *initiated* by a source, if it is *extended* by any downstream node, the new message is added to this set.
$\mathcal{A}$	The blackbox PPT adversary.
$\mathcal{C}$	The challenger in the game that plays the role of the destination node and the honest nodes.
$\mathcal{MC}$	The challenger in the Existential Unforgeability under Chosen Message Attack (EUF-CMA) security game.
$\mathcal{B}$	$\mathcal{B}$ plays the role of $\mathcal{C}$ against $\mathcal{A}$ and acts as the adversary to $\mathcal{MC}$.
$\mathcal{HN}$, $\mathcal{MN}$	The sets of honest nodes and the malicious nodes.

master, IEDs, PLCs, substation gateways). Alternatively, a smaller number of F2-Pro devices could be strategically installed only for critical components.

At a high level, an F2-Pro device at the source end of the communication intercepts and "wraps" a message for additional security. Another F2-Pro at the destination end performs verification and security policy enforcement based on the contained provenance information before it "unwraps" and passes the original message to the target device. If the message passes through intermediary nodes to which F2-Pro devices are installed, intermediary BITW devices can further "wrap" the message to attest that the message has indeed passed through the intermediary. If an intermediary device performs protocol translation, the translator's identity and both the original and translated messages can be verified at the destination F2-Pro device.

Interception and policy enforcement by F2-Pro can be performed selectively based on, for example, the types of messages, protocols, target devices, and so forth. This way, F2-Pro has minimal impact on system throughput.

At its core, F2-Pro is based on aggregate message authentication codes [6]. We use hash-based message authentication code ($HMAC_K$) employing SHA256 in the construction with the key K denoted with a subscript. In the rest of the article, we use the function "Agg" defined as follows:

Let "||," "⊕," "ev," and "pID" denote the concatenation operation, XOR operation, previous node's evidence, and previous node's ID in the chain, respectively:

$$\text{Agg (ev, K, ts, msg, pID)} = \text{ev} \oplus \text{HMAC}_K \text{ (ts } \| \text{ msg } \| \text{ pID)}.$$

Here, the timestamp is to prevent replay attacks, while the pID serves to establish the order in the chain. Each source and intermediary node computes the Agg function with the ev it obtains from the previous node (if any) and passes the resulting evidence to the next node in the chain.

All source BITW devices store a unique secret key SK, and all destination BITW devices store a unique verification key for each source, hereby called authentication token or AT, derived by hashing the secret key of each source with a salt value unique to the destination, such that AT_{SD} = hash(SK_S || $salt_D$) for source S and destination D. Similarly, a hash function with sufficient bit

Fig. 3. Message provenance with a single source and destination.

security, such as SHA256, can be employed. An **authentication token (AT)** is used as the key to the Agg function.

To *witness* the messages passing through the intermediary nodes, intermediary BITW devices store both a secret key and the authentication tokens corresponding to the source devices. When a message passes through an intermediary node, its BITW device computes an evidence and attaches the result to the encapsulated packet to attest to witnessing the message. When a message is transformed (e.g., via protocol translation) by an intermediate node, the corresponding F2-Pro device needs to act both as a source for the transformed message and as a witness to the original message; we call this operation *extend*.

In the following, we present the construction of F2-Pro. Concretely, the F2-Pro protocol creates a chain of keyed cryptographic digests derived from the messages and device identifiers, hence offering a time-stringent provenance scheme. The core F2-Pro protocol supports a wide range of functionalities required in a smart grid, such as verification by intermediate nodes (e.g., firewall/gateway), combining *extend* and intermediate verify (e.g., message aggregation by a data concentrator when reporting measurements from IEDs).

3.3 F2-Pro Construction

F2-Pro is a tuple of algorithms (*Setup, Initiate, Witness, Extend, Verify*). *Setup* creates the keys and authentication tokens and distributes them. *Initiate* generates cryptographic evidence for a message that has been sent by the host device that the F2-Pro device is attached to. *Witness* alters the already attached cryptographic evidence as the message passes through the host device to which an F2-Pro device is attached. *Extend* is similar to *witness* except that the message is also altered.

Finally, *Verify* checks the authenticity of the cryptographic evidence and either accepts or rejects it.

Initiate-verify: Figure 3 illustrates the protocol in a communication setting with one source and one destination node. The aim is to allow the user to prove its identity and message integrity to the destination with minimal overhead in terms of time added to the original protocol.

The source F2-Pro possesses a secret key (SK_S) and the destination F2-Pro stores an authentication token (AT_{SD} = hash (SK_S || $salt_{SD}$)). The authentication request, which proves identity to the destination F2-Pro, is a cryptographic evidence (ev_{SD} = Agg (“”, hash (SK_S || $salt_{SD}$), ts, “”, “”)) of a message with no content and no previous node ID and evidence. Since there may be more than one destination, there is a unique salt value associated with each destination node. This salt value can be the destination **message authentication protocol (MAC)**/IP address, randomly generated from a seed, or can be stored. To prevent replay attacks, the evidence should be time-bound. Hence, to send the authentication request, the source F2-Pro adds a timestamp (ts) to the evidence. To authenticate a message along with its identity, the source F2-Pro adds the message content (msg) to the computation, resulting in

$$ev_{SD} = \text{Agg (“”, hash } (SK_S \,||\, salt_{SD}), \text{ ts, msg, “”)},$$

where the previous ev and the previous node ID fields are empty. For verification, the destination F2-Pro uses the timestamp, the message, and its stored authentication token (ev’ = Agg (“”, AT_{SD}, ts, msg, “”) and checks two things: if the timestamp is fresh and if ev’ equals the received ev_{SD}.

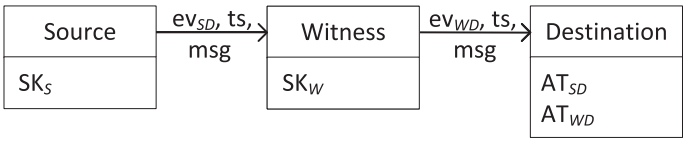


Fig. 4. Message provenance with single witness.

This basic case covers some of the communication models discussed in Section 2.3, namely *Automated Control in Substation*, *Status Update in Substation*, *Protection to Switchgear*, and *Local Maintenance*. In the following, we demonstrate the flexibility of this design to support various smart grid communication models before we move to more time-stringent segments.

Adding Witnesses En Route (Witness): As discussed in Section 2.3, many smart grid communication models involve multi-hop communication. In some cases, a message is received by an intermediate node and then simply forwarded to the destination. For example, in the *Remote Maintenance* scenario, a VPN interface may operate in this way. In the *SCADA Control/Monitoring* case, the high-voltage substation may also forward information.

In Figure 4, note that the destination F2-Pro has two ATs, one for the source node and the other for the witness node (AT_{WD} = hash (SK_W || $salt_{WD}$)). The source F2-Pro node *initiates* the chain by sending ev_{SD} to the witness node. The witness F2-Pro then derives its own cryptographic evidence, using SK_W and the cryptographic evidence from the source F2-Pro (or from the last witness F2-Pro if any):

$$ev_{WD} = \text{Agg}\ (ev_{SD}, \text{hash}\ (SK_W \parallel salt_{WD}), \text{ts}, \text{msg}, \text{S}).$$

Aggregating the chain of cryptographic evidence in this manner limits the size of evidence to a single HMAC output, regardless of the number of witnesses on the path.

Verification at the destination side entails calculating the cryptographic evidences from the first to the last. Then, the destination checks if the last ev' is equal to the received cryptographic evidence. In our example, with one source and one witness, the destination performs

$$ev'_0 = \text{Agg}\ (\text{“”}, AT_{SD}, \text{ts}, \text{msg}, \text{“”})$$
$$ev'_1 = \text{Agg}\ (ev'_0, AT_{WD}, \text{ts}, \text{msg}, \text{S})$$

and then checks if ev'_1 equals ev_{WD}, the evidence received from the witness.

While the Initiate and Witness algorithms' complexities are both constant, i.e., (O(1)), the complexity of the verification algorithm is linear on the number of intermediaries, i.e., (O(N)), where N is the number of intermediaries).

Message Transformation (*Extend*): In the setting with two sequential sources (as in Figure 5), the *extending* (second) source F2-Pro also acts as a witness while concatenating a new (transformed) message and calculating the next value in the chain. For instance, upon receiving the inputs, "ev_{S1D}, ts, msg," a witness would calculate its cryptographic evidence: Agg (ev_{S1D}, hash(SK_W || $salt_{WD}$), ts, msg, S1). Instead, an *extending* source F2-Pro renders it as follows: ev_{S2D} = Agg (ev_{S1D}, hash (SK_{S2} || $salt_{S2D}$), ts, msg || msg2, S1), where msg2 is the new (transformed) message. Then, it sends ev_{S2D}, ts, msg, msg2 to the next node. Note that the cryptographic evidence does not prove the correctness of translation from msg to msg2 but enables the verification of the extending node's identity and the integrity of msg and msg2.

We present the verification pseudocode in Algorithm 3.1 for an arbitrary number of nodes on the path. Lines 5 to 6 decide the indices of messages to be processed on Line 7, where the evidences are computed.

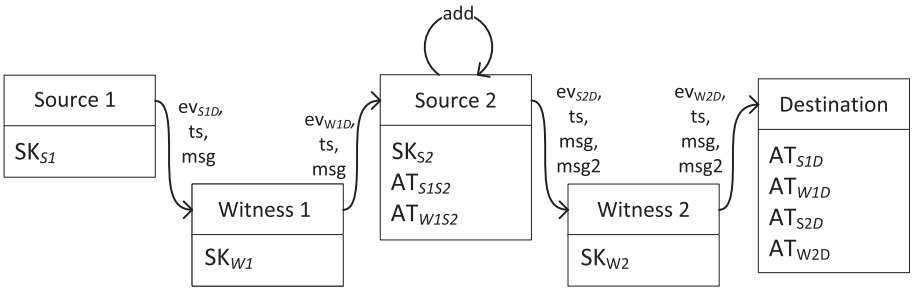


Fig. 5. Example: the second source modifies the message.

ALGORITHM 3.1: Provenance Verification

Data: ev, ts, MSG, PATH
Result: accept/reject

```
ev'_0 = Agg ("", AT_SD, ts, MSG[0], "") c = 1
sourceCounter = 0
while c++ ≤ PATH.length do
    node_x = PATH[c];
    if node_x is another source then
        sourceCounter++
    ev'_c = Agg (ev'_{c−1}, AT_xD, ts, MSG[0 : sourceCounter], PATH[c − 1])
if ev'_c = ev then
    accept
else
    reject
```

4 DEFINITIONS, CORRECTNESS, AND SECURITY

In this section, we present the definitions and the formal proof for the core protocol. Based on our threat model, an attacker cannot physically access the verifier BITW device implemented at the protected zone of outstations. In addition, because the verifier device is implemented as a transparent, bump-in-the-wire device, it is not addressable by network-based attackers. Therefore, as long as the software running on it and the operating system are secure against attacks such as buffer overflow, the secret information (e.g., keys) on the destination BITW devices is secure.

4.1 Algorithm Definitions

We use subscripts to denote an element of a set and superscript to denote a state of the set. For instance, if MSG_1 extends MSG_0, the extended message of $\{MSG_0, MSG_1\}$ is denoted as MSG^1. The cryptographic evidence (ev) is to be sent along with the plaintext timestamp (ts) and message. The setup is performed at every F2-Pro device, where each device calculates the authentication tokens from its secret key and distributes them to the corresponding destinations. At the destination F2-Pro device, the algorithm "Verify" derives the information about which authentication tokens to use and which step of the chain involves message transformation (to add the corresponding MSG_i as an input to the chain). In our construction, the "Witness" algorithm is identical to the "Extend" algorithm with an empty message field.

- Setup: $SK_S \leftarrow KeyGen(1^{\mathcal{K}})$ – Given the security parameter $\mathcal{K}$, a secret key (SK_S) is generated for every source S in the system.

$AT_{SD} \leftarrow$ AuthTokenGen(SK_S, D) – Each source S, given the secret key and a salt value, generates an authentication token AT_{SD} per destination D. We assume secure distribution of these tokens to the corresponding destinations.

- ev_0, MSG $\leftarrow$ Initiate(SK_0, ts, message, D) – Given the secret key, the timestamp, a message, and the destination, outputs the first cryptographic evidence (ev_0) of the chain and the set of messages with one message in—the later nodes in the path may add messages to the set (see "Extend" below).
- $ev_c \leftarrow$ Witness(ev_{c-1}, D, SK_c, ts, MSG) – Given the previous node's cryptographic evidence, the destination, the secret key, the timestamp, and the set of messages, outputs a witness cryptographic evidence.
- ev_c, MSG' $\leftarrow$ Extend(ev_{c-1}, D, SK_c, ts, message, MSG) – Given the previous node's cryptographic evidence, the destination, the secret key, the timestamp, a message, and the set of previous messages, outputs an extension cryptographic evidence and the extended message set.
- accept/reject $\leftarrow$ Verify(ev_c, ts, AT_{SD}, $AT_{1D}, \ldots, AT_{cD}$, MSG) – Given the last node's cryptographic evidence (ev_c), the timestamp, the set of authentication tokens (that consists of the verification keys of the source node AT_{SD} and other nodes on the path $AT_{1D}, \ldots, AT_{cD}$), and the set of messages, outputs "1" for accept, "0" for reject.

After successful verification, the verified provenance information is handed over to policy checking, which is outside of the core protocol algorithms.

4.2 Protocol Correctness

Below, we denote the number of nodes and sources on the path by $\mathcal{X}$ and $\mathcal{Y}$, respectively. By the correctness requirements in Section 3.1, F2-Pro's correctness is as follows.

$\forall$ $MSG_i \in \{0,1\}^*$, if **[**$SK_S \leftarrow$ KeyGen($1^{\mathcal{K}}$), $AT_{SD} \leftarrow$ AuthTokenGen (SK_S, D) for $S \in [0, \ldots, \mathcal{S}-1]$ and $D \in [0, \ldots, \mathcal{D}-1]$**]**; ($ev_0$, MSG^0) $\leftarrow$ Initiate(SK_S, ts, MSG_0, D); **[** $ev_x \leftarrow$ Witness(ev_{x-1}, D, SK_x, ts, MSG^y) **OR** (ev_x, MSG^y) $\leftarrow$ Extend(ev_{x-1}, D, SK_x, ts, MSG_y, MSG^{y-1}) **]** $\forall$ x $\in [1, \ldots, \mathcal{X}]$ and $\forall$ y $\in [1, \ldots, \mathcal{Y}]$; **then** Verify($ev_{\mathcal{X}}$, AT_{0D}, …, $AT_{(\mathcal{S}-1)D}$, $MSG^{\mathcal{Y}}$) = 1 with probability 1.

4.3 Correctness Proof

To prove the correctness of the protocol, we need to show that if a message is correctly sent from a source S to a destination D through a set of nodes as witnesses/extenders, then the Verify algorithm will return 1 with probability 1.

Proof. Assuming that the Initiate algorithm and the Extend/Witness algorithms are implemented correctly, the correctness of the protocol hinges on the properties of the cryptographic evidence produced by these algorithms. In particular, we need to show that if a witness cryptographic evidence ev_x is generated correctly by node x based on the previous node's cryptographic evidence ev_{x-1} and the set of messages, then the Verify algorithm will correctly verify ev_x with the corresponding authentication tokens and messages.

We prove the correctness of the protocol by induction on the length of the chain. For the base case, we consider a chain with only one node (i.e., source). In this case, the Initiate algorithm produces the cryptographic evidence ev_0 for the message MSG_0. Then, the Verify algorithm checks if ev_0 is valid by verifying it against the destination's authentication token AT_{0D}, which is generated by the source node 0 using its secret key SK_0 and the salt value D. Since the token is distributed securely to the destination, the Verify algorithm will accept the message if ev_0 is valid.

For the inductive step, we assume that the protocol is correct for chains of length up to $x-1$ and prove that it is also correct for a chain of length x. In this case, we assume that the cryptographic evidence ev_{x-1} produced by node $x-1$ is valid and that the Verify algorithm will correctly accept

it. Then, we need to show that if node x produces a valid witness/extend cryptographic evidence ev_x based on ev_{x-1} and the set of messages, then the Verify algorithm will correctly accept it.

To prove this, we note that the Extend algorithm generates the extended message set MSG' by appending the new message to the set of previous messages MSG. Then, the algorithm generates a witness/extend cryptographic evidence ev_x based on ev_{x-1} and the extended message set MSG'. Since ev_{x-1} is valid, it implies that the previous message set MSG is valid and was correctly sent from the source node to node $x-1$. Therefore, by appending the new message to the valid message set, the extended message set MSG' is also valid. Furthermore, the witness cryptographic evidence ev_x is generated using the salt value D and the secret key SK_x, which is only known to node x. Therefore, node x generated the valid evidence ev_x with the knowledge of SK_x. Thus, ev_x is valid, and node x generated ev_x correctly.

Finally, the Verify algorithm checks if ev_x is valid given that ev_{x-1} is valid, by verifying it against the authentication token of x, i.e., AT_{xD} derived from SK_x. The Verify algorithm checks if ev_x is valid by verifying it against the destination's authentication token AT_{xD}, which is generated by the node x using its secret key SK_x and the salt value D. Given that the token is distributed securely to the destination, the Verify algorithm will accept the message if ev_x is valid.

Therefore, by induction, if the Initiate, Extend/Witness, and Verify algorithms are implemented correctly and the message is correctly sent from a source node to a destination node through a set of nodes as witnesses/extenders, then the Verify algorithm will return 1 with probability 1. Hence, the protocol is correct. □

4.4 Security Proof

We state the security for a verifiable provenance scheme using a game.

Definition 1. A verifiable message provenance scheme is secure if, for all **probabilistic polynomial-time (PPT)** adversaries $\mathcal{A}$, the probability that $\mathcal{A}$ wins in the below game is negligible in $\mathcal{K}$ (the security parameter).

Forgery Game: If any node on the path of the message from a source to a destination deviates from the honest behavior or a message does not pass through all the honest nodes on the claimed path, then the destination will detect it with a high probability. We define a forgery game to be played between the adversary $\mathcal{A}$, who acts as the malicious parties, and the challenger C, who plays the role of the destination and honest nodes.

- **Setup:**
  - $\mathcal{A}$ chooses two sets of node IDs for malicious nodes $\mathcal{MN}$ and for honest nodes $\mathcal{HN}$ of size h. $\mathcal{A}$ picks unique secret keys for the nodes in $\mathcal{MN}$ in the format defined in Setup in Section 4.1. $\mathcal{A}$ sends $\mathcal{MN}$, the secret keys they share with the destination SK_{l_i},[1] $\forall\, l_i \in \mathcal{MN}$, and $\mathcal{HN}$ to C.
  - C generates the secret keys for the nodes in $\mathcal{HN}$, i.e., SK_{l_i}, $\forall\, l_i \in \mathcal{HN}$.
- **Query:**
  - $\mathcal{A}$ sends to C a polynomial number of queries, each of which includes a timestamp ts, messages MSG = $\{m_1, \ldots, m_j\}$ (m_1 is for Initiate, $m_i = m_{i-1}$ implies Witness, otherwise Extend), and the sequence of node IDs for a selected path $L = \{l_1, \ldots, l_j\}$ excluding the destination.
  - C calculates and returns $\{ev_1, \ldots, ev_j\}$ after each query.

[1]In any given path from the source to the destination, all nodes (l_i) share pairwise symmetric keys with the destination, denoted as AT_{iD}. To improve readability, we will refer to the secret key shared between any node i and the destination as SK_{l_i}.

- **Challenge:** $\mathcal{A}$ sends MSG* = $\{m_1^*, \ldots, m_k^*\}$, $L^* = \{l_1^*, \ldots, l_k^*\}$, ts*, ev* to C.
- **Winning Condition:** $\mathcal{A}$ wins if Algorithm 3.1 returns accept with ev*, ts*, MSG*, L^* as inputs and there exists at least one honest node $l_i^* \in L^*$, such that $l_i^* \in \mathcal{HN}$ and $\{m_1^*, \ldots, m_i^*\}$, $\{l_1^*, \ldots, l_i^*\}$ is not queried as a prefix of MSG, L with ts = ts* during the query phase.

Theorem 1. *The proposed F2-Pro scheme provides verifiable message provenance according to Definition 1, against any static active PPT adversary who controls the network and the malicious clients, assuming the underlying MAC scheme, specifically HMAC, is deterministic, collision resistant (Definition 4.10 in [22]), and* ***existentially unforgeable under adaptive chosen message attacks (EUF-CMA)*** *(Definition 4.2 in [22]).*

Proof. We reduce the security of our scheme to that of the underlying HMAC function. If a PPT adversary $\mathcal{A}$ wins the security game of our scheme with non-negligible probabilitym we use it to construct another PPT algorithm $\mathcal{B}$ that breaks the HMAC function's existential unforgeability under adaptive chosen message attack with non-negligible probability. $\mathcal{B}$ acts as the adversary in the security game with h EUF-CMA challengers $\mathcal{MC}_{l_i}, \forall\, l_i \in \mathcal{HN}$ picked by the Adversary in the Setup phase. In parallel, $\mathcal{B}$ plays the role of the challenger in our game with $\mathcal{A}$.

- **Setup:**
  - $\mathcal{A}$ chooses two sets of node IDs for malicious nodes $\mathcal{MN}$ and for honest nodes $\mathcal{HN}$ of size h. $\mathcal{A}$ picks the secret keys for the nodes in $\mathcal{MN}$ in the format defined in Setup in Section 4.1. $\mathcal{A}$ sends $\mathcal{MN}$, the secret keys they share with the destination $\text{SK}_{l_i}, \forall\, l_i \in \mathcal{MN}$, and $\mathcal{HN}$ to $\mathcal{B}$.
  - Each $\mathcal{MC}_{l_i}$ generates the secret key $\text{SK}_{l_i}, \forall\, l_i \in \mathcal{HN}$.
- **Query:**
  - $\mathcal{A}$ sends to $\mathcal{B}$ a polynomial number of queries, each of which includes a timestamp ts, messages MSG = $\{m_1, \ldots, m_j\}$, the sequence of node IDs for a selected path $L = \{l_1, \ldots, l_j\}$ excluding the destination.
  - To calculate $\{\text{ev}_1, \ldots, \text{ev}_j\}$, $\mathcal{B}$ calculates the HMAC for $l_i \in \mathcal{MN}$ and queries $\mathcal{MC}_{l_i}$ for $l_i \in \mathcal{HN}$; i.e., $\mathcal{B}$ calculates for each $i \in \{1, \ldots, j\}$ with ev_0,m_0 = "":
    $m_i' = \text{ts} \| \; m_i \; \| \; l_{i-1}$

$$\text{ev}_i = \begin{cases} \text{ev}_{i-1} \oplus \text{HMAC}_{SK_{l_i}}(m_i') & \text{if } l_i \in \mathcal{MN} \\ \mathcal{MC}_{l_i}(m_i') & \text{if } l_i \in \mathcal{HN}. \end{cases}$$

    $\mathcal{B}$ then sends $\{\text{ev}_1, \ldots, \text{ev}_j\}$ to $\mathcal{A}$ after each query and stores all queried $\{\text{ev}_i\}$s and $\{\text{ev}_{i-1}\}$s in database D (i.e., $\text{D}[\text{ev}_i] = \{\text{ev}_{i-1}\}$).
- **Challenge:**
  - $\mathcal{A}$ sends MSG* = $\{m_1^*, \ldots, m_k^*\}$, $L^* = \{l_1^*, \ldots, l_k^*\}$, ts*, ev* to $\mathcal{B}$, where k is such that ev_k = ev*.
  - $\mathcal{B}$ iterates over $i = \{k, \ldots, 1\}$. For each i:
    - If $l_i^* \in \mathcal{HN}$ and ev_i is queried before (i.e., $\text{D}[\text{ev}_i] \neq$ null), then $\mathcal{B}$ decrements a pointer p by 1 starting from k, and $\mathcal{B}$ fetches the previous ev queried with ev_i in database D.
    - If $l_i^* \in \mathcal{MN}$, $\mathcal{B}$ calculates the previous ev by $ev_i \oplus \text{HMAC}_{SK_{l_i}}(m_i^*, \text{ts}^*, l_{i-1}^*)$.
    - If $l_i^* \in \mathcal{HN}$ and ev_i is not queried before (i.e., $\text{D}[\text{ev}_i] =$ null), then p is set to i and $\mathcal{B}$ breaks this loop.

    At this point, $\mathcal{B}$ has found an evidence ev_p that has not been an output to any query before. Knowing the new winning output at p leading $\mathcal{A}$ to win the game, $\mathcal{B}$ now needs to

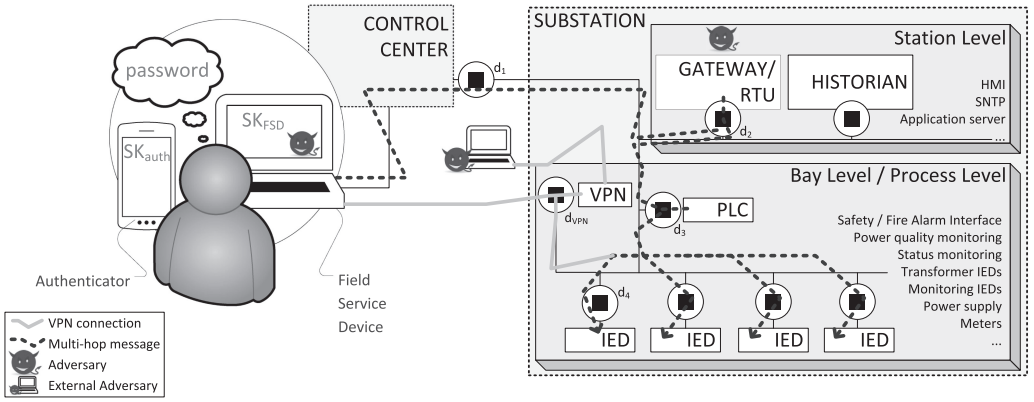


Fig. 6. ICS operator device (a field service device/laptop (FSD) in this example) interaction with an industrial control system (a modernized substation system in smart grid system in this example).

compute the previous evidence such that when XOR'ed with ev_p, it gives the output to $\text{HMAC}_{SK_{l_i}}(m_i^{*'})$ for $\mathcal{B}$ to use against $\mathcal{MC}_{l_p}$.

- $\mathcal{B}$ calculates ev_{p-1} by calculating ev_i from the beginning of the chain until $i = p-1$; i.e., for each $i = \{1, \ldots, p-1\}$ with $\text{ev}_0, m_0^* =$ "":

$$m_i^{*'} = \text{ts}^* \| \; m_i^* \; \| \; l_{i-1}^*$$

$$\text{ev}_i = \begin{cases} \text{ev}_{i-1} \oplus \text{HMAC}_{SK_{l_i^*}}(m_i^{*'}) & \text{if } l_i^* \in \mathcal{MN} \\ \mathcal{MC}_{l_i^*}(m_i^{*'}) & \text{if } l_i^* \in \mathcal{HN}. \end{cases}$$

- Then $\mathcal{B}$ challenges $\mathcal{MC}_{l_p^*}$ with $m_i^{*'}$ and $\text{ev}_{p-1} \oplus \text{ev}_p$.

$\mathcal{A}$ cannot win: In the scenario where $\mathcal{A}$ wins with a probability of ϵ, the probability of $\mathcal{B}$ winning is $\epsilon/h - \delta$. The division by h results from the breaking of $1/h$ EUF-CMA challengers. δ is the probability of $\mathcal{A}$ finding a collision in the HMAC function and is negligible. Since h is a constant, any non-negligible value of ϵ will also result in a non-negligible value of $\epsilon/h - \delta$. □

5 MULTI-FACTOR AUTHENTICATION WITH F2-PRO

In this section, we describe the use of F2-Pro to start the provenance chain from the human user. For control and configuration management by field service engineers/technicians (field person), we propose a multi-factor authentication solution that enables the verification of user identity, seamlessly integrated with the provenance verification provided by F2-Pro. As illustrated in Figure 6, the field person configuring the IEDs through the control center is authenticated via multiple factors: a memorized password (knowledge factor), a field service device, and an authenticator such as a smartphone, both associated with the field person (possession factors).

Without the multi-factor authentication, F2-Pro would operate as follows. The devices marked as d_1 to d_4 in Figure 6 represent our BITWdevices installed in the network to provide security. In the running example, the control center is sending a command to an IED. All the nodes on the path generate *verifiable evidence*. In this case, the device d_1 of the control center *initiates* the message, d_2 of the substation gateway *extends* it with the protocol-translated message, and d_3 of the PLC *witnesses* the message. Lastly, d_4 at the destination device *verifies* the source identity, message integrity, and message delivery path before enforcing security policies based on the verified provenance information. Next, we describe the multi-factor authentication for access via ICS user interfaces.

5.1 Provenance Chain Starting from the Human User

When the field person uses an ICS operator device, e.g., a field service device to manage control and configuration, the destination verifies two additional pieces of information: user identity and ICS operator device identity. The proof of ICS operator device identity is identical to that of other nodes in the system. To harden the system against the weaknesses introduced by the human factor, the proof of user identity is provided by multiple authentication factors. Here, the possession factor comes into the picture, to be used along with the password that the user memorized to generate the user fingerprint. In the multi-factor authentication *Setup*, the list of authentication tokens is generated as below for each user-destination pair and stored at the destinations.

$$\mathrm{AT}_{UD} = \mathrm{hash}(\mathrm{SK}_U \parallel \mathrm{password} \parallel \mathrm{salt}_D),$$

where SK_U is the secret key associated with user U, and AT_{UD} is the authentication token stored at destination D to verify messages from user U. Later, when the user uses the field service device to send a message, the user keys in the password in the authenticator device. The authenticator runs *Initiate* to calculate the user **one-time password (OTP)**:

$$\mathrm{OTP}_{UD} = \mathrm{hash}(\mathrm{AT}_{UD} \parallel \mathrm{ts}).$$

The user is required to select the destination for its message since the salt value and hence the OTP is associated with the destination ID. The user is expected to key in the OTP to the field service device. To provide easy and error-free use, a QR code generator/reader, NFC reader, or any one-way communication between the authenticator and field service device is beneficial. We used both NFC and QR codes in our product. Figure 7 visualizes the steps of the multi-factor authentication process detailed below.

Initiate: The value that has been generated by the authenticator is treated as an initiation without message content. Naturally, the length of a user session should span multiple messages. The user identity (e.g., username, ID) can be sent in plaintext for cases where there is more than one user so that the destination can choose the right verification key for each user.

Extend: After the authenticator's initiation, the generated OTP is transferred to the ICS operator device. Then, the ICS operator device *extends* the *initiated* chain with the message (e.g., configuration command) and passes it to the control center, where the message goes through the pipeline described in Section 3.

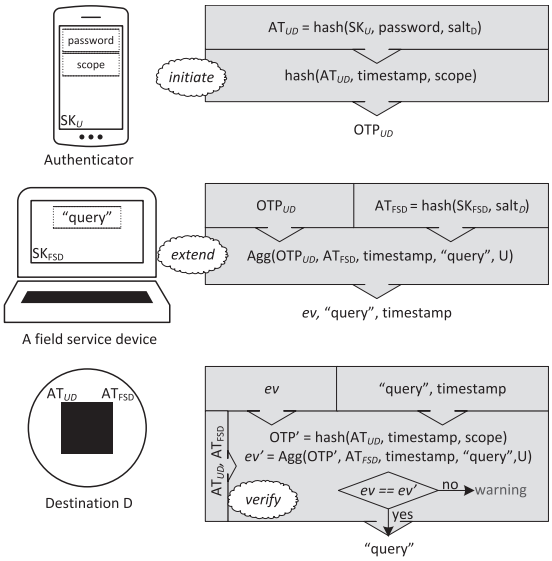


Fig. 7. Steps of multi-factor authentication.

5.2 Starting the Chain with a Public-key-based Scheme

An option to implement multi-factor authentication is employing an RSA-like public-key-based scheme (SETUP, SIGN, VERIFY) and integrating it into the provenance checking mechanism as follows. At the authenticator:

$$\mathrm{OTP} = \mathrm{SIGN}_{priv_k}(\mathrm{username}, \mathrm{ts}).$$

The authenticator needs to send the username and the timestamp in plaintext to the ICS operator device together with the OTP. At the ICS operator device:

$$\text{ev} = \text{Agg}(\text{OTP}, \text{hash}(\text{SK}_{FSD} \parallel \text{salt}), \text{ts}, \text{msg}, \text{pID}).$$

The ICS operator device then needs to send the username and timestamp in plaintext to the control center, and these values should propagate until the destination together with the ev. To verify, the destination XORs all the evidence on the path and verifies the resulting value with the public key of the user.

The verification overhead at the destination is heavy, and this method is only possible if the destination device supports PKI. For instance, in our selected embedded device (BeagleBoard-X15 [23]), extra verification for the multi-factor authentication takes 40 usec, whereas the same verification step takes around 300 usec with a public key signature scheme. This affects the throughput of the destination device where multiple sources may have been messaging, and it also weakens the system against denial-of-service attacks.

Fig. 8. Prototype authenticator device.

5.3 Limiting the Scope of Authentication

The extension of F2-Pro with multi-factor authentication hardens the security for authenticating human users who connect to ICS via ICS operator devices. These commercial-of-the-shelf devices, such as laptops and tablets, are vulnerable to malware. In the case where a field service device is infected, the malicious party can use the user's time-bound code (OTP) to issue other commands in the background for the duration of the session. Therefore, as another layer of security, we limit the scope of the OTP generated at the authenticator. Simply put, we enable the authentication scope selection for the user commands prior to OTP generation. The destination node understands the packet scope and starts the verification chain accordingly without requiring any additional communication overhead. The OTP at the authenticator is then calculated as follows:

$$\text{OTP}_{UD} = \text{hash}(\text{hash}(\text{SK}_U \parallel \text{password} \parallel \text{salt}_D) \parallel \text{ts} \parallel \text{scope}).$$

Authentication scope restriction introduces minimal computation overhead yet prevents malicious software from issuing irrelevant and potentially more harmful commands.

A screenshot from the proof-of-concept multi-factor authentication Android application is shown in Figure 8. The user selects a destination device and session scope from a drop-down menu and scans the resulting QR code (containing the OTP) with the field service device.

5.4 Policy Enforcement at Verifier

With the use of an authenticator, upon the successful verification of the cryptographic evidence attached to ICS messages or configuration, the verifier can derive the following information from the multi-factor user authentication added to the end-to-end provenance information through F2-Pro: (1) identity of the user, (2) identity of the authenticator, (3) identity of the ICS operator device, (4) timestamp, and (5) authentication scope.

Based on this information, the verifier device can enforce policies in terms of the combination of these. For instance, if we enforce $user_1$ to use a specific authenticator $authenticator_1$

and ICS operator device (e.g., mobile field service device) iod_1, we can define a policy to allow $user_1, authenticator_1, iod_1$. If $user_1$ is allowed to use another ICS operator device (e.g., SCADA HMI), say iod_2, we can define another policy that allows $user_1, authenticator_1, iod_2$. The timestamp and the scope can be further defined for each combination. This way, we can flexibly define security policies to support various use cases commonly seen in the ICS context.

6 SECURITY DISCUSSION, IMPLEMENTATION, AND EVALUATION

The demonstration videos of F2-Pro against attack scenarios discussed in this section are publicly available in [24] (on a controlled environment) and [25] (on a realistic testbed).

6.1 Attacks Countered by F2-Pro

To validate the security features of F2-Pro, we implemented three attacks that succeeded on an unprotected IEC 61850-based smart grid testbed. The first attack abuses a compromised entity within the system. This attack is a generalized rendition of CrashOverride (Industroyer) malware, which can issue IEC 61850 and IEC 104 commands either autonomously or orchestrated by a remote attacker. Once a device is compromised, the adversary can issue any command to any destination. Therefore, by compromising an entity in the system (e.g., a substation gateway from another substation), the compromised device can issue commands (e.g., a command to a circuit breaker) to several IEDs. Such commands are successfully executed, potentially damaging the system. The second attack abuses Shellshock vulnerability [2] to compromise the VPN service and escalates privileges to execute arbitrary shell commands. Consecutively, we can issue commands to IEDs and PLCs. All such commands are also executed in an unprotected system. The third attack connects a rogue laptop to the substation switch. The point of access is different, but the attack is still successful. When we activate F2-Pro devices on the wires, the first attack is countered; even though cryptographic evidence for the message is generated by the compromised device's F2-Pro device (i.e., a genuine secret key) and sent along with it, the destination F2-Pro device knows from which source and from which path to expect a particular command. As the actual network path is different from the path expected by the destination F2-Pro device, the F2-Pro device of the destination will not run the verification algorithm (Algorithm 3.1), stating, "The followed path cannot be found in the policy." For the second and third attacks, we assume a smart attacker would somehow spoof IPs and manipulate the system to claim that the correct path is followed. However, the corresponding verification cannot be deceived, as discussed with the security proof at Section 4.4, and the F2-Pro device returns "Cryptographic evidence mismatch."

6.2 Dependability; Failure of an F2-Pro Device or Its Network Connectivity

A failure of the F2-Pro device can be handled by redundancy. A critical smart grid device often has hot standby redundancy and/or it may have multiple network interfaces. Redundant F2-Pro devices can be connected to the hot standby devices and to each of the standby interfaces of a critical device. Hence, the failure of a single F-Pro device can be tolerated. The degree of redundancy should be decided based on the criticality of the host system and devices. Such an architecture would also include a watchdog mechanism that monitors the availability of the F2-Pro devices, e.g., based on timeout, to enable automated fail-over.

6.3 Verification Key Security

The security of the verification keys depends on the preimage resistance of the employed hash function family. Simply put, if a probabilistic polynomial time adversary can find the secret key SK, given a verification key (= hash(SK || salt)), then we can break the preimage resistance of the

chosen hash function. Therefore, if the hash function employed is preimage resistant, then the verification keys are secure.

6.4 Key Management

Key management is the fundamental basis of every cryptographic application. Key management for SCADA systems has been extensively studied. The main constraints have been presented in [26]. The key management concept consists of key generation, distribution, storage, and revocation, all of which are required in our proposed solution. There are two types of key management schemes. The first and the more studied type is automated key management. The second is the manual key distribution, which is more suitable for static setups. Key management software is often a small part of the program; however, designing a key management system is a task on its own. For instance, in [27] one of the first key management protocols was proposed. Soon after, a flaw was pointed out in [28], and a fix was proposed, which then was cracked in [29]. [30] revealed a new flaw in the original protocol. All these flaws are obvious when pointed out; however, they remained hidden for a long while from inexperienced eyes. This well-known example shows us that key management should not be taken lightly, and the most advanced scheme in the literature at the time of implementation shall be employed. In this work, we assume secure key distribution. F2-Pro remains neutral to the underlying method of key distribution. F2-Pro implements a manual key distribution approach, as detailed in Section 6.6, and similarly applies a basic authentication token derivation scheme that is independent of the proposed protocol and its extensions. The key derivation component is a modular aspect of the protocol that can be substituted with alternative techniques.

There are minor advantages of using authentication tokens over pairwise symmetric keys in cyberphysical systems. The first advantage is that authentication tokens provide flexibility to the source entity to generate tokens on the fly, using their single master key, without needing pairwise symmetric keys for bidirectional communication with every entity in the system. This approach saves the source from having to store a linearly growing number of keys for every destination node. Another minor advantage is that calculating the symmetric key on the fly using the master key is faster than calling a particular key for a destination from storage.

6.5 Attacks Countered by Multi-factor Authentication

Added to the attacks in Section 6.1, we have experimented with other hypothetical but realistic attacks from three different entry points in the smart grid context, as shown in Figure 6 for the authenticator device security. All these attacks were successful on the vanilla system. For these experiments, for the sake of flexibility and manageability, we utilized an open-source, software-based smart substation testbed [31] based on the IEC's reference model of a modernized substation [14].

First, we consider an attacker compromising a VPN interface, which is typically prepared for remote maintenance by vendors or as a backup channel for SCADA communication. In particular, we set up a VPN service with *shellshock* vulnerability [2], which allows a remote attacker to elevate privileges to execute the shell commands. Using the compromised VPN server as a stepping stone, the attacker's device, impersonating the ICS operator device, is able to inject malicious control commands to IEDs in the substation.

Second, we simulated a remote attacker via WAN connecting to the control center and substations. In other words, an attacker impersonates a control center to inject malicious commands. We have attached a bogus SCADA master, which does not have a legitimate key on it, to the gateway/RTU and issued ICS commands to IEDs.

The third attack simulates an attacker physically in the substation. Recall that, based on our threat models, an attacker can have access to the station level of the substation and may be able

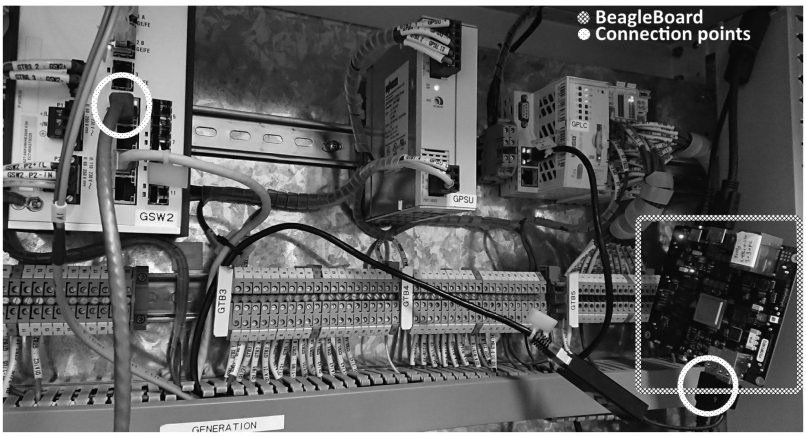


Fig. 9. F2-Pro integration into EPIC testbed [32]. On the right side, we highlighted one of the F2-Pro devices connected to a switch in the generation segment (GSW2) with the two highlighted cables in a bump-in-the-wire manner.

to plug his or her computer into the network or manipulate the HMI deployed there. All of these attacks are successful in executing the malicious control commands on IEDs without the proposed multi-factor authentication system in place.

We set up multiple ICS operator devices and authenticators assigned to different users. For the setup, we have given the passwords to be memorized by the users. Alternatively, the users could decide on a password and then upload their verification keys to the destination devices in a secure environment. All three attacks mentioned above are not possible after the implementation of our solution owing to the lack of valid cryptographic evidence.

Namely, in the first scenario, even if the attacker is successful in compromising the VPN server without the possession of a valid authenticator and field service device, the attack attempt is blocked. A similar argument can be made for the second case. In the third case, even if the stolen device still stores the cryptographic evidence generated in the past for some reason, a replay attack against the same or different target IED can be prevented by enforcing the scope and lifetime at the verifier. Even if the attacker learns the password of a user, steals his or her authenticator, and compromises an ICS operator device, as long as he or she cannot lay hands on the specific ICS operator device that is associated with the said user, he or she still cannot issue valid commands. In addition to compromising a user password, valid authenticator, and field service device, an attacker is required to place the device at the right network location in the expected ICS message delivery path.

6.6 Implementation

Figure 9 shows the deployment of the BITW device on the testbed. We have selected BeagleBoard-X15 (called BeagleBoard for short) [23]. It has been selected for its low cost and support for two Ethernet interfaces that enable bumping it in the wire. Hardware details can be found in [23].

6.6.1 The Bump-in-the-wire Device for F2-Pro. The BITW device is deployed for each smart grid device to be protected and is responsible for transparently intercepting incoming or outgoing packets of the protected device. F2-Pro implementation utilizes either *iptables* (in case F2-Pro only handles IP-based protocols such as IEC 60870-5-104 and IEC 61850 MMS) or *ebtables* (in case of IEC 61850 GOOSE), along with *NFQUEUE* to intercept packets of interest and pass them to the F2-Pro

deployed in the user space. The F2-Pro module is responsible for parsing an incoming packet to extract the accompanying cryptographic evidence, if any; verifying and/or generating cryptographic evidence according to the algorithm discussed in Section 3; and compiling an outgoing packet. If the cryptographic evidence is not valid or any other issue is found, the packet is dropped. In future work, it is possible to implement F2-Pro also in kernel space for better performance.

6.6.2 The Authenticator Device. We have implemented the authenticator discussed in Section 5 on the Android platform and performed our tests and measurements using a Samsung Galaxy Note 8 (processor: Exynos 9 Octa 8895). For QR code generation, we have employed the Zxing Library [33]; for the NFC tag generation, we have employed the CardEmulation Library [34]; and to read the generated tag at the ICS operator device we have employed ACR122U USB NFC Reader [35]. The application can be seen in Figure 8. With the setup button, the application randomly generates and stores a key offline. Then, given the memorized password as input by the user, the application generates a setup code. This code is salted per destination to be stored as the corresponding verification keys (AT) at the destination nodes. We can do the salting at the authenticator device producing many codes if there are multiple destinations since our solution supports data transfer employing both NFC and QR code methods. Note that these methods of communication, even though they may be less accessible, are one-way communication channels. If a two-way communication channel between the authenticator and the client is established (e.g., Bluetooth connection, a wired connection), it makes it possible for an adversary to compromise the authenticator remotely and/or reach the secret key stored. Once the setup is established, a user can start using the authenticator immediately to generate the codes by entering the password, choosing the scope of the command (we chose read/write requests, configurations commands, and update files as examples) that will be issued by the ICS operator device to acquire the one-time password. As in Section 5.1, the said one-time password is to be used as *Initiate* output, to be *extended* with the command issued at the ICS operator device.

6.7 Compatibility Testing

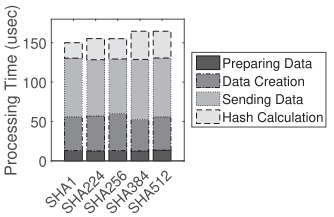
We tested our devices on the **Electric Power and Intelligent Control (EPIC)** testbed [32]. EPIC is a smart power grid testbed (open to external access), which includes generation, transmission, microgrid, and smart home components. Each of the four segments of EPIC has its PLCs, IEDs, and communication systems in a fiber-optic ring network. The EPIC testbed utilizes IEC 61850 MMS and GOOSE protocols as summarized in [36]. The demo of both the F2-Pro and its authenticator working on EPIC is shown in [25].

For the compatibility testing, we connected our F2-Pro devices into the generation segment (as seen in Figure 9) of the testbed for adding and verifying the cryptographic evidence as defined in Section 3. We demonstrated compatibility with the smart grid devices and network in the testbed for IEC 61850 MMS and GOOSE communication.

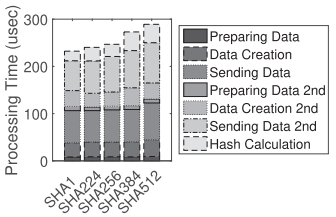
As with most ICSs, the EPIC testbed has a fixed topology and expected routing paths for messages. In systems with dynamically updated routing paths, certain path verification checks can still be performed with F2-Pro. If the nodes securely share their dynamically updated routing tables with each other, the destination device can verify if the message followed the right path for the given routing setting.

6.8 Evaluation

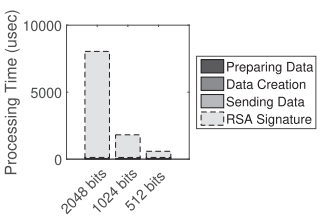
We first evaluate the core F2-Pro functions and then the two-factor authentication. Our evaluation focuses on F2-Pro's time overhead, including the time spent on cryptographic calculations, other packet-processing operations, and the end-to-end delay when deploying our implementation in a



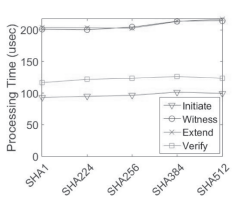
(a) Processing time of F2-Pro Initiate operation under different hash algorithms.



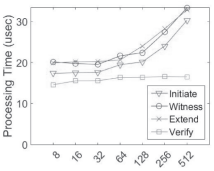
(b) Processing time of F2-Pro Witness operation under different hash algorithms.



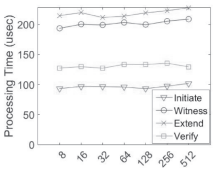
(c) The time required for RSA based signatures when initiating for different key sizes.



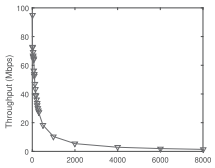
(d) The overall time spent for each F2-Pro operation under different hash functions.



(e) The time spent at the hashing step for each F2-Pro operation for different key sizes (in Bytes).



(f) The overall time spent for each F2-Pro operation for different secret key sizes (in Bytes).



(g) The effect of computation (in usec) per packet on the embedded device to the throughput.

Fig. 10. Time overhead measurements.

smart-grid-like network environment. We vary the hash functions and key sizes used by F2-Pro and compare it with the overhead of the public-key encryption scheme of RSA. We also report the message size overhead of F2-Pro and the throughput of our implementation. All results are the average of 1,000 runs.

Time Spent on Cryptographic Hash Calculations and Other Packet-processing Operations: Figure 10(a) and Figure 10(b) show the breakdown of the time spent for each operation during *Initiate* and *Witness*, respectively. The performance overhead for *Extend* is similar to Witness and thus omitted to save space. As can be seen, the time spent on the cryptographic hash operation is very short (i.e., less than 40 microseconds even for HMAC construction with SHA512), even when implemented on a low-cost embedded device [23]. In fact, it only constitutes a small fraction of the overall packet processing latency. Other necessary operations (such as preparing, creating, and sending the data packets) take significantly longer time than the cryptographic operations. Note that *Initiate* takes less time than *Witness*. This is because *Initiate* only processes a packet once, while Witness needs to receive/process/send a packet twice—first for the incoming packet to its associated smart grid device, and again for the processed packet leaving the associated device. In comparison, Figure 10(c) shows the overheads for signing a packet using RSA signatures. We have employed the OpenSSL library [37] to sign the same amount of data that we require in our setting. With the key size of 2,048 bits—which is the minimum of the ones considered secure—and with SHA256 as the digest function, the signing operation alone takes 8 ms and dominates the other packet processing operations. Thus, it is obvious that they are not suitable for use in time-stringent operations (e.g., those requiring an end-to-end delay of a few milliseconds). Figure 10(d) shows the time spent for all the four core operations by F2-Pro on BeagleBoard [23]. As the figure shows, the time overheads of F2-Pro operations remain low regardless of the hash function employed. Figure 10(e) shows the time spent by F2-Pro for the hashing operations for the cryptographic evidence or the verification using different key sizes. In our implementation, the AT is stored by the destination F2-Pro device, so the time spent for verification does not change with the key size. The other three operations compute ATs on the fly using the secret keys, and

Table 3. Extra End-to-end Latency Introduced by F2-Pro

no hop	1 hop	2 hops	3 hops	4 hops
<0.7ms	<1.7ms	<2.6ms	<3.5ms	<4.5ms

hence the time consumed increases with the key size. Figure 10(f) shows the overall time spent by F2-Pro to perform each operation for different key sizes. Again, there is little increase in the overall time overhead with the increase of the secret keys utilized. Overall, F2-Pro's latency is low enough when paired with practically secure key size and hash function.

Communication Overhead: The communication overhead is very minimal at 320 bits (h: 256 bits in the case of SHA256 and ts: 64 bits) per packet, and most importantly, its size remains the same regardless of the hop count.

Throughput Measurements: Figure 10(g) shows throughput measurements of F2-Pro also on a BeagleBoard, using *iperf* over TCP with message size 1,514 Bytes and with different computational latency added. The *Initiate* operation takes 140 to 160 usec as shown in Figure 10(a). Therefore, as seen in Figure 10(g), the F2-Pro on BeagleBoard can support 39.5 MBits/s at a source F2-Pro. And the *Witness* operation takes 280 to 300 usec, shown in Figure 10(b); therefore, the BeagleBoard can support 27 MBits/s at an intermediate device. This translates to 3,420 packets per second for the *Initiate* operation and 2,337 packets per second for the *Witness* operation. The computation time needed for the *Verify* operation and to support multicast can be longer, but F2-Pro can still sustain about 866 packets per second throughput when each packet incurs an extra 1 ms of computation time. In addition, our packet throughput measurement is a conservative lower bound since the packet sizes in both MMS and GOOSE are usually smaller than 1,514 Bytes. Therefore, the throughput of F2-Pro on BeagleBoard is sufficient for typical traffic intensity.

End-to-end Latency Overhead: To measure the end-to-end latency overhead introduced by F2-Pro, we set up a controlled environment using virtual machines for multiple hops. The total time overhead introduced by F2-Pro is shown in Table 3. The overhead is measured as the difference between the original end-to-end latency of the system without F2-Pro devices and that of a system where an F2-Pro device is attached to each device along the path. While the no-hop case only involves F2-Pro devices *initiating* and *verifying*, in the other cases the intermediate nodes are witnessing. The overall latency for no hop, which corresponds to the most latency-stringent scenarios, is low enough to meet the 2 ms message delivery time requirement. The cases with more than two hops are for SCADA communication and the time stringency is not as strict as those with one or no hops (see Table 1), and the added latency is not significant.

6.8.1 Multi-factor Authentication.

Generating an OTP: For a single destination generating an OTP takes between 300 and 800 usec depending on the processor activity of the phone at the time of computation. When OTPs for multiple destinations are computed, each computation will be done at faster speeds. For instance, it takes around 10 ms to calculate OTPs for 20 destinations. While the computation takes negligible amounts of time, considering that the authenticator will be used by a human, if the QR code transfer method is chosen, then it takes the application 1.7 sec to generate a QR code. On the other hand, generating the NFC tag takes 10 ms.

Regarding the readability of the QR code, we tested OTP up to 10 destinations, where the authenticator transfers 256 bits per destination. Even with the camera implemented on a commodity laptop, the corresponding QR code displayed on a full-HD smartphone screen was readable without any issue. While there are other similar implementations, a conventional QR code can carry 2,953 Bytes of data. If more data are to be transferred, then multiple QR codes may be generated; however, this would be troublesome for the user. On the other hand, our other option that employs

Table 4. Verification Time and Its Effect on the BITW Device Bandwidth

Method Used	No Hop		Three Hops	
	Verification Time (μs)	Bandwidth (Mbps)	Verification Time (μs)	Bandwidth (Mbps)
No user authentication	40	67	160	40
Username/password authentication	80	56	200	36
Our multi-factor authentication	80	56	200	36
RSA-2048 based multi-factor authentication	340	26	450	19

NFC does not have such limitation since as long as the device touches the NFC reader, we can transfer multiple rounds of data seamlessly.

We tested the setup, including the authenticator running on an Android phone, ICS operator device (a field service device) with the OTP reader module implemented on a commodity laptop, and a verifier module (on BeagleBoard) on the EPIC testbed [32]. The EPIC testbed includes IEDs and PLCs that are compliant with IEC 61850 standards. We ran the IEC 61850 client module on a laptop, which is used as a field service device, and evaluated a case where the field service device with the authenticator can send a control command to a target PLC. The bump-in-the-wire verifier module was placed in front of the PLC to perform authentication before the command reaches the destination PLC (see Figure 9).

Verification at the Destination and Bandwidth: Table 4 shows the verification times at the BITW device before the destination under different user authentication settings. No hop is the case where the ICS operator device (e.g., a field service device) is directly connected to the switch to which the destination devices are connected through the BITW devices. This example represents the shortest provenance chain. The three-hop case, on the other hand, represents the blue dashed line example in Figure 6. There are two main inferences regarding Table 4. First, both a simple username/password authentication and our multi-factor authentication increase the provenance chain length by one, causing almost the same verification overhead at the destination. Second, if the verification time is higher (e.g., a public key scheme), it affects the throughput of the BITW device before the destination drastically. For instance, even though the verification overhead of the RSA signature scheme is considerably lower than signing, it still constitutes most of the verification process, even when verifying a provenance chain.

7 RELATED WORK

We present the related work under three subsections. First, we explore the techniques to facilitate provenance verification, and then, we probe the bump-in-the-wire approach and conclude the section with the related work and concepts for multi-factor authentication.

7.1 Provenance Verification

Several schemes employ public key infrastructure and ameliorate it for path authentication with aggregate signatures [38, 39]. Aggregate signature enables multiple senders to sign different messages without increasing the signature size. To reduce computation and communication complexity, techniques such as signature amortization [40] and the space-efficient techniques of aggregate signatures [41, 42] have been proposed. However, under stringent latency constraints in ICSs, they are no longer options. Due to latency constraints, the IEC 62351-6 standard [43] recommends MAC-based solutions [44, 45] for message authentication, but these schemes do not focus on message provenance. Therefore, we use the aggregate MACs formally introduced by Katz and Lindell [6] as a primitive in our construction. [46, 47] provide certain public key guarantees with tradeoffs; however, they also lack efficient extension for provenance verification. Path verification in the smart

grid context was explored in [48], but it focused on demand response services, which are less latency stringent. Certain schemes use blockchain to provide data provenance [49, 50] in energy systems but lack a fast message provenance scheme.

7.2 Bump-in-the-wire Approach

BITWsecurity solutions have been proposed for ICS systems, as they enable easy deployment and updating of security mechanisms without requiring major upgrades or replacements on existing ICS devices [18]. Closely related to our work are BITW solutions for implementing confidentiality and integrity protection, such as [18–20]. [18, 19] deploys BITW devices at each end of the communication, which are responsible for adding security metadata (e.g., MAC) at the sender side and verifying and stripping it at the other end. While our focus is on the provenance of ICS messages, ours also adopts a similar deployment model to maintain compatibility with legacy ICS devices. Unfortunately, all of these do not consider end-to-end path verification in a multi-hop network. In ICSs, multi-hop communication is common and often involves protocol translation or checking middleboxes. Thus, we aim to provide provenance checking, including path verification.

7.3 Multi-factor Authentication

The main factors of user authentication used today are what a user has (e.g., a hardware token, a sim card to send an SMS) and what a user is (e.g., fingerprint, iris) as an addition to the memorized username and password [51]. Asking the user for a shared secret is another common practice (e.g., a pre-decided photo from a list of photos), but it is not comparable to other methods regarding the added security provided. While many of the above applications increase the bar for an attacker, they have generally been implemented by employing a trusted third party. Some examples are a server issuing a smart card [52], a server knowing all shared information between a smartphone and itself so that necessary information can be generated to be verified at the server [53], and the server generating and sending the code to the user [54].

It is a common understanding that the conventional username/password authentication alone is not considered secure [55–57]. This has led to several commercial solutions for two-factor authentication, such as RSA's SecurID [58] that works with SMS, e-mail, hardware, and software tokens; Google Duo [59] that works via U2F [60] USB device, phone calls, and mobile applications; and Symantec VIP [61] that uses hardware tokens, smartphones, and biometrics.

Kogan et al. proposed a scheme based on Lamport's one-time password authentication scheme [62], which has evolved in time under the name S/Key [63]. In their design, a central server is considered to verify all authentication requests. The authors, therefore, place emphasis on requiring no secrets stored on the server; however, since in an ICS setup the verifiers are fully distributed and an attack on the server that can expose all second factors for all users is not the main concern, the added overheads are not justifiable in our case. Our focus is on a quickly verifiable and thus symmetric-key-based (unlike public key-based approaches [64, 65]), and a multi-factor authentication mechanism that first does not depend on any third party or centralized online entity and second works in harmony with our provenance verification scheme tailored for an ICS environment.

8 CONCLUSION

In this work, we have proposed a practical provenance verification solution for industrial control systems. We have provided the formal definitions and cryptographic proofs. Our solution protects ICSs against commonly encountered attacks by restricting the attacker's entry points to the system and ensuring that the messages go through in-place security mechanisms. Our solution is legacy compliant and supports multi-factor authentication for field service personnel. We demonstrated

it in a controlled environment [24] and a smart power grid testbed [25], with a sub-millisecond overhead per network hop.

ACKNOWLEDGMENTS

We would like to express our gratitude to Mohammad Etemad and Alptekin Küpçü for the valuable discussions.

This research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme.

REFERENCES

[1] Ertem Esiner, Daisuke Mashima, Binbin Chen, Zbigniew Kalbarczyk, and David Nicol. 2019. F-Pro: A fast and flexible provenance-aware message authentication scheme for smart grid. In *2019 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm'19)*. IEEE, 1–7.

[2] Symantec Security Response. 2014. ShellShock: All you need to know about the Bash Bug vulnerability. Retrieved June 8, 2018, from https://www.symantec.com/connect/blogs/shellshock-all-you-need-know-about-bash-bug-vulnerability

[3] Kim Zetter. 2016. Inside the Cunning, Unprecedented Hack of Ukraineâ€™s Power Grid. http://www.wired.com/2016/03/inside-cunning-unprecedented-hack-ukraines-power-grid/

[4] Luc Moreau, Paul Groth, Simon Miles, Javier Vazquez-Salceda, John Ibbotson, Sheng Jiang, Steve Munroe, Omer Rana, Andreas Schreiber, Victor Tan, and Laszlo Varga. 2008. The provenance of electronic data. *Communications of the ACM* 51, 4 (2008), 52–58.

[5] Utku Tefek, Ertem Esiner, Daisuke Mashima, and Yih-Chun Hu. 2022. Analysis of message authentication solutions for IEC 61850 in substation automation systems. (Accepted for publication). In *IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm'22)*. 1–7.

[6] Jonathan Katz and Andrew Y. Lindell. 2008. Aggregate message authentication codes. In *Topics in Cryptology (CT-RSA'08): The Cryptographers' Track at the RSA Conference 2008, Proceedings*. Springer, 155–169.

[7] Theodore J. Williams. 1994. The Purdue enterprise reference architecture. *Computers in Industry* 24, 2–3 (1994), 141–158.

[8] Peter Maynard, Kieran McLaughlin, and Berthold Haberler. 2014. Towards understanding man-in-the-middle attacks on IEC 60870-5-104 SCADA networks. In *Proceedings of the 2nd International Symposium on ICS & SCADA Cyber Security Research 2014*. BCS, 30–42.

[9] Yi Yang, Kieran McLaughlin, Timothy Littler, Sakir Sezer, Eul Gyu Im, Z. Q. Yao, B Pranggono, and H. F. Wang. 2012. Man-in-the-middle attack test-bed investigating cyber-security vulnerabilities in smart grid SCADA systems. In *International Conference on Sustainable Power Generation and Supply (SUPERGEN'12)*. IET, 1–8.

[10] Yao Liu, Peng Ning, and Michael K. Reiter. 2011. False data injection attacks against state estimation in electric power grids. *ACM Transactions on Information and System Security (TISSEC)* 14, 1 (2011), 13.

[11] National Electric Sector Cybersecurity Organization Resource (NESCOR). 2013. Electric sector failure scenarios and impact analyses.

[12] Joe Slowik. 2019. Crashoverride: Reassessing the 2016 ukraine electric power event as a protection-focused attack. *Dragos, Inc* (2019).

[13] Robert M. Lee, Michael J. Assante, and Tim Conway. 2016. Analysis of the cyber attack on the ukrainian power grid. *USA: Electricity Information Sharing and Analysis Centre (E-ISAC)*. (2016).

[14] IEC TC57. 2015. IEC 61850-90-2 TR: Communication networks and systems for power utility automation - part 90-2: Using IEC 61850 for the communication between substations and control centres. In *International Electro Technical Commission Std.*

[15] Jared Mraz and Keith Gray. 2015. Demonstrating the flexibility provided by GOOSE messaging for protection and control applications in an industrial power system. In *Proceedings of the 42nd Annual Western Protective Relay Conference*.

[16] IEEE Power and Energy Society. 2004. IEEE standard communication delivery time performance requirements for electric power substation automation.

[17] Department of Energy. 2010. Communications Requirements of Smart Grid Technologies. Retrieved June 8, 2018, from https://www.energy.gov/gc/downloads/communications-requirements-smart-grid-technologies

[18] Patrick P. Tsang and Sean W. Smith. 2008. YASIR: A low-latency, high-integrity security retrofit for legacy SCADA systems. In *IFIP International Information Security Conference*. Springer, 445–459.

[19] Andrew K. Wright, John A. Kinast, and Joe McCarty. 2004. Low-latency cryptographic protection for SCADA communications. In *International Conference on Applied Cryptography and Network Security*. Springer, 263–277.

[20] John Henry Castellanos, Daniele Antonioli, Nils Ole Tippenhauer, and Martín Ochoa. 2017. Legacy-compliant data authentication for industrial control system traffic. In *Proceedings of the Conference on Applied Cryptography and Network Security (ACNS'17)*. DOI: http://dx.doi.org/10.1007/978-3-319-61204-1_33

[21] Mehdi Sabraoui, Jeffrey Hieb, Adrian Lauf, and James Graham. 2019. Modeling and machine-checking bump-in-the-wire security for industrial control systems. In *International Conference on Critical Infrastructure Protection*. Springer, 271–288.

[22] Jonathan Katz and Yehuda Lindell. 2020. *Introduction to Modern Cryptography*. CRC Press.

[23] 2022. BeagleBoard-X15. Retrieved May 17, 2022, from https://beagleboard.org/x15

[24] 2022. Message Authentication and Provenance Verification for Industrial Control Systems. YouTube. Retrieved November 25, 2022, from https://youtu.be/aOLKykAe3Xg/

[25] 2022. F-Pro and its authenticator on EPIC. YouTube Retrieved November 25, 2022, from https://youtu.be/PfHEcSXJaKQ/

[26] Ludovic Piètre-Cambacédès and Pascal Sitbon. 2008. Cryptographic key management for SCADA systems-issues and perspectives. In *2008 International Conference on Information Security and Assurance (ISA'08)*. IEEE, 156–161.

[27] Roger M. Needham and Michael D. Schroeder. 1978. Using encryption for authentication in large networks of computers. *Communications of the ACM* 21, 12 (1978), 993–999.

[28] Dorothy E. Denning and Giovanni Maria Sacco. 1981. Timestamps in key distribution protocols. *Communications of the ACM* 24, 8 (1981), 533–536.

[29] Martin Abadi and Roger Needham. 1996. Prudent engineering practice for cryptographic protocols. *IEEE Transactions on Software Engineering* 22, 1 (1996), 6–15.

[30] Gavin Lowe. 1995. An attack on the Needham-Schroeder public- key authentication protocol. *Information Processing Letters* 56, 3 (1995), 131–133.

[31] Prageeth Gunathilaka, Daisuke Mashima, and Binbin Chen. 2016. Softgrid: A software-based smart grid testbed for evaluating substation cybersecurity solutions. In *Proceedings of the 2nd ACM Workshop on Cyber-physical Systems Security and Privacy*. ACM, 113–124.

[32] iTrust, Singapore University of Technology and Design (SUTD). 2018. Electric Power and Intelligent Control (EPIC) Testbed. Retrieved February 12, 2019, from https://itrust.sutd.edu.sg/testbeds/electric-power-intelligent-control-epic/

[33] 2019. Zxing Library. Retrieved March 4, 2019, from https://github.com/zxing/zxing/

[34] Google LLC, Open Handset Alliance. 2019. CardEmulation Library. Retrieved March 4, 2019, from https://developer.android.com/reference/android/nfc/cardemulation/CardEmulation/

[35] Advanced Card Systems Ltd. 2019. ACR122U USB NFC Reader. Retrieved March 4, 2019, from https://www.acs.com.hk/en/products/3/acr122u-usb-nfc-reader/

[36] Ahnaf Siddiqi, Nils Ole Tippenhauer, Daisuke Mashima, and Binbin Chen. 2018. On practical threat scenario testing in an electric power ICS testbed. In *Proceedings of the Cyber-physical System Security Workshop (CPSS'18), Co-located with ASIACCS*. DOI: http://dx.doi.org/10.1145/3198458.3198461

[37] John Viega, Matt Messier, and Pravir Chandra. 2002. *Network Security with OpenSSL: Cryptography for Secure Communications*. O'Reilly Media, Inc.

[38] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. 2003. Aggregate and verifiably encrypted signatures from bilinear maps. In *Advances in Cryptology (EUROCRYPT'03)*, Eli Biham (Ed.). Springer, Berlin, 416–432.

[39] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. 2006. Sequential aggregate signatures and multisignatures without random oracles. In *Advances in Cryptology (EUROCRYPT'06)*, Serge Vaudenay (Ed.). Springer, Berlin, 465–485.

[40] Meiyuan Zhao, Sean W. Smith, and David M. Nicol. 2005. Aggregated path authentication for efficient BGP security. In *Proceedings of the 12th ACM Conference on Computer and Communications Security*. ACM, 128–138.

[41] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. 2003. A survey of two signature aggregation techniques. *RSA Cryptobytes* 6, 2 (2003), 1–10.

[42] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. 2003. Aggregate and verifiably encrypted signatures from bilinear maps. In *Advances in Cryptology (EUROCRYPT'03): International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 416–432.

[43] International Electrotechnical Commission. 2020. Power systems management and associated information exchange-Data and communications security-Part 6: Security for IEC 61850.

[44] S. M. Suhail Hussain, Shaik Mullapathi Farooq, and Taha Selim Ustun. 2019. Analysis and implementation of message authentication code (MAC) algorithms for GOOSE message security. *IEEE Access* 7 (2019), 80980–80984.

[45] Mikel Rodríguez, Armando Astarloa, Jesús Lázaro, Unai Bidarte, and Jaime Jiménez. 2018. System-on-programmable-chip AES-GCM implementation for wire-speed cryptography for SAS. In *2018 Conference on Design of Circuits and Integrated Systems (DCIS'18)*. IEEE, 1–6.

[46] Ertem Esiner, Utku Tefek, Hasan S. M. Erol, Daisuke Mashima, Binbin Chen, Yih-Chun Hu, Zbigniew Kalbarczyk, and David M. Nicol. 2022. Lomos: Less-online/more-offline signatures for extremely time-critical systems. *IEEE Transactions on Smart Grid* 13, 4 (2022), 3214–3226.

[47] Utku Tefek, Ertem Esiner, Daisuke Mashima, Binbin Chen, and Yih-Chun Hu. 2022. Caching-based multicast message authentication in time-critical industrial control systems. In *IEEE Conference on Computer Communications (IEEE INFOCOM'22)*. IEEE, 1039–1048.

[48] Daisuke Mashima, Ulrich Herberg, and Wei-Peng Chen. 2014. Enhancing demand response signal verification in automated demand response systems. In *2014 Innovative Smart Grid Technologies Conference (ISGT'14)*. 1–5.

[49] Jianbin Gao, Kwame Omono Asamoah, Emmanuel Boateng Sifah, Abla Smahi, Qi Xia, Hu Xia, Xiaosong Zhang, and Guishan Dong. 2018. GridMonitoring: Secured sovereign blockchain based monitoring on smart grid. *IEEE Access* 6 (2018), 9917–9925.

[50] Cristina Alcaraz, Juan E. Rubio, and Javier Lopez. 2020. Blockchain-assisted access for federated smart grid domains: Coupling and features. *Journal of Parallel and Distributed Computing* 144 (2020), 124–135.

[51] Jesper Mikael Johansson, Darren Ernest Canavor, Daniel Wade Hitchcock, and Bharath Kumar Bhimanaik. 2018. Approaches for providing multi-factor authentication credentials. (Jan. 9 2018). US Patent 9,864,852.

[52] Guomin Yang, Duncan S. Wong, Huaxiong Wang, and Xiaotie Deng. 2008. Two-factor mutual authentication based on smart cards and passwords. *Journal of Computer and System Sciences* 74, 7 (2008), 1160–1172. DOI: http://dx.doi.org/10.1016/j.jcss.2008.04.002

[53] F. Aloul, S. Zahidi, and W. El-Hajj. 2009. Two factor authentication using mobile phones. In *Proceedings of the IEEE/ACS International Conference on Computer Systems and Applications (AICCSA'09)*. 641–644. DOI: http://dx.doi.org/10.1109/AICCSA.2009.5069395

[54] Google LLC. 2022. [Google] About 2-Step Verification. Retrieved March 7, 2022, from https://support.google.com/accounts/answer/180744?hl=en&ref_topic=1099588/

[55] Joseph Bonneau and Sören Preibusch. 2010. The password thicket: Technical and market failures in human authentication on the web. In *WEIS*.

[56] Lee Mathews. 2022. File with 1.4 Billion Hacked and Leaked Passwords Found on the Dark Web. Forbes. Retrieved March 7, 2022, from https://www.forbes.com/sites/leemathews/2017/12/11/billion-hacked-passwords-dark-web/#38f0abf521f2/

[57] Alistair Charlton. 2022. iCloud accounts at risk of brute force attack as hacker exploits "painfully obvious" password flaw. IBTimes. Retrieved March 7, 2022, from https://www.ibtimes.co.uk/icloud-accounts-risk-brute-force-attack-hacker-exploits-painfully-obvious-password-flaw-1481623/

[58] RSA Security LLC. 2022. RSA SecurID Suite. Retrieved March 7, 2022 from https://www.rsa.com/products/securid/

[59] Google LLC. 2022. GoogleDuo. Retrieved November 21, 2022, from https://duo.com/product/trusted-users/two-factor-authentication/

[60] Fast IDentity Online (FIDO) Alliance. 2022. U2F (open authentication standard). Retrieved March 7, 2022, from https://www.yubico.com/solutions/fido-u2f/

[61] Gen Digital Inc. 2022. Symantec VIP. Retrieved March 7, 2022, from https://vip.symantec.com/

[62] Leslie Lamport. 1981. Password authentication with insecure communication. *Communications of the ACM* 24, 11 (1981), 770–772.

[63] Neil Haller. 1995. The S/KEY one-time password system. *Request for Comments: 1760*.

[64] Ertem Esiner and Anwitaman Datta. 2016. Layered security for storage at the edge: On decentralized multi-factor access control. In *Proceedings of the 17th International Conference on Distributed Computing and Networking*. ACM, 9.

[65] Ertem Esiner and Anwitaman Datta. 2019. Two-factor authentication for trusted third party free dispersed storage. *Future Generation Computer Systems* 90 (2019), 291–306.

Received 28 November 2022; revised 16 May 2023; accepted 25 June 2023

Safe Maintenance of Railways using COTS Mobile Devices: The Remote Worker Dashboard

TOMMASO ZOPPI, Dept. of Mathematics and Informatics, University of Florence, Florence, Italy
INNOCENZO MUNGIELLO, R&D Department of Rete Ferroviaria Italiana - RFI, Afragola (Naples), Italy
ANDREA CECCARELLI, Dept. of Mathematics and Informatics, University of Florence, Florence, Italy
ALBERTO CIRILLO, R&D Department of Rete Ferroviaria Italiana - RFI, Afragola (Naples), Italy
LORENZO SARTI, Dept. of Mathematics and Informatics, University of Florence, Florence, Italy
LORENZO ESPOSITO, R&D Department of Rete Ferroviaria Italiana - RFI, Afragola (Naples), Italy
GIUSEPPE SCAGLIONE and SERGIO REPETTO, R&D Department of Rete Ferroviaria Italiana - RFI, Osmannoro (Florence), Italy
ANDREA BONDAVALLI, Dept. of Mathematics and Informatics, University of Florence, Florence, Italy

The railway domain is regulated by rigorous safety standards to ensure that specific safety goals are met. Often, safety-critical systems rely on custom hardware-software components that are built from scratch to achieve specific functional and non-functional requirements. Instead, the (partial) usage of Commercial Off-The-Shelf (COTS) components is very attractive as it potentially allows reducing cost and time to market. Unfortunately, COTS components do not individually offer enough guarantees in terms of safety and security to be used in critical systems as they are. In such a context, *RFI* (Rete Ferroviaria Italiana), a major player in Europe for railway infrastructure management, aims at equipping track-side workers with COTS devices to remotely and safely interact with the existing interlocking system, drastically improving the performance of maintenance operations. This paper describes the first effort to update existing (embedded) railway systems to a more recent cyber-physical system paradigm. Our Remote Worker Dashboard (RWD) pairs the existing safe interlocking machinery alongside COTS mobile components, making cyber and physical components cooperate to provide the user with responsive, safe, and secure service. Specifically, the RWD is a SIL4 cyber-physical system to support maintenance of actuators and railways in which COTS mobile devices are safely used by track-side workers. The concept, development, implementation, verification, and validation activities to build the RWD were carried out in compliance with the applicable CENELEC standards required by certification bodies to declare compliance with specific guidelines.

CCS Concepts: • **Computer systems organization → Embedded and cyber-physical systems**; *Dependable and fault-tolerant systems and networks* • **Security and privacy** → *Systems security*;

Additional Key Words and Phrases: Safety, security, SIL4, railway, CPS, track-side maintenance, mobile devices, CENELEC, COTS

This work has been funded by RFI - Rete Ferroviaria Italiana S.p.A.

Authors' addresses: T. Zoppi, A. Ceccarelli, L. Sarti, and A. Bondavalli, Dept. of Mathematics and Informatics, University of Florence, Viale Morgagni 65, 50134 Florence, Italy; emails: {tommaso.zoppi, andrea.ceccarelli, lorenzo.sarti, bondavalli}@unifi.it; I. Mungiello, A. Cirillo, and L. Esposito, R&D Department of Rete Ferroviaria Italiana – RFI, Via Arena 1, 80021 Afragola (Naples), Italy; emails: {i.mungiello, a.cirillo, lo.esposito}@rfi.it; G. Scaglione and S. Repetto, R&D Department of Rete Ferroviaria Italiana – RFI, Osmannoro, Via Curzio Malaparte 50145, Florence, Italy; emails: {g.scaglione, se.repetto}@rfi.it.

2378-962X/2023/10-ART25
https://doi.org/10.1145/3607193

ACM Reference format:
Tommaso Zoppi, Innocenzo Mungiello, Andrea Ceccarelli, Alberto Cirillo, Lorenzo Sarti, Lorenzo Esposito, Giuseppe Scaglione, Sergio Repetto, and Andrea Bondavalli. 2023. Safe Maintenance of Railways using COTS Mobile Devices: The Remote Worker Dashboard. *ACM Trans. Cyber-Phys. Syst.* 7, 4, Article 25 (October 2023), 20 pages.
https://doi.org/10.1145/3607193

1 INTRODUCTION

Critical systems, whose (mis) behavior may lead to fatalities, severe injuries, or major damage to the environment, must adhere to appropriate guidelines to ensure that specific non-functional requirements are met [1, 22, 25]. In particular, their design and realization must provide safety [1], *avoiding catastrophic consequences on the user(s) and the environment*, thus being able to mitigate, manage, and isolate potential failures. Several domains have strong and straightforward safety implications, such as avionics, automotive, and railways. In these domains, components and systems are typically associated to a ***Safety Integrity Level*** (**SIL** [2, 7]), which identifies both qualitative and quantitative classes for the safety of the target component or system. In addition, attackers may craft and conduct malicious activities to harm those systems and generate cascading (safety) issues. Examples include, but are not limited to, hijacking autonomous vehicles to remotely control braking [36], the infection of the Deutsche Bahn systems by the WannaCry virus [38], and the multiple security issues in avionics that have been reported in surveys [37]. Consequently, safety-critical systems also have to deal with *security* [26, 45, 46], as they are required to guarantee *availability for authorized actions only, confidentiality, and integrity* [1].

The railway domain is based on embedded hardware-software systems that were built decades ago, and that are being maintained and updated along the evolution of the applicable standards as the ***European Rail Traffic Management System/European Train Control System*** (**ERTMS/ECTS**, [24, 27]). Indeed, recent technologies offer many additional opportunities to improve control systems by remotely providing critical functionalities in a completely safe manner. Unfortunately, railway control systems are not generally willingly updated due to (i) the reluctance of authorities and (ii) the cost and effort to design, develop, implement, verify, and validate a system that must comply with CENELEC [3, 7, 8] standards and has to be approved by a certification body before installation and operation. Moreover, **Commercial Off-The-Shelf (COTS)** components that may meet specific needs at a cheaper price are usually designed without safety in mind, and therefore cannot be employed as they are. As a result, control systems in the railway domain work properly and are maintained efficiently, but do not fully exploit novel technologies**.**

RFI* (Rete Ferroviaria Italiana)**, the company which manages railway infrastructure in Italy, has undergone a process to renew the current procedure to maintain rails, connected devices, and actuators. Maintenance activities are currently planned and coordinated via a ***Worker Dashboard **(WD)**, which is located in the central offices of train stations and requires track-side workers to physically move there whenever they initiate and conclude each maintenance action. The WD shows the state of actuators (e.g., railway switches) and other devices in the station as a *synoptic* diagram, often referred to as *mimic panel*. After physical access to the WD, track-side workers phone the control room operators through a dedicated and protected phone line for any information which is not available on such synoptic. Moreover, to block transit of trains in the track segment or actuator to be maintained, the workers may need to move to a separate – albeit close

to the WD – room which contains the key cabinet. This cabinet contains switch-lock keys for each area or actuator in the station, which have to be removed by track-side workers before starting operations and re-inserted only when concluding maintenance.

To improve and optimize such operations, *RFI* is promoting the usage of COTS tablets and smartphones to (i) eliminate the need to physically access the WD and the key cabinet to acquire the physical token, (ii) provide remote access to the synoptic of the station to the track-side worker, and (iii) reduce the need of phone calls with the control room operator. It is worth mentioning that such an update will dramatically improve the efficiency of maintenance operations, minimize train delays and reduce workers' movements – which indeed have to be tracked [34] - between the train station and railways, with positive impact on the personal safety of the workers themselves.

To the best of our knowledge, this paper describes the first ongoing work that aims at updating existing (embedded) railway systems to a more recent cyber-physical system paradigm by using non-customized COTS components. The contribution of the paper mostly revolves around bringing mechanisms that are state-of-the-art for researchers into a real cyber-physical system that has critical safety requirements. In fact, companies are usually reluctant to upgrade their critical systems, because even a small update may require massive time and money investment for conceptualizing, implementing, and producing all the necessary documentation for certifying the system before deployment in its final environment.

As such, we introduce a SIL4 **Remote Worker Dashboard (RWD)** to support maintenance of railways, which connects and orchestrates physical components to provide the track-side worker with responsive, safe, and secure services, with clear economic benefits in the medium-long term period. According to the CENELEC EN50126 [7] lifecycle, we describe Concept, System Definition, Hazard and Risk Analysis, System Architecture, and some details on implementation and V&V activities, focusing on mechanisms to guarantee safety and/or security. We also explain how COTS mobile devices can be safely used in the RWD without customization, as it is instead common practice in other railway systems [28, 29], providing additional relevance and novelty of the concept and design of the RWD.

The paper is organized as follows: related works and applicable standards are presented in Section 2, while an overview of railway maintenance and the current procedures is described in Section 3. Section 4 provides details about the system concept and architecture, Section 5 presents the safety and security mechanism developed for the RWD system, while Section 6 elaborates on the updated maintenance procedure. Section 7 elaborates on relevant Verification and Validation activities, letting Section 8 to finally conclude the paper.

2 APPLICABLE STANDARDS AND RELATED WORKS

2.1 Safety Integrity Level

The CENELEC standards represent landmarks for the development of programmable electronics in the field of railway applications in Europe. Particularly, EN 50126 [7], EN 50128 [8], EN 50129 [21], and EN 50159 [3] specialize the general-purpose IEC61508 [2] for the railway domain: compliance with those standards guarantees a safe development of hardware-software systems for the railway domain. The likelihood that a system satisfactorily performs the required safety functions is typically called *safety integrity level* [2, 7]. When a product is developed using methods, tools, and techniques appropriate to a specific safety integrity, it is possible to claim that the product is a *Safety Integrity Level* "X" product [7]. In the railway domain [21], SILs range from SIL0 (no safety constraints exist), to SIL4, which is quantified as a probability of catastrophic failure lower than 10^{-9} per hour and is the reference level for most railway systems.

2.2 COTS Components and Safety-Critical Systems

Companies, researchers, and practitioners usually aim at reducing time-to-market and costs related to the development of the whole system. This may be achieved by adopting COTS hardware and software components [22, 23, 25] that interact with custom safety-critical applications, operating systems, or hardware to provide the desired functionalities. However, COTS hardware, boards, and components do not usually accomplish safety requirements and do not embed diagnosis strategies [17, 18] to timely identify unsafe states. Consequently, researchers and practitioners proposed multiple approaches based on wrappers [16], redundant hardware [15, 19], or diverse software [30], which can adequately manage COTS components in safety-critical systems.

2.3 Safety of Track-Side Workers

The safety of track-side workers is obviously one of the main concerns [46, 50, 52, 53] when planning maintenance activities in railways. However, "[...] the continuing requirement for people to go on to the track to place and remove red lamps and explosive detonators, as part of the arrangements for protecting engineering work on the railway", is an issue that the UK Rail Accident Investigation Branch – among others – acknowledges since decades [48].

Consequently, different strategies were proposed throughout the years and in different countries to provide a safe environment for track-side workers that can be quickly set up and disassembled once maintenance is over. Most solutions rely on proposing ad-hoc wearable devices [28, 47, 49] which went through rigorous risk assessment [51] processes before production. Each worker is equipped with a device that rings or shakes to alert the worker whenever a train is expected to pass by the area under maintenance. Noticeably, those systems need components installed in neighboring areas that observe railways through radars, lidars, GPS, webcams, and oscillometers to eventually trigger wearable devices. Wrapping up, wearable devices can enhance safety of track-side workers, but still require ad-hoc devices and the setup of train detectors in areas neighboring the maintenance site, which still require time and effort to be installed. Differently, [54] proposes the adoption of high-visibility clothing that allows workers to be visible to automatic object detectors even with scarce or adverse lighting conditions. Such an approach proves to be effective in building sites or docks, but it is not effective in railways as a train at cruising speed most likely will end up hitting the track-side worker due to the very long time it needs to stop.

Despite the relevant body of research, to the best of our knowledge, no SIL4 railway system realizes part of its safety functions through COTS smartphones and tablets. Mobile devices are often used in the railway domain, but always require ad-hoc development of safety-critical software and a rugged architecture [28, 29]. Differently, this paper presents a new step for the research and innovation in the railway domain as it presents a SIL4 system that safely orchestrates unsafe COTS components, without requiring any customization.

3 RAILWAY MAINTENANCE

Track-side workers have multiple responsibilities and have to fulfill different tasks to support the correct provision of railway services. Most of their work is carried out inside the stations, where the majority of physical actuators and almost all the software of railway ground systems (e.g., railway switches, semaphores, and crossings) are located. Workers may need to request several authorizations to block trains passing by the areas under maintenance [14]. This section describes the main actors involved in the maintenance of railways and the current procedure to conduct common maintenance operations.

3.1 Maintenance of Railways

Italian railways are managed by **RFI (Rete Ferroviaria Italiana)**, which administers and maintains approximately 2,200 stations and a total of 16,723 km (10,391 mi) of active lines, 45% of which

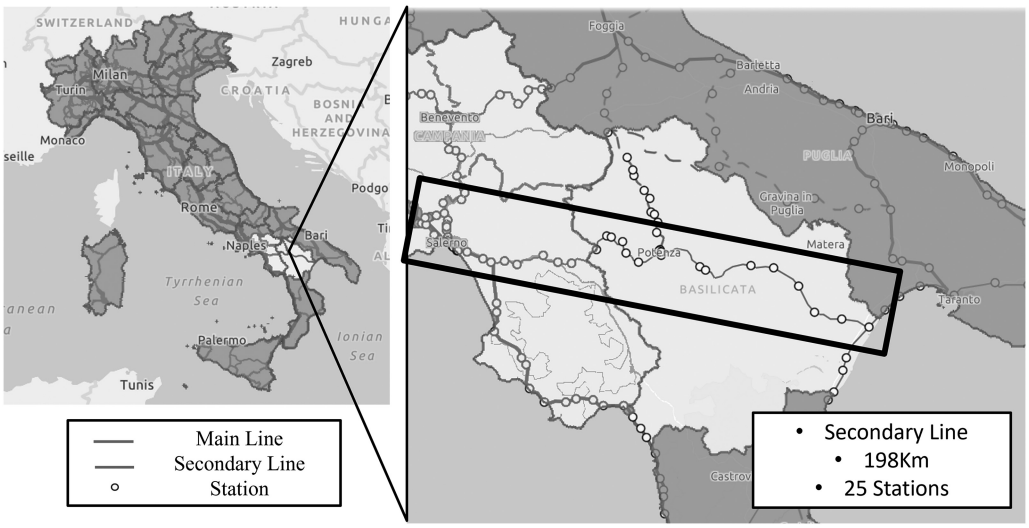


Fig. 1. Diagram of tracks, railways, and stations in Italy, highlighting a secondary line [44].

have double railway tracks [44]. The left of Figure 1 shows the current topology of the railway network, and allows identifying (i) stations, (ii) main lines, which have high traffic and good infrastructure quality, and (iii) secondary lines, which have less traffic and are responsible for connecting medium or small regional centers.

Maintaining such safety-critical and cyber-physical infrastructure poses mechanical and electrical challenges, but also has to guarantee the personal safety of track-side workers, customers, and personnel on trains, trying to minimize delays due to maintenance operations. For instance, the schedule of planned maintenance operations for a secondary line of approximately 200 km in length (e.g., see right of Figure 1, length of 198 Km), or rather the 1.18% of the overall extension of Italian railways [44], usually allocates between 450 and 500 operations per month. In one month of 2020, the following maintenance operations were planned, implemented, and decommissioned on a line similar to (the exact line cannot be revealed due to non-disclosure constraints) the one in Figure 1:

- Replacement (73/477, 15.3%), which aims at substituting, upgrading, or partially replacing components that may be degrading or too old.
- Periodic Inspection (134/477, 28.1%), which is planned periodically and aims at manually inspecting actuators or railways to identify potential mechanical or electronic issues.
- Reconditioning (65/477, 13.6%), or rather maintenance actions directed to restore a component to its original state without replacing.
- Generic Maintenance (92/477, 19.3%) and Generic Action (113/477, 23.7%) where no further details were provided by the infrastructure manager RFI.

All these 477 maintenance operations (i.e., 15.4 scheduled operations as daily average) require disconnecting one or more actuators or physical components from the railway network, temporarily preventing the transit of trains to guarantee personal safety of track-side workers. Consequently, reducing maintenance time does not only improve the throughput of those operations, but also – and more importantly - reduces potential delays due to train re-routing.

It is worth mentioning that 67% of these operations were scheduled at night-time (i.e., between 11 pm and 6 am), where the railway network is usually subject to a very low to absent traffic. Such

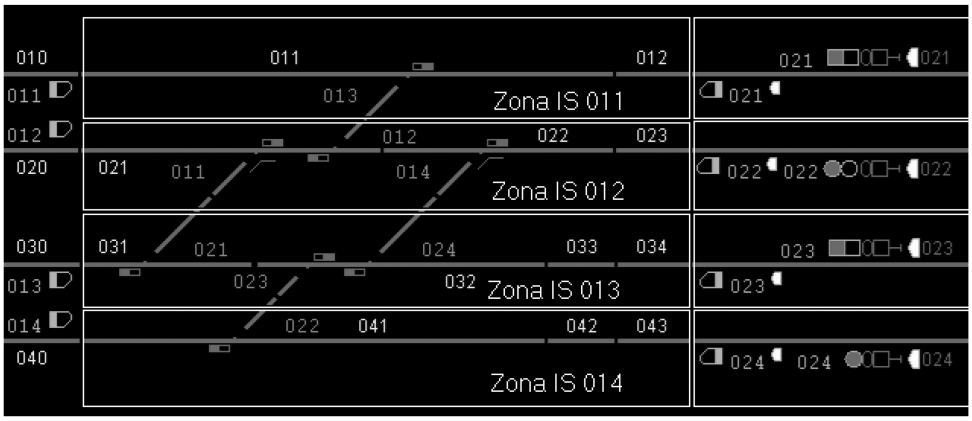


Fig. 2. Synoptic including rails, their interconnections, and several actuators and physical components.

policy allows conducting maintenance without impacting at all on train schedules. Additionally, *periodic inspections* and many *generic maintenance* operations are entirely carried out by visual inspections, thus requiring minimal maintenance time. Regardless, there were approximately five daily maintenance operations to be carried out in such secondary line during high-peak traffic hours. Roughly, we can estimate around 600 maintenance operations to be carried out daily in high peak hours in the whole infrastructure of Italian railways, which clearly motivates the need for an adequate support to optimize those operations.

3.2 Railway Stations, Synoptic and Track-Side Workers

Railway stations are usually partitioned into areas representing track segments, connected components, and actuators [14]. The state of those areas is usually depicted as a *synoptic* or mimic panel (see Figure 2), which reports on rails and components, labeled with numbers that are painted with different colors depending on their state. When an area, or an actuator, is under maintenance, its state is considered excluded, which is translated to temporarily disconnect a specific component from the railway network. Note that each area (e.g., "Zona IS" in Figure 2) is independent from the others: as a result, different actions can be simultaneously performed in two different neighbouring areas. For example, the area "Zona IS 011" in the figure may be excluded, while the neighbouring "Zona IS 012" may allow transit of trains.

Maintenance activities may be either planned or immediate, whenever facing unexpected issues. In both cases, as described in [14], track-side workers (simply *workers*, for brevity in the remainder of the paper) rely on the ***Worker Dashboard* (WD)**, which can be accessed only from the station *central office* with workers' credentials. The WD enables workers to (i) observe the synoptic to check the state of actuators (e.g., rail switches can be "straight" or "diverging"), or (ii) see details about train routing. Moreover, they can also (iii) use the WD to send *commands,* which are usually directed to change the state of actuators to exclude areas and avoid train routing, allowing workers to safely reach specific areas of the station that need maintenance. For a detailed list of requirements of the Italian WD, please refer to the directive [20].

3.3 The Maintenance Procedure

Maintenance procedures in Italian stations follow a two-step process to authorize commands. First, the worker asks for a given command using the WD; then, the command is forwarded to the control room of the station where the station manager authorizes or denies it. If this authorization

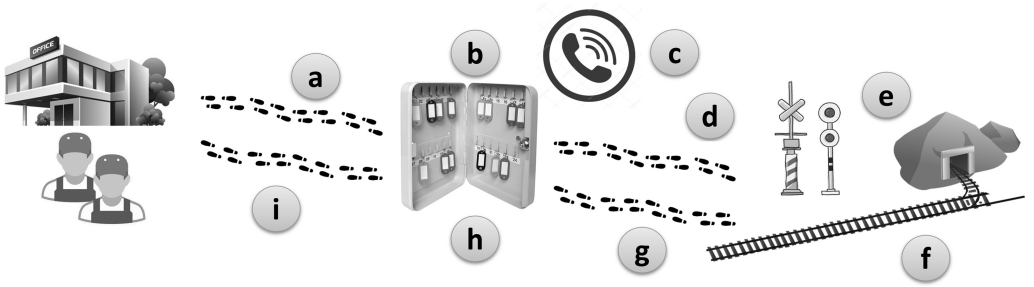


Fig. 3. Schema of the current maintenance procedure [14].

is granted, the worker may be asked to perform additional actions before starting maintenance. For example, whenever a worker has to exclude an area of the station, he/she is required to go to the WD, ask for the command, wait for the confirmation of the station manager, and, additionally, remove a switch-lock key corresponding to the area involved in maintenance from a key cabinet before starting operations.

As represented also in Figure 3, once the worker gets to the premise of the WD and the key cabinet (a), he has to authenticate to the local security personnel, then he has to turn the switch-lock key (b), call the station manager to confirm the maintenance operation (c), and turn the lock key again, removing it from the key cabinet (d). If the key corresponding to a given area is not in the key cabinet, such area is excluded from the railway network. At this stage, all the required authorizations have been granted and therefore workers can (e) reach the area under maintenance, and (f) perform the required actions. When the workers complete maintenance, they have to (g) go back to the key cabinet, (h) re-insert the key corresponding to the area and turn it on to match the initial position. This action re-connects the area to the network, completing maintenance and enabling trains to transit again in the area previously excluded. Finally, (i) workers can come back to the central office and are ready to handle further actions.

3.4 Limitations of the Current Maintenance Procedure

This procedure was adopted decades ago and still complies with the applicable railway standards, but there is room for innovation, mainly considering the following aspects:

- Access to the WD for simple operations such as viewing the synoptic requires the workers to physically move throughout the station between areas, actuators, and the central office every time.
- The central office may be physically far from the position of the workers and the maintenance site. This wastes workers time, since they have to move to get the required authorization.
- Time required for maintenance can be reduced, minimizing delays to train re-routing.

4 A REMOTE WORKER DASHBOARD

To tackle the issues above and update maintenance of railways, RFI and its academic partner defined a Remote Worker Dashboard (RWD) to be installed first in Italian stations and potentially applicable to other stations worldwide. This section summarizes the main items of our work regarding phases 1-4 of the CENELEC EN50126 [7] lifecycle, namely Concept, System Definition, Risk Analysis, and System Requirements. Those were cross-validated by a certification body that provided a SIL4-compliance certificate to CENELEC EN50126 lifecycle up to step 4 (included).

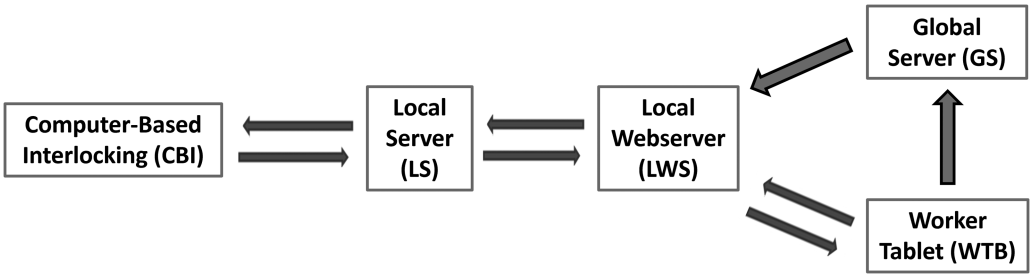


Fig. 4. Preliminary architecture of the RWD showing components and their inter-connections.

4.1 System Concept and Motivation

The RWD aims at updating the existing procedure and support systems by providing workers with a COTS tablet that displays the WD and allows a safe and secure interaction with the existing SIL4 interlocking systems when sending commands. RWD will allow workers to perform their daily operations without the necessity to physically go to the central office to ask for and execute commands. The RWD has analogous functionalities of the WD and can remotely interact with the existing interlocking machinery. Consequently, the RWD system as a whole has to conform to SIL4 [7] requirements. Since some of the components and communications are carried out in an uncontrolled and potentially hostile environment, security threats shall be accounted, as they may have severe impact on safety properties. In other words, the RWD system has to be safe, and its safety integrity should not be affected by possible security breaches or exploits of vulnerabilities.

4.2 System Definition

Functional requirements that guide a precise definition of the system overlap with the requirements of the already existing WD [20]. At this phase, the CENELEC lifecycle requires devising a preliminary system architecture that (i) serves as a baseline for the hazard analysis, and (ii) will be updated to include mitigations to identified hazards to constitute the final system architecture.

Overall, our preliminary architecture specification of the RWD ended up identifying five components, namely WTB, GS, LWS, LS, and CBI, which are depicted in Figure 4. The worker primarily relies on a tablet (Worker Tablet, WTB), which they use to gather information such as the synoptic and to send commands to the **Computer-based Interlocking (CBI)**. To such extent, we employ a webserver **(Local Webserver LWS)** specific for each station that exposes a web app to be accessed from the WTB through browsing. The worker can connect to the specific LWS through a Global Server GS which works as a single and unified access point and re-routes the worker to the LWS installed in the specific station they work in. Each LWS will execute commands asked by workers through WTB; therefore, it will safely communicate with the existing SIL4 CBI. Interfacing LWS and CBI is quite complex: therefore, we plan a **Local Server (LS)**, disconnected from the Internet but able to bridge the connection between CBI and LWS and provide additional control routines or support tasks e.g., check preconditions before applying commands.

4.3 Hazard and Risk Analysis

Hazard Analysis Techniques. The preliminary architecture specification allows conducting a ***preliminary hazard analysis*** **(PHA)** to identify and analyze *hazards* due to component failures, human mistakes, or attackers willing to damage the system, leading either to availability or safety issues. There are many different techniques for identifying criticalities and operability problems, ranging from *Checklists*, ***Fault Modes and Effects Analysis*** **(FMEA)** [6], ***Hazard and***

***Operability study* (HAZOP)** [5, 51] and ***Fault Tree Analysis* (FTA)**. All these methods output a hazard log, or rather a list of potential threats to the system.

Likelihood, Severity, and Risk. Regardless of the domain of interest, each item of the hazard log needs to be assigned to qualitative levels of *likelihood* and *severity* that help build the *risk* function. The standard CENELEC EN50126 [7] lists severities from best to worst as *Insignificant, Marginal, Critical*, and *Catastrophic*, to be combined with *Frequent, Probable, Occasional, Rare, Improbable*, and *Highly Improbable* likelihoods. Each possible couple of <severity, likelihood> is assigned in the standard [7] to a risk, which may be *Negligible, Tolerable, Undesirable*, or *Intolerable*. The resulting risk for each hazard drives the need to identify a mitigation that may (slightly) modify the system architecture and usually aim either at decreasing likelihood or at reducing severity to lower risk associated to hazards. As a result, the identification of mitigations to hazards contributes to building the final system architecture. At this stage, the PHA has to be updated to match the final system architecture, building the final **Hazard Analysis (HA)**.

Hazard Analysis of the RWD. The industrial partner and the consultant set up hazard meetings to identify potential threats to the system through brainstorming, checklists, and more importantly by systematically applying the HAZOP [5] methodology. For each critical function of the RWD system, e.g., "worker requests a command", the HAZOP methodology demands applying the following keywords:

- *Not*, or rather: what happens if the function is NOT executed?
- *More/Less:* what happens if the function is repeated MORE/LESS times than expected*?*
- *Part Of:* what happens if the function is only partially executed?
- *Early/Late:* what happens if the function is executed EARLIER/LATER than expected?
- *Other Than*, which points to execution of the function by means OTHER THAN expected, or with specific environmental conditions, that do not cover other options above.

As a result, potential hazards are identified in a structured way. In addition, we devise the root cause and consequences of each hazard, as well as likelihood, impact, and risk according to the categories above, and an explanation about why the residual risk is acceptable, or mitigations to be applied to make the risk tolerable.

Table 1 shows an extract of the Hazard and Risk Analysis of the RWD system.[1] The table reports five hazards related to different functionalities: H1–H4 were extracted from the PHA, while H5 only appears in the final HA as it involves a component that was added to mitigate threats in the PHA. From the top of the table, in column "hazard", a potential threat H1 could lie in dropped message due to a malfunction of the channel or the LTE network. The impact of this hazard may be catastrophic if the message that is dropped carries critical information such as "area cannot be freed due to trains expected to pass by". As mitigation, communication should be protected through a safe protocol (SSL/TLS itself is not enough) regulated by CENELEC EN50159 [3] that – amongst other characteristics – does not fail silent if a message is lost. Similarly, we have to be aware that WTB is a COTS tablet that is not built according to safety requirements. Therefore, the same message may be sent more than once (line H2 in Table 1) to LWS due to problems or congestion in the network adapter of WTB. This may also have catastrophic consequences if the duplicated message is an old synoptic that was previously cached, which shows an area as excluded, i.e., disconnected and safe to go for the worker, despite it being connected and ready for train routing. Here the adoption of a safe protocol is not enough even in the case of an EN50159-compliant [3]

[1]Note that mitigations to specific threats cannot be shared entirely due to non-disclosure agreements and to protect the Intellectual Property of the industrial partner (and funder) of this project.

Table 1. Items of the (Preliminary) Hazard and Risk Analysis of the RWD – HAZOP Methodology

#	Compo-nent(s)	Function	HAZOP Key	Hazard	Root Cause	Consequences	Likeli-hood	Impact	Risk	Mitigation
Preliminary Hazard Analysis										
H1	WTB, LWS	*Data exchange with LWS*	NOT/NO	Network packet/Message is dropped	*Channel Error*	A message is dropped. Possible loss of critical data or malfunctioning of procedure executed by the worker.	Probable	Catastrophic	*Intolerable*	Adoption of a Safe and Secure communication protocol, compliant to EN 50159 standard.
H2	WTB, LWS	*Data exchange with LWS*	MORE	Message is repeated	*Tablet Error*	Could be shown old and erroneous information (e.g., because of data caching) about critical commands (e.g., exclusion of an area).	Rare	Catastrophic	*Undesirable*	Duplicated information about commands (e.g., timestamp, actuator/component involved) must be shown in a replicated channel.
H3	WTB	*Command Request*	OTHER THAN	The worker selects a command, its execution get notified, but command actually not executed	*Malicious (Attack)*	Workers may take wrong decisions e.g., going to an area they think it is disconnected from the network, while it is not.	Probable	Catastrophic	*Intolerable*	Feedbacks about execution or failure of a command should be reported through two diverse and isolated channels.
H4	WTB	*Pressing Command Button on UI*	OTHER THAN	The worker presses the wrong button	*Worker Error*	The Worker input in the IMR System a command different from the one intended.	Occasional	Catastrophic	*Intolerable*	Each command needs additional confirmation by using also a replicated channel.
Final Hazard Analysis										
H5	WTB, WS	*Command Request*	OTHER THAN	The attacker compromises both personal devices of the worker at the same time	*Malicious (Attack)*	The attacker can act freely as a worker, asking for commands and confirming it through the both channels, with potentially catastrophic consequences.	Highly Improbable	Catastrophic	*Tolerable*	Not Needed: Tolerable risk.

protocol able to defend also against repetition attacks. Consequently, we figured out that we have to provide additional information to the worker (e.g., a timestamp to identify the "freshness" of the image) both on the WTB and on a replicated channel, which may be another device such as the personal smartphone of the worker. This mitigation would help also with the threat H3, which is caused by an attacker that installs malware on the tablet to change the content of the feedback message from LWS to harm the worker.

Similarly, it may happen that the touchscreen or the browser of the WTB do not function properly and send to the LWS the request of a command other than the one the worker wanted to execute. Also in this case, the usage of a replicated channel where LWS sends feedback by means of a device different from WTB reduces the likelihood of this hazard, which becomes "Highly Improbable" and results in a "Tolerable" risk.

Implementing Mitigations. The most frequent mitigations require all safety-critical information to be made redundant (e.g., relevant information in the synoptic will also appear as separate text in the webpages) for safety purposes, while a duplicated communication channel helps mitigating also security threats. To achieve these mitigations, we mimic an approach commonly used in online banking that requires the worker to insert in the WTB a **One-Time Password (OTP)** they get from a different device (the **Worker Smartphone WS**) to confirm the execution of a command or for other critical actions, e.g., Login. This provides coverage against failures (even malicious) of either WTB or WS devices: common mode failure/hacking still shows a Catastrophic impact in Table 1 (H5), but we consider the likelihood of this event to be Highly Improbable for this system, making the overall risk as Tolerable.

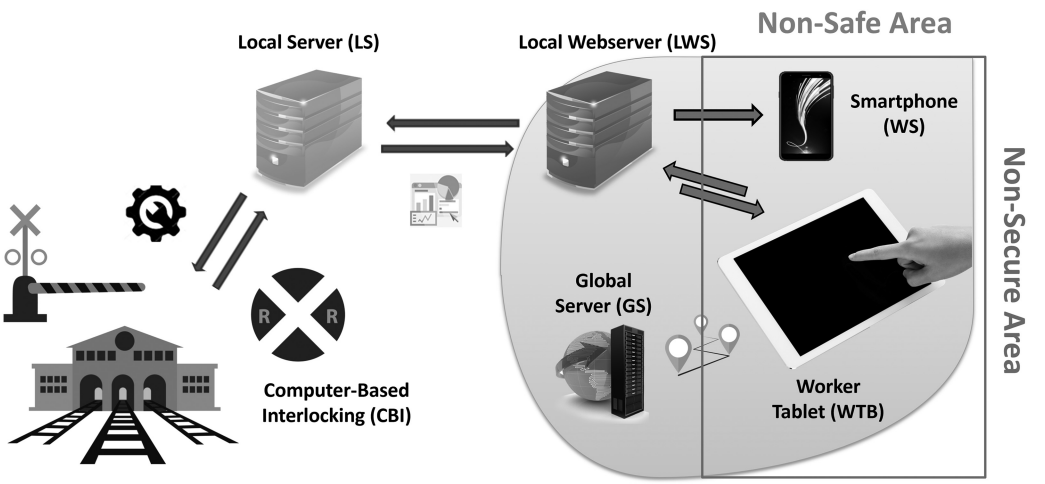


Fig. 5. Final architecture of the RWD. Components that lie in non-safe or non-secure areas do not have to comply with safety or security requirements.

4.4 System Architecture Specification

The mitigations identified during (P)HA allowed devising the final System Architecture that is reported in Figure 5. The left part of the picture depicts components (CBI, LS) that: (i) are located in a premise of the station whose access is severely controlled; (ii) are intrinsically safe as they are designed, developed, verified and validated as SIL4, and (iii) can rely on secured (wired) connections that shield them from network threats.

Instead, the right part of the picture represents devices and interconnections lying in a "non-safe area", where components are individually built without safety in mind. Therefore, they may independently fail, or may be subject to attacks, which may also impact communication channels. The non-safe region of Figure 5 embraces the global (web)server GS, the web server of the station (LWS), and two personal COTS devices available to the worker: a tablet (WTB) and a smartphone (WS). Indeed, GS and LWS are physically located in the station's premises and protected by a firewall to ensure security against network intrusions and therefore do not belong to the non-secure area.

Finally, it is worth pointing out that each worker is responsible of mobile devices WTB and WS and they have to immediately notify the IT department of the railway manager if either of the two devices is lost or unavailable. Insights on each of the six components of the RWD are provided below.

4.4.1 Computer-Based Interlocking (CBI). The Computer-Based Interlocking manages all the actuators such as railway switches, semaphores or level crossings in a semi-automatic way. It manages interlocking activities through SIL4 routines and control loops that periodically query all connected devices to check their state and (when possible) their health indicators. In the RWD system, the CBI can be accessed only by LS through a wired point-to-point channel that is managed through a safe [3] protocol.

4.4.2 Local Server(LS). LS is a SIL4 component running a Real-Time Embedded operating system with cyclic tasks. This is necessary to correctly interact with CBIs, which typically rely upon hard real-time OSs with cycles of either 350 or 500 milliseconds [40]. LS is in charge of: (i) managing workers' authentication (bridged by LWS); (ii) generating One Time Passwords through Pseudo-RNG with a periodical refresh of the seed and performing their verification; (iii) checking

the applicability of the commands asked by workers and – if applicable - forward them to the CBI; and (iv) query the CBI to get the state of the physical devices to build the synoptic. Note that LS is not directly connected to the internet; instead, it has two separate point-to-point connections (see Figure 5) protected by a safe and secure protocol to interact with CBI and LWS.

4.4.3 Local Web Server (LWS). LWS runs a commercial (i.e., Apache) webserver which exposes the user interface of the RWD through webpages. Moreover, LWS acts as a proxy between LS and WTB and is in charge of (i) bridging requests and feedback of user commands requested by the worker to LS and (ii) assembling web pages with information provided by LS and the synoptic, so that they can be accessed by the worker through WTB. The LWS is physically located in the premises of the station and protected by rigorous access control and a firewall to ensure security against physical and cyber intrusions. Additionally, the communication between LWS and WTB happens by means of a safe and secure protocol [3].

4.4.4 Global Server (GS). GS stores all information (e.g., IP addresses), needed to re-route workers to specific LWS. Each station has its own LWS and WS; instead, the GS works as a unified entry point to the system that asks the worker for a first login and redirects them to the LWS of the station they are interested in. GS hosts a web server that workers can use for a username-password login which, if successful, enables the worker to be redirected to the LWS of a station and complete login. The GS lies in the "Secure Area" in Figure 5 as it has security requirements (i.e., login activities), but it is not considered an intrinsically safe component.

4.4.5 Worker Smartphone (WS). WS is a COTS smartphone that shows a textual message received through a wireless LTE network. Whenever needed, LWS sends SMSs that contain different information which include – but are not limited to – the name of the station, the requested command, a timestamp (details in Section 5.3.2).

4.4.6 Worker Tablet (WTB). WTB is a COTS tablet the worker uses to remotely access the RWD through mobile network. The worker uses this COTS tablet in conjunction with the WS to (i) first navigate to the GS to be redirected to the correct LWS; (ii) authenticate to the LWS of the station the worker has to operate, and (iii) perform the actions needed for authorization, start, and decommission of a specific maintenance procedure.

4.5 Interactions between Synchronous and Asynchronous Components

In a generic railway system, collisions between trains are avoided by never allowing more than one train to occupy a track segment at any time. Regardless of the specific protocol to avoid collisions, which is implemented in the CBI, those actions have strict hard-real-time requirements that need to be fulfilled alongside with other safety requirements. Consequently, the RWD should have at least a SIL4 component (the LS) equipped with a cyclic hard-real-time operating system. LS directly interacts with the CBI and with all the machinery required for compliance to the railway standards whenever needed. However, the WTB should be able to occupy a track segment asynchronously, when workers complete a maintenance request.

This exposes the RWD system to a potential design issue: while the LWS and WTB communicate through a client-server asynchronous paradigm, the LS, the CBI, and other railway actuators execute a cyclic and synchronous exchange of information and completion of tasks. As in Section 4.4.3, the LWS is in charge of bridging communications between LWS and LS, where only the latter is a cyclic synchronous system. Therefore, LWS implements a communication module that, every cycle:

- forwards to LS any request that was sent to LWS by WTB in the last cycle, if any;
- sends an "I am alive" default message to LS if no request was received from the WTB.

Table 2. Summary of Communications Happening in the RWD Systems, and Protocols they Rely Upon

Communication Channel	CENELEC 50159 Network Cat.	Physical Transmission Medium	Application-Level Protocol	Type	Purpose
CBI - LS	*Cat.1 Closed Transmission System*	Wired	PVS	Synch	Communications of the RWD with the interlocking railway system
LS - LWS	*Cat.1 Closed Transmission System*	Wired	Custom over PVS	Synch	Exchange of requests and data to build webpages
LWS - WTB	*Category 3 – Open Transmission System*	Wireless	Custom over HTTPS	Asynch	Exposing webpages to the tablet
WTB - GS	*Category 3 – Open Transmission System*	Wireless	HTTPS	Asynch	Managing authentication of the user and routing to the correct LWS
LWS - WS	*Category 3 – Open Transmission System*	Wireless	HTTPS	Asynch	Sending OTPs to the WS

LS reads data from the network interface with LWS once every cycle and reacts accordingly. When the request from WTB is completed, LS sends a feedback to LWS, which updates the webpage to be shown on the WTB. In case no information is received by LS for a few consecutive cycles, the LS flags the communication channel with LWS as malfunctioning, and stops accepting any request that may come from this channel until restoration. Importantly, the payload of requests from LWS to LS complies to an application-level-protocol that specifies, among others (i) the ID of the worker and the WTB, (ii) the code of the request, and (iii) additional information whenever needed. Any request that does not comply with this protocol specification is discarded before being processed: exposing a restricted set of requests provides yet another mechanism to guarantee that LS deals only with requests it is able to manage.

5 INSIGHTS ON SAFETY AND SECURITY MECHANISMS

COTS devices as WTB and WS are not customized, rugged, or run with limited privileges and therefore are affected by a multitude of threats. Consequently, interactions with other components of RWD need to be strictly regulated to avoid unsafe behaviours or potential security breaches will not propagate to the system. The rest of this section provides insights on specific mechanisms we devised to mitigate safety and security threats.

5.1 Communications between Components

Communications between components of the RWD should comply with different requirements: therefore, different communication protocols are required for different channels (see Table 2). LWS provides data to WS through a regular LTE network, and then is accessed on the WS by using a commercial messaging app. WTB and GS communicate via a regular HTTPS (SSL/TLS) protocol as no particular safety requirement involves the Global Server GS.

All the other communications between components of the RWD should instead be carried out through safe (and often secure) protocols [3]. Interactions between LS and LWS, as well as between LS and CBI, are managed through the Vital Standard Protocol (***Protocollo Vitale* Standard - PVS** [9]). PVS is a lightweight protocol that stems from the Subset-037 of ERTMS ETCS *Euroradio* FIS [11] and Subset-098 of *RBC-RBC Safe Communication Interface* [10]. It adopts techniques and algorithms that have been identified among those highly recommended in EN50159 [3]. As

a consequence, *this protocol is becoming a de-facto standard for safety-related communications between cyber-physical components* in the Italian railways allowing interoperability of components produced by different manufacturers as Hitachi, Alstom and Bombardier. PVS [9] provides the *Session* and *Presentation* OSI layers over a TCP/IP transport stack and provides defences against CENELEC EN50159 [3] threats through mechanisms as *sequence number, safety codes* (extended CRC), *cryptography* (optional) through *AES* [12] and *AES-CMAC* [13], and a numerical counter that ensures "freshness" of the message called execution cycle.

Instead, communications between LWS and WTB are regulated by a custom protocol which provides safety by custom handshaking upon the standard HTTP with SSL/TLS commonly used for secured web communications. This custom protocol was deemed necessary since the PVS is meant to manage cyclic communications and does not adequately fit the client-server paradigm behind web-pages.

5.2 Safety Mechanisms

We list here safety mechanisms SaM1 to SaM6 that were employed to mitigate hazards alongside with measures to protect communications that we described in the previous section. Note that some of these mechanisms may also mitigate security threats.

5.2.1 SaM1 – Redundancy and Device Duplication (WTB and WS). Redundancy is a design pattern that is widely adopted when building safety-critical systems [41], even through diversity [42]. We apply redundancy by providing the worker with two personal COTS devices to execute commands, handling redundant information which may help to understand ongoing malfunctions. For example, a worker may see a command "exclusion of area 15" on a device, and "inclusion of area 15" on the other, detecting a malfunction and stopping operations.

5.2.2 SaM2 - Synchronization. After login, LS establishes a point-to-point connection with WTB through LWS. Then, we calculate and store the potential clock drift between LS and WTB to be used as offset when producing timestamps to be sent through web pages. This avoids misinterpretation of the "freshness" of the information (e.g., a synoptic) by the worker due to clock skew between LS and COTS devices, which may not rely on unified timings as NTP [43].

5.2.3 SaM3 – Static Webpages. Webpages generated by LWS using information received from the LS are designed to be defined server-side (PHP), easy to interpret and avoid client-side scripting (i.e., JavaScript) aside from showing current time, to minimize potential hazards due to running scripts on the potentially unsafe and insecure COTS tablets (WTB).

5.2.4 SaM4 – Generation of Images. LS gets the state of physical devices through CBI and builds the synoptic as a GIF/JPEG. Those images are then sent to LWS, which embeds them in the webpage with no further modifications. To mitigate some of the threats identified during HA, LS slightly modifies such image under specific circumstances by watermarking [31] heterogeneous information to be provided to the worker (details are not shared due to non-disclosure). Image generation is entirely performed by the SIL4 LS: therefore, the image is always generated safely, and its integrity is preserved thanks to the safe communications between LS, LWS, and WTB.

5.2.5 SaM5 – Command Filtering and Applicability Check. Once a worker selects a given area or actuator on the web app, the LS (through LWS) provides the worker with a list of applicable commands. Applicable commands are filtered by the LS according to the current state of the area/actuator: for example, it is not possible to disconnect a railway switch when it is already disconnected. Such filtering prevents the worker to ask for inapplicable commands. Nevertheless, the

applicability of commands is always cross-checked by CBI before actuation, which prevents the execution of malformed or non-valid commands and guarantees safety of those operations.

5.2.6 SaM6 – Revert Commands. Commands can be reverted (e.g., re-connecting an area that was previously disconnected) only by the worker that requested it at a first stage or by the station manager in the control room. This denies a worker to re-connect an area that was previously disconnected by another worker, who may still be acting there.

5.3 Security Mechanisms

We report here the security mechanisms of the RWD that pair and synergize with the safety mechanisms and the communication protocols discussed above.

5.3.1 SeM1 - Authentication. The worker should first authenticate to the RWD through a two-step process. First, the worker accesses GS, which asks a login with something the worker *knows*: a username and a password. This enables the worker to choose a station in which they are authorized to operate. Then, the worker is automatically redirected to the LWS of the specific station, where they have to provide an OTP to complete login. Such OTP is generated by LS and sent to the WS through LWS. Workers' credentials and the phone number of the WS are stored in the LS, allowing authentication to be entirely conducted server-side, delegating all these critical actions to the SIL4 LS.

5.3.2 SeM2 - OTP. Login or command requests have to be confirmed using a One-Time-Password. In those cases, the LS (i) generates an OTP through a state-of-the-art Pseudo-Random Number Generator, (ii) associates the OTP to the ID of the worker who requested the operation and, at the same time, (iii) builds a message containing the OTP and supporting information (details are not shared due to non-disclosure), and (iv) forwards the message to LWS alongside with a phone number, to allow the LWS to send such message to the WS as SMS. After the worker types such OTP, (v) WTB forwards such OTP to LWS and LS, allowing the latter to (vi) verify the compliance of the OTP typed by the worker with the one that was generated at step (i). Similarly, to SeM1, critical tasks such as generating and checking OTP are demanded to the SIL4 LS.

5.3.3 SeM3 – Command Feedback. When a command is correctly executed by the CBI, the LS prepares an updated GIF of the synoptic in which additional information is watermarked at random coordinates without overlapping with existing information in the image. The updated synoptic is sent to the WTB, and a message is simultaneously sent to the WS, allowing the worker to check compliance of those two items. Instead, if the command is *not* executed, LS transmits to the WTB a synoptic that *does not* contain additional information, and LS *does not* transmit anything to the WS. If the worker does not get any message on WS, he can assume that the command was not executed, even if malfunctions happen on WTB.

6 MAINTAINANCE PROCEDURE WITH RWD

The RWD calls for an update of the current maintenance procedure we describe below with the help of Figure 6, Figure 7, Figure 8, and Figure 9 and with a sequence diagram in Figure 10 that details the execution of a generic command.

6.1 Login and Station Selection

The worker performing maintenance on a station first connects with GS through WTB. The Global Server asks for login credentials (SeM1, Figure 6), allows the choice of the station, and redirects to the specific LWS (Station_1 in Figure 7) which completes the login asking for an OTP (SeM1, SeM2).

Once the address of the LWS is retrieved by GS, the WTB establishes a direct connection with the LWS performing the initial handshake needed for the safe and secure protocol and for clock

Fig. 6. Login page of the GS asking for Username and Password.

Fig. 7. Login page of the LWS, which completes the two-step login initiated by GS. It is completed by inserting the OTP the worker receives on the WS.

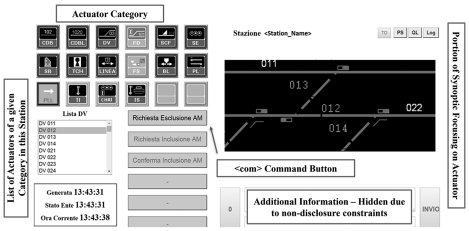


Fig. 8. Execution of a command: step 1 – selection of railway switch DV 012.

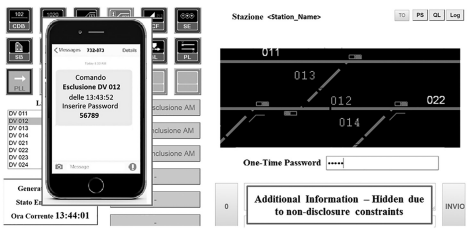


Fig. 9. Execution of a command: step 3,4 – exclusion. (disconnection) of railway switch DV 012.

synchronization (SaM2). If different workers operate in different areas of the same station simultaneously, each worker will have his own WTB connected with a dedicated point-to-point channel to LWS.

6.2 Check the Synoptic

Once the worker is authenticated to the LWS, he may want to query the synoptic of the station, ask for planned train routing and review planned maintenance operations. When LS grants login to the worker, it gathers data about the state of physical components of the station through CBI, assembles the required information, and sends the image (plus textual information about specific components, whenever applicable - SaM1) back to LWS, which delivers it to the worker through a webpage (SaM3).

6.3 Execution of Commands

Once logged in, the worker may ask to remotely execute commands. Let *com* be a command the worker wants to execute, e.g., temporarily disconnect an area from the railway network. The procedure to execute the command *com,* shown as sequence diagram in Figure 10, is orchestrated as follows:

(1) The worker selects an area or an actuator category (e.g., railway switch). Then he selects the specific area/actuator (e.g., railway switch DV 012) from a dropdown list; this shows (Figure 8) the portion of the synoptic focusing on the chosen actuator alongside with available commands (SaM5). To execute the command *com* the worker presses the “com” button that is shown in the middle of the page.
(2) LS receives the command.

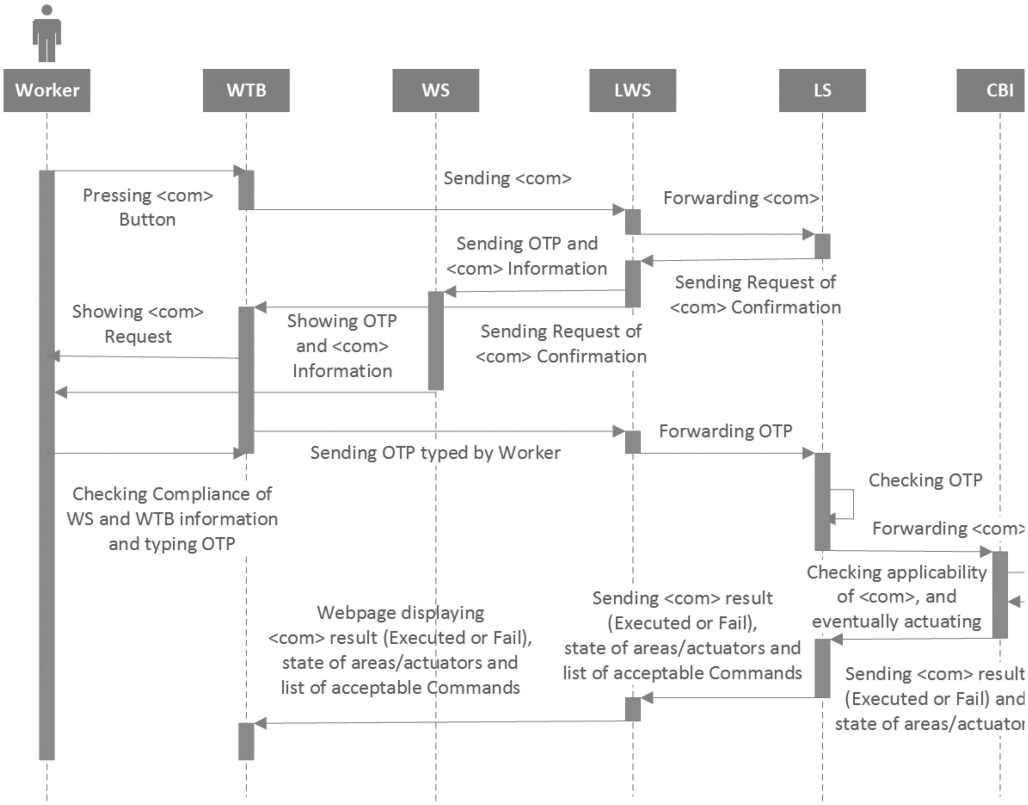


Fig. 10. Sequence diagram to execute a command <com>. Numbers match bullets in Section 6.3.

(3) LS replies via LWS: (i) to the WTB: an updated synoptic showing the area/actuator and the received command (SaM1, SaM4), and (ii) to the WS: an OTP plus additional information about the requested command (SeM2, Figure 9).
(4) The worker types the OTP received on WS in the WTB after checking compliance of additional information to confirm the command *com*.
(5) The OTP is received by LWS and forwarded to LS. After successfully verifying the validity of the OTP, LS (i) forwards the command to the CBI for actuation, and (ii) stores the requested command (and the feedback the CBI gives) in its internal storage for logging purposes.
(6) Then, LS prepares a feedback to the worker (SeM3), which is delivered through LWS. The worker may be asked to execute additional actions to confirm they correctly got the feedback, which helps mitigate specific threats.
(7) Ultimately, the RWD shows a webpage (similar to Figure 8) that contains an updated synoptic and potentially allows requesting other commands.

Various commands can be sent by the workers according to the same procedure, either if they want to re-connect the area that was previously disconnected, or if they want to disconnect a new one for the sake of maintenance. Note that *once a worker asked for disconnection of an area or an actuator, only that specific worker will be able to reconnect it (SaM6), avoiding critical personal safety problems.*

7 VERIFICATION AND VALIDATION ACTIVITIES

When developing a railway system, each phase of the development lifecycle must be conducted in compliance with CENELEC standards [3, 7, 8, 21]. Among these, EN50128 [8] specifies a set of guidelines that shall be followed for safety-related software. This translates for example into the adoption of adequate code quality metrics [32, 35], and coding standards (e.g., MISRA C [4], widely adopted in the safety-critical domain), as well as the selection and the execution of the proper **Verification and Validation (V&V)** activities. The EN50128 standard sets up best practices to be adopted to comply with a given SIL, even letting the system architect choose amongst many alternatives depending on their expertise or specific project requirements.

Specifically, V&V techniques [8] selected for the RWD are: (i) *Dynamic Analysis and Testing*, applied according to principles *Test Cases Boundary Value Analysis* and *Equivalence Classes and Input Partition Testing*; (ii) *Functional/black box testing*, performed to check the satisfaction of all the functional, safety and security requirements; (iii) *Traceability*, carried out through the filling of *traceability matrixes* to provide an immediate mapping between test case and requirement under test; (iv) *Test Coverage*, verified according to criteria *statement* and *compound condition* and, finally, (v) *Static Analysis*, performed through the use of software tool *Polyspace* [33], to verify the compliance of the source code with selected coding standard and metrics. Such activities have been performed for the LS. Other components LWS, WTB, GS, and WS instead follow specific SIL0 V&V activities, which include a subset of those mentioned above. Given the heterogeneity of the system components, different testing environments have been set up for verification and validation purposes. In particular, we used an ARM board equipped with the *FreeRTOS* [39] real-time operating system for the LS, Linux machines for GS and LWS, and Android devices for WTB and WS.

V&V activities, alongside system concept, architecture, and hazard analyses are currently been used to develop a *safety case*, which reports the compliance of the system to the selected coding standard and metrics, demonstrates the satisfaction of each requirement, and contains a user manual which describes how to deploy and use the RWD system in its operational environment.

8 CONCLUSIONS AND ONGOING WORKS

This paper presented a Remote Worker Dashboard (RWD) to enhance current maintenance of railway infrastructures. RWD allows reducing maintenance time, minimizing delays on train schedule, and increasing safety of track-side workers. The RWD will support workers in managing maintenance operations through safe and secure interaction with the existing Computer-Based Interlocking. Workers can take advantage of personal COTS mobile devices to interact with the system, with clear advantages in terms of safety and time needed to fulfil operations.

To the best of our knowledge, RWD is the first cyber-physical railway system that embeds COTS mobile devices as tablets and smartphones without requiring rugged hardware or special configurations thanks to a consistent and coordinated system engineering effort of both the system owner and the consultants.

We described the current maintenance procedures, motivating the need of an RWD to reduce time needed for such operations and to optimize the overall movements of track-side workers in the station. After recalling key items of the applicable CENELEC standards, we reported on the concept, the definition, the hazard analysis, and the final architecture of the RWD system providing insights about the most relevant safety and security-related aspects that allowed RWD to be certified as a SIL4-compliant system regarding the first four phases of the CENELEC lifecycle [21].

Current works are directed to complete the implementation and V&V of the system, which is currently targeting the LS component. Other components as the webserver LWS and all connected

devices were already exercised together and proposed to a group of track-side workers, who provided interesting feedbacks about usability without raising concerns regarding safety and/or availability of the RWD system as a whole. Once implementation, verification, and validation are completed, we will come back to the certification body to complete the assessment which is mandatory to install the RWD in Italian stations.

REFERENCES

[1] Algirdas Avizienis et al. 2004. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing* 1, 1 (2004), 11–33.

[2] IEC, IEC61508. 2010. 61508 functional safety of electrical/electronic/programmable electronic safety-related systems. *International Electrotechnical Commission* (2010).

[3] CENELEC, EN. 50159. Railway applications - Communication, signalling and processing systems - Safety-related Communication in Transmission Part 2 (2011).

[4] MISRA C:2012. Guidelines for the use of C in Critical Systems, 978-1-906400-11-8, Motor Industry Research Association (2013).

[5] International Electrotechnical Commission and Technical Committee 56. 2016. Hazard and operability studies (HA- Q6 641 ZOP studies): Application guide. IEC61882:2016.

[6] D. H. Stamatis. 2003. *Failure Mode and Effect Analysis: FMEA from Theory to Execution.* ASQ Quality Press.

[7] CENELEC EN 50126. 2017. Railway applications - The specification and demonstration of Reliability, Availability, Maintainability and Safety (RAMS) – part 1 (2017).

[8] CENELEC EN 50128. 2012. Railway applications - Communication, signalling and processing systems - Software for railway control and protection systems. (2012).

[9] D. Bertieri, A. Ceccarelli, T. Zoppi, I. Mungiello, M. Barbareschi, and A. Bondavalli. 2021. Development and validation of a safe communication protocol compliant to railway standards. *Journal of the Brazilian Computer Society* 27, 1 (2021), 1–26.

[10] UNISIG, ERTMS/ETCS RBC-RBC Safe Communication Interface, Subset-098, (2012)

[11] UNISIG, UNISIG ERTMS/ETCS - Euroradio FIS, Subset 037 (2016)

[12] Joan Daemen and Vincent Rijmen. AES proposal: Rijndael. 1999.

[13] Junhyuk Song et al. The AES-CMAC algorithm. *RFC* 4493, June, 2006.

[14] RFI – Worker Dashboard, RFI DTCDNSSS SR IS 14 000 C (07/2013).

[15] Alejandro Carlos, Torres-Echeverría, Sebastián Martorell, and H. A. Thompson. 2011. Modeling safety instrumented systems with MooN voting architectures addressing system reconfiguration for testing. *Reliability Engineering & System Safety* 96, 5 (2011), 545–563.

[16] P. Popov, L. Strigini, S. Riddle, and A. Romanovsky. 2001. Protective wrapping of OTS components. In *Proc. 4th ICSE Workshop on Component-Based Software Engineering: Component Certification and System Prediction*, Toronto.

[17] M. Serafini, A. Bondavalli, and N. Suri. 2007. Online diagnosis and recovery: On the choice and impact of tuning parameters. *IEEE Trans. on Dependable and Secure Computing* 4, 4 (2007), 295–312, 2007.

[18] G. Carrozza, D. Cotroneo, and S. Russo. 2008. Software faults diagnosis in complex OTS based safety critical systems. In *2008 7th European Dependable Computing Conference*. IEEE, 25–34.

[19] F. Di Giandomenico and L. Strigini. 1990. Adjudicators for diverse-redundant components. In *Proceedings Ninth Symposium on Reliable Distributed Systems*. IEEE, 114–123.

[20] RFI, Specifica dei Requisiti del Terminale Operatore (TO), codifica RFI DTC STS SR SR SS40 001 A, 2013.

[21] CEI EN50129, Railway applications - Communication, signalling and processing systems - Safety related electronic systems for signalling. (2004).

[22] Zeeshan E. Bhatti, Partha S. Roop, and Roopak Sinha. 2016. Unified functional safety assessment of industrial automation systems. *IEEE Transactions on Industrial Informatics* 13, 1 (2016), 17–26.

[23] C. Yang, C. Yang, T. Peng, X. Yang, and W. Gui. 2017. A fault-injection strategy for traction drive control systems. *IEEE Transactions on Industrial Electronics* 64, 7 (2017), 5719–5727.

[24] H. W. Lim, W. G. Temple, B. A. N. Tran, B. Chen, Z. Kalbarczyk, and J. Zhou. 2019. Data integrity threats and countermeasures in railway spot transmission systems. *ACM Transactions on Cyber-Physical Systems* 4, 1 (2019), 1–26.

[25] R. E. Bloomfield, P. Popov, K. Salako, V. Stankovic, and D. Wright. 2017. Preliminary interdependency analysis: An approach to support critical-infrastructure risk-assessment. *Reliability Engineering & System Safety* 167 (2017), 198–217.

[26] A. Khaled, S. Ouchani, Z. Tari, and K. Drira. 2020. Assessing the severity of smart attacks in industrial cyber-physical systems. *ACM Transactions on Cyber-Physical Systems* 5, 1 (2020), 1–28.

[27] Richard Bloomfield. 2006. Fundamentals of European rail traffic management system-ERTMS. (2006): 165–184.

[28] Andrea Ceccarelli et al. 2012. Design and implementation of real-time wearable devices for a safety-critical track warning system. *High-Assurance Systems Engineering (HASE), 14th Symp. on*. IEEE.
[29] Muhammad Mahtab Alam and Elyes Ben Hamida. 2014. Surveying wearable human assistive technology for life and safety critical applications: Standards, challenges and opportunities. *Sensors* 14, 5 (2014), 9153–9209.
[30] Gustav Dahll, Mel Barnes, and Peter Bishop. 1990. Software diversity: Way to enhance safety?. *Information and Software Technology* 32, 10 (1990), 677–685.
[31] S. Katzenbeisser and F. A. P. Petitcolas. 2000. *Digital Watermarking*. Artech House, London.
[32] Exida Consulting LLC, C/C++ Coding Standard Recommendations for IEC 61508, Version V1, Revision R2, 2011.
[33] Polyspace static analysis tool, MATLAB, (product description website) https://it.mathworks.com/products/polyspace.html.
[34] T. Wang, W. Wang, A. Liu, S. Cai, and J. Cao. 2018. Improve the localization dependability for cyber-physical applications. *ACM Transactions on Cyber-Physical Systems* 3, 1 (2018), 1–21.
[35] Jill Britton. (PERFORCE) - Programming Research, Which Software Quality Metrics Matter? Online at https://www.perforce.com/resources/qac/which-software-quality-metrics-matter. Accessed 11/7/2023.
[36] Andy Greenberg. 2013. Hackers Reveal Nasty New Car Attacks-With Me Behind the Wheel (Video - online). https://bit.ly/3lWRAIN.
[37] Michal Klenka. Major incidents that shaped aviation security. *Journal of Transportation Security* 12.1-2 (2019), 39–56.
[38] Deutsche Bahn attacked by Wannacry (online), https://www.railtech.com/digitalisation/2017/12/11/wannacry-virus-was-wake-up-call-for-railway-industry/.
[39] FreeRTOS reference manual: API functions and configuration options. *Real Time Engineers Limited* (2009).
[40] Zhuo Li et al. 2016. NDN-GSM-R: A novel high-speed railway communication system via named data networking. *EURASIP Journal on Wireless Communications and Networking* 2016, 1 (2016), 1–5.
[41] F. Flammini, S. Marrone, N. Mazzocca, and V. Vittorini. 2009. A new modeling approach to the safety evaluation of N-modular redundant computer systems in presence of imperfect maintenance. *Reliability Engineering & System Safety* 94, 9 (2009), 1422–1432.
[42] F. Di Giandomenico and L. Strigini. 1990. Adjudicators for diverse-redundant components. In *Proceedings Ninth Symposium on Reliable Distributed Systems*. IEEE, 114–123.
[43] D. L. Mills. 1991. Internet time synchronization: The network time protocol. *IEEE Transactions on Communications* 39, 10 (1991), 1482–1493.
[44] RFI – La Rete oggi (online) https://www.rfi.it/it/rete/la-rete-oggi.html.
[45] A. Ceccarelli, T. Zoppi, A. Vasenev, M. Mori, D. Ionita, L. Montoya, and A. Bondavalli. 2018. Threat analysis in systems-of-systems: An emergence-oriented approach. *ACM Transactions on Cyber-Physical Systems* 3, 2 (2018), 1–24.
[46] Carmen Cheh et al. 2019. Modeling adversarial physical movement in a railway station: Classification and metrics. *ACM Transactions on Cyber-Physical Systems* 4, 1 (2019), 1–25.
[47] S. Banerjee, M. Hempel, and H. Sharif. 2017. A review of workspace challenges and wearable solutions in railroads and construction. In *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*. IEEE, 91–96.
[48] Network Rail Urged to Eliminate "Victorian Methods of Protection" (online), https://rail.nridigital.com/future_rail_oct19/getting_on_track_with_safety_for_railway_workers.
[49] M. M. Alam and E. Ben Hamida. 2014. Advances in wearable sensor technology and its applications in mobile workforce's health monitoring and safety management. In *SPE Middle East Health, Safety, Environment & Sustainable Development Conference and Exhibition*. Society of Petroleum Engineers.
[50] S. Kliuiev, I. Medvediev, and N. Khalipova. 2020. Study of railway traffic safety based on the railway track condition monitoring system. In *IOP Conference Series: Materials Science and Engineering*. IOP Publishing 985, 1 (2020), 012012.
[51] N. Noorudheen, M. McClanachan, Y. Toft, and G. Dell. 2013. Keeping track workers safe: A socio-technical analysis of emerging systems and technology. *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit* 227, 5 (2013), 517–528.
[52] C. Baldry and J. Ellison. 2006. Off the rails: Factors affecting track worker safety in the rail industry. *Employee Relations*.
[53] J. M. Sanne. 2008. Framing risks in a safety-critical and hazardous job: Risk-taking as responsibility in railway maintenance. *Journal of Risk Research* 11, 5 (2008), 645–658.
[54] R. Mosberger, H. Andreasson, and A. J. Lilienthal. 2013. Multi-human tracking using high-visibility clothing for industrial safety. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 638–644.

Received 2 August 2022; revised 14 April 2023; accepted 29 June 2023

Investigating the Security of EV Charging Mobile Applications as an Attack Surface

KHALED SARIEDDINE and MOHAMMAD ALI SAYED, The Security Research Centre, Concordia University, Canada
SADEGH TORABI, Center for Secure Information Systems, George Mason University, USA
RIBAL ATALLAH, Hydro-Quebec Research Institute, Canada
CHADI ASSI, The Security Research Centre, Concordia University, Canada

The adoption rate of EVs has witnessed a significant increase in recent years driven by multiple factors, chief among which is the increased flexibility and ease of access to charging infrastructure. To improve user experience and increase system flexibility, mobile applications have been incorporated into the EV charging ecosystem. EV charging mobile applications allow consumers to remotely trigger actions on charging stations and use functionalities such as start/stop charging sessions, pay for usage, and locate charging stations, to name a few. In this article, we study the security posture of the EV charging ecosystem against a new type of remote that exploits vulnerabilities in the EV charging mobile applications as an attack surface. We leverage a combination of static and dynamic analysis techniques to analyze the security of widely used EV charging mobile applications. Our analysis was performed on 31 of the most widely used mobile applications including their interactions with various components such as cloud management systems. The attack scenarios that exploit these vulnerabilities were verified on a real-time co-simulation test bed. Our discoveries indicate the lack of user/vehicle verification and improper authorization for critical functions, which allow adversaries to remotely hijack charging sessions and launch attacks against the connected critical infrastructure. The attacks were demonstrated using the EVCS mobile applications showing the feasibility and the applicability of our attacks. Indeed, we discuss specific remote attack scenarios and their impact on EV users. More importantly, our analysis results demonstrate the feasibility of leveraging existing vulnerabilities across various EV charging mobile applications to perform wide-scale coordinated remote charging/discharging attacks against the connected critical infrastructure (e.g., power grid), with significant economical and operational implications. Finally, we propose countermeasures to secure the infrastructure and impede adversaries from performing reconnaissance and launching remote attacks using compromised accounts.

CCS Concepts: • **Security and privacy → Distributed systems security;**

Additional Key Words and Phrases: Electric vehicle charging, cyber-physical systems, security analysis, mobile application

This research was conducted and funded as part of the Concordia University/Hydro-Quebec/NSERC research collaboration project "Large-scale Integration of EVCSs into the Smart Grid: A Comprehensive Cyber-physical Study and Security Assessment." Grant reference: ALLRP 567144-21.

Authors' addresses: K. Sarieddine, M. A. Sayed, and C. Assi, The Security Research Centre, Concordia University, 1515 Saint-Catherine St W, Montreal, QC, H3G 1S6, Canada; emails: khaled.sarieddine@mail.concordia.ca, Mohammad.sayed@mail.concordia.ca, assi@encs.concordia.ca; S. Torabi, Center for Secure Information Systems, George Mason University, 10401 York River Rd, Fairfax, VA 22030, United States; email: storabi@gmu.edu; R. Atallah, Hydro-Quebec Research Institute, 1800 Bd Lionel-Boulet, Varennes, QC, J3X 1S1, Canada; email: atallah.ribal@hydroquebec.com.

2378-962X/2023/10-ART26 $15.00
https://doi.org/10.1145/3609508

ACM Reference format:
Khaled Sarieddine, Mohammad Ali Sayed, Sadegh Torabi, Ribal Atallah, and Chadi Assi. 2023. Investigating the Security of EV Charging Mobile Applications as an Attack Surface. *ACM Trans. Cyber-Phys. Syst.* 7, 4, Article 26 (October 2023), 28 pages.
https://doi.org/10.1145/3609508

1 INTRODUCTION

Climate change and increased greenhouse gas emissions are fueling society's embrace of a green technology mindset. Governments are diligently working on shifting the traditional transportation system to a smarter one by utilizing **Electric Vehicles (EVs)**. Many countries have already implemented policies to reach carbon neutrality, partly by adopting EVs, with an aim to reach 30% EV market share by 2030. For instance, China's plan to reach carbon neutrality resulted in a significant 141% increase in the deliveries of EVs in October 2021 [1]. Similarly, multiple federal/provincial governments in North America along with various European countries have set policies to increase EV adoption in the next decade while banning the sale of new gasoline-powered passenger cars in the near future [2, 3]. Consequently, there is a rapid deployment of **EV Charging Stations (EVCSs)** to match EV adoption. Furthermore, governments are actively investing in the EV charging infrastructure to address the lag of EVCSs with respect to the increasing demand caused by the shift toward EVs [4].

The leading automotive companies are investing in incorporating advanced technological features in EVs and the **Cyber-Physical System (CPS)** associated with them for remote management. The **cloud management system (CMS)** manages all operations and functionalities of the public EVCS. As a result of the need to commercialize the EV charging ecosystem and to drive the adoption of EV technologies, the EV charging mobile application established itself as a core component of the EV ecosystem [5]. Mobile applications are used by EV consumers to remotely control EVCSs and manage charging operations (e.g., start and stop charging sessions) through the CMS. They also provide various online functionalities such as checking availability and handling payments, to name a few. The importance of studying this component in a fine-grained manner lies in its ability to control the EVCS and its wide distribution among customers. The existence of multiple vendors, along with minimal efforts to standardize the development and deployment of EVCS components, has exposed the ecosystem and the interconnected critical infrastructure (e.g., power grid) to a wide range of remote attacks [6, 7]. Moreover, the massive adoption of EVs caused a compelling change in the transportation system and the power grid, not only increasing the load but also increasing the complexity of the ecosystem as a whole [8]. This adoption led to the rapid deployment of Internet-enabled EVCSs along with various supporting EV charging mobile applications. Moreover, the number of product and cross-product applications is increasing tremendously, where one mobile application might have access to more than one EVCS network (e.g., Flo, Blink, and EVgo), thus amplifying the impact by exposing more products and users simultaneously. It is worth highlighting that such extended remote functionalities provide adversaries with a new attack surface, which could be utilized to compromise the vulnerable mobile application and thus exploit the underlying EVCSs and their operation.

A number of studies focused on the security of the EV ecosystem by exploring the security of the firmware, the installed management systems, and the communication link [7, 9–12]. For instance, Nasr et al. [7] studied the security of the EVCS firmware and management systems and discovered 13 severe vulnerabilities. Furthermore, Antoun et al. [12] presented a detailed security analysis of the ecosystem while assuming a highly privileged attacker that can fake, intercept, inject, and modify messages. Alcaraz et al. [9] studied the security posture of the OCPP protocol, which is

the main protocol used to control EVCSs, and discovered its susceptibility to **Man-in-the-Middle (MITM)** attacks. Despite such efforts to explore the security of various components within the EV charging ecosystem, there is a lack of knowledge about the security posture of the existing EV charging mobile applications as an attacker entry point. Moreover, there is a lack of understanding about the extended attacker capabilities and attack implications when leveraging vulnerabilities across widely used mobile applications to perform large-scale coordinated attacks against various stakeholders and components within the EV charging ecosystem.

In this study, we focus on investigating the security posture of the EV charging mobile applications as an attack surface on the EV charging ecosystem and the underlying infrastructure. We are the first to systematically analyze the most widely used EV charging mobile applications that are utilized to manage EVCSs across various geographical locations. We focus on identifying design flaws in the interaction between the different components as it requires a deeper understanding of the ecosystem and advanced threat modeling. We utilize reverse engineering and code analysis techniques to get insight into the functionalities instilled in the EV charging mobile applications, along with the security measures implemented by these applications (e.g., bot detection libraries). Consequently, guided by the static analysis, we perform dynamic functionality analysis to test the interactions of these applications with other components while identifying their main implemented EV charging-related functionalities that can be triggered remotely (e.g., remote start and stop). Furthermore, we analyze the traffic exchanged between the EV charging mobile application and the CMS to understand their interactions. We also examined the exchanged information by breaking SSL using MITM monitoring to unravel encrypted interactions. Additionally, we utilize a deep understanding of the ecosystem and the available literature to complement our analysis of the EVCS-Cloud communication [7, 9, 13, 14].

Our analysis of the exchanged traffic/interactions allowed us to infer state transitions between the mobile application and other entities in the EV charging ecosystem. We found that the analyzed mobile applications lack adequate EV ownership verification. Moreover, most applications implement improper authorization for initiating critical functions (e.g., start charging), which only binds users with EVCSs without binding/authenticating users to the EVs. This is closely related to the lack of adequate EV ownership, which allows insecure initiation of a critical function such as EV charging/discharging. Indeed, we found a lack of proper ownership checking and improper authorization for critical functions (start and stop charging). While such vulnerabilities demonstrate the insecurity of the EV charging ecosystem, they also highlight the immaturity of the deployed software components, which hinders the advancement to reach the goals set by the industry. Furthermore, 29 out of 31 studied mobile platforms are susceptible to remote charging session hijacking while enabling mass (dis)charging attacks. Additionally, our analysis illustrates that 19 of the vulnerable EV charging mobile application platforms can be exploited to perform large-scale oscillatory load attacks on the connected infrastructure through the unauthorized remote start/stop charging capabilities. Consequently, we study the feasibility of synchronized remote charging attacks and their impact on the underlying infrastructure by studying transmission losses and generation costs, along with overloading and transmission line tripping and power grid stability. Finally, we provide recommendations to prevent remote attacks utilizing the EV charging mobile application. To this end, we frame the main contributions of this work as follows:

- To the best of our knowledge, we are the first to study the security posture of EV charging mobile applications as an attack surface against the power grid. We study the interactions of the different components and validate them on a real-time co-simulation test bed. Our findings demonstrate the insecurity of such mobile applications while highlighting design and implementation weaknesses that can be exploited to perform unauthorized operations on the underlying EVCSs.

- We leverage static and dynamic analysis techniques to analyze the implementation of EV charging mobile applications and their interactions with various components within the ecosystem. We utilize Finite State Machines to provide an abstract representation and model the interactions of the mobile applications with the cloud management system and the EVCS counterparts. Our analysis indicates several vulnerable interactions due to unverified user/vehicle ownership and improper authorization for critical functions. We demonstrate such attacks by showing successful proof-of-concept attack scenarios that exploit the design flaws we discovered.
- Given the feasibility of the identified vulnerabilities across widely used EV charging mobile applications, we investigate the implications of wide-scale remote attack scenarios by constructing synchronized botnet attacks that utilize the mobile applications as a recon to perform various unauthorized charging operations including large-scale voltage/frequency instability attacks on the power grid. We discuss practical attack implications against the stakeholders, precisely, and the connected power grid infrastructure. Finally, we propose design/implementation countermeasures to mitigate such attacks in the future.

The remainder of this article is organized as follows. In Section 2, we present background information and basic concepts related to the EV ecosystem. In Section 3, we discuss the analysis methodology. In Section 4, we discuss our findings in terms of identified interactions and vulnerabilities, along with attack feasibility. We discuss detailed attack implications against various stakeholders in Section 5. In Section 6, we discuss a mitigation framework along with security measures that will help defend against such exploitation before concluding the article in Section 7.

2 SYSTEM MODEL, RELATED WORKS, AND THREAT MODEL

The EV charging ecosystem is a cyber-physical system, composed of interacting hardware and software components. In what follows, we provide details about these components and their interactions.

2.1 System Model

The EVCS ecosystem incorporates multiple entities that collaborate and interact to provide a vital service to customers (individuals and businesses). It is the main enabler for EVs, which have been spreading rapidly due to governmental policies that have driven their adoption. The EVCS ecosystem consists of a cyber and a physical layer, as shown in Figure 1.

2.1.1 Cyber Layer. The Cyber Layer is composed of multiple software components coupled with the hardware/physical counterpart. The mobile applications are publicly available and distributed through application stores (Google Play and Play Store). These applications are needed by users to control EVCSs remotely and view EVCSs' status through their communication with the CMS. The mobile applications could be either operator specific (manage EVCSs belonging to one operator) or multi-operator (manage EVCSs of multiple operators). The multi-operator mobile applications were introduced to simplify the charging process and enable EV roaming among different operators without the need for operator-specific subscriptions. Consequently, based on the above distinction, the operator-specific mobile application communicates with the operator's CMS, whereas the multi-operator mobile application communicates with its owner's backend, which in turn forwards the requests to the respective operator's CMS using the **Open Charge Point Interface (OCPI)** [13].

The CMS plays an equally important role in the ecosystem since it provides API endpoints for the mobile application to communicate with the EVCS. Each operator has their own CMS that is responsible for reservation, scheduling, payments, management, monitoring, and so forth. The

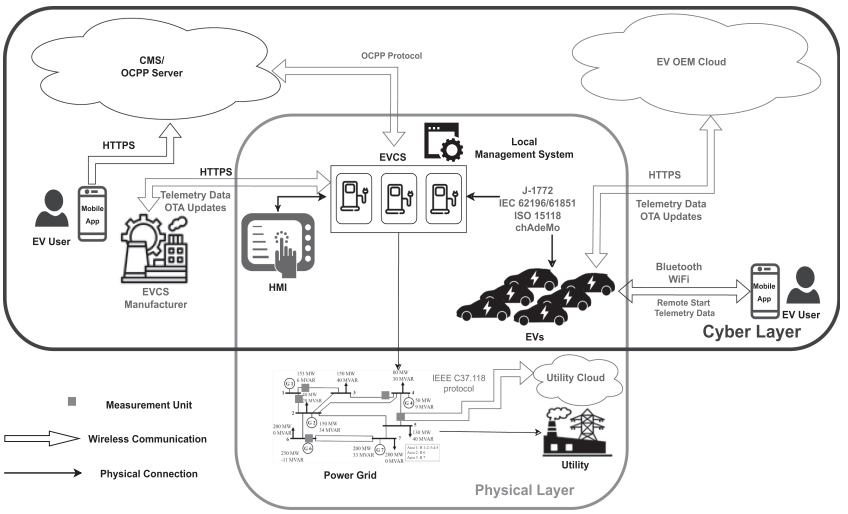


Fig. 1. Overview of the EV charging ecosystem and its interactions.

CMS is the most computationally capable component and it is considered the main driver of the ecosystem. However, to control EVCSs, the CMS communicates with the EVCS using the **Open Charge Point Protocol (OCPP)**.

The OCPP is the de facto standard that is utilized to manage EVCSs remotely. The OCPP defines two main roles, a lightweight client (EVCS) and a central server (CMS), which utilizes full-duplex communication over a TCP connection. The communication of the OCPP protocol is in the form of transaction functional blocks, where each entity requires a response to the initiated transaction. This standard is maintained and developed by an alliance of multiple companies working in the industry. Moreover, a connection is usually maintained between the EVCS and the original manufacturer, which helps in collecting logging information about the performance of the station. We have validated these interactions between the different components and the responsibility of each entity with our industrial partner Hydro-Quebec, a major North American utility.

Additionally, different cyber components exist but are not considered as a core component of the EVCS ecosystem and the connection is optional and does not necessarily always exist. For example, EVCSs maintain a connection with the **original equipment manufacturer (OEM)** to share telemetry data about the health of the device and its operational efficiency, which can be used later on to identify and enhance the product. Moreover, this connection is also used to push **over-the-air (OTA)** updates to ensure up-to-date software and the delivery of security patches in a timely manner. Moreover, the smart vehicles also are connected to their OEM cloud, which is used to share telemetry data and OTAs as well. Finally, there are mobile applications that are used by smart vehicle owners to start/stop the engine of a vehicle, Turn the air conditioning on, and so forth.

2.1.2 Physical Layer. The Physical Layer is represented by different entities. Namely, the EVCS hardware includes the human-machine interface that is used by the users to interact physically with the EVCS. After an EVCS is manufactured and bought by an operator, the manufacturer maintains a connection to push firmware updates remotely or can make the updates available online for the operator to manage the process. Moreover, the EVs have multiple hardware and software components including remotely accessible components such as an On-board Diagnostic Port and a CAN bus. The EV charging ecosystem was established to match the demand of EVs and their need

to charge. Two types of EVs are dependent on the EV charging ecosystem, which are the main foci when studying the security of the EV charging ecosystem: **Plug-in Hybrid EVs (PHEVs)** and Battery EVs [11, 15]. Other types of EVs such as Hybrid Electric Vehicles and Fuel Cell Electric Vehicles do not require external charging [16]. Consequently, EVs connect to the charging stations using various standards (SAE J-1772/J-2293/J-2847/J-2836, IEC 62196/61851, ISO/IEC 15118, and chAdeMO) [6, 7, 11], which are a part of the efforts to standardize communication in the EV charging ecosystem. Moreover, the IEEE 2030.5-2018 standard defines the application layer in the context of TCP/IP and facilitates the management of various utility functions related to end-user energy environments. These functions include demand response, load control, time-of-day pricing, distributed generation management, and electric vehicle integration. While this standard primarily focuses on the application protocol and its direct interaction, it also specifies the mechanisms for exchanging application messages, the specific content of those messages (including error messages), and the security features employed to safeguard the application messages. The application profile defined in this standard draws elements from various existing standards such as IEC 61968 and IEC 61850. Additionally, the **Combined Charging System (CCS)** is a communication protocol and charging standard used in EVs. It enables high-power charging by providing a single connector and communication interface for both AC and DC charging. It is widely adopted, and the CCS allows EVs to negotiate charging parameters and access a network of charging stations, making it a dominant standard in regions like Europe and North America. Moreover, ISO/IEC 15118 provides a standardized and secure interface for exchanging information during the charging process. The standard covers both wired and wireless communication interfaces, ensuring interoperability between different EVs and charging stations. It includes the Charging Communication Controller as a key component, which is responsible for handling communication between the EV and the charging infrastructure. ISO/IEC 15118 also introduces the concept of "Plug and Charge," allowing for automatic authentication and authorization of the charging session without the need for additional user interactions. Additionally, SAE J-1772, also known as the SAE Electric Vehicle Conductive Charge Coupler, is a standard that defines the physical and electrical interface between EVs and charging stations. It specifies the connector and signaling requirements for Level 1 and Level 2 charging, which are the common AC charging levels for EVs. The standard ensures interoperability and safety by providing a standardized connector design for charging infrastructure. It includes provisions for communication between the vehicle and the charging station, enabling features like power delivery control and safety interlocks. SAE J-1772 has been widely adopted in North America and is a crucial component of EV charging infrastructure in the region.

Moreover, there are several EVCS classifications. Level 1 chargers (slow chargers) are being replaced by Level 2 chargers, which are mostly used commercially as public EVCSs. Moreover, Level 3 chargers (providing a higher charging rate) are being introduced to improve the user experience and decrease charging times [6]. In this study, we focus on public EVCSs deployed by companies, governmental entities (e.g., Circuit Electric and ChargePoint), or private EVCSs that are made publicly available by the owner to earn extra income. It is worth highlighting that the EVCSs are also connected to the power grid (critical infrastructure) to draw the needed power for EVs to charge. Measurement units are distributed over the infrastructure and share information with the utility cloud. The utility utilizes this information to ensure visibility over the power grid.

2.2 Overview of EV Charging Mobile Application Platforms

To manage the ever-increasing number of EVCSs from different vendors, many companies have put in place a mobile application management platform for EV owners to monitor and control EVCSs. The mobile application is an indispensable component for the operation of the EV charging ecosystem. Consequently, we give an overview of the mobile application, which encompasses its

role and the communication protocols used when communicating with the different entities (CMS and EVCS). Mobile applications communicate with the CMS using **Hypertext Transfer Protocol Secure (HTTPS)**, which incorporates the **Secure Socket Layer (SSL)/Transport Layer Security (TLS)**, to secure and encrypt the HTTP communication and to preserve the privacy of users by protecting against on-path attacks. Various functionalities are instilled in the mobile application that provide users with an interface to remotely monitor and control EVCSs as illustrated in Figure 1. To name a few, a discovery service is provided by mobile applications to allow users to find nearby EVCSs. Furthermore, a control service is instilled to allow remote start and stop of charging sessions and schedule charging.

Various vendors provide mobile applications that allow remote monitoring/charging. To name a few, ChargePoint, a leading operator, has over 500,000 users. Similarly, ChargeHub is a popular mobile application that provides users with services to remotely control EVCSs. Indeed, the number of downloads of EV charging applications, which can be extracted from Google Play, shows the growth of the EV ecosystem and the wide distribution of the mobile application among consumers. This sheds light on the popularity and wide distribution of these applications that are used by many users. Moreover, the EVCSs and their management system in most cases belong to one or more different networks, thus allowing cross-application control. For example, the ChargePoint mobile application controls Blink, SemaCharge, and of course its own network. This highlights the importance of developing secure mobile applications due to its cross-operator collaboration, which adds more heterogeneity to the ecosystem, especially with the lack of standardization.

2.3 Related Work

In this section, we survey and discuss previous work that tackled the security of the EV charging ecosystem's components. The security was analyzed from various perspectives, one of which discussed the security software component and the communication protocols, and the implications of the security vulnerabilities on the infrastructure.

2.3.1 Software Components Security. Nasr et al. [7] studied and examined the security posture of the EVCSs and their management systems. They managed to find vulnerabilities across 13 severe vulnerability classes in firmware and management systems (mobile applications and websites). It is worth mentioning that in [7], mobile applications were analyzed using only static analysis, whereas our analysis utilizes both aspects to understand the interaction of the different components without taking into consideration the interactions of the components and design flaws. Moreover, outside of academia, the Kaspersky Lab team [17] analyzed the security of the ChargePoint home charging station and found significant vulnerabilities in its firmware and mobile management application. In our study, we provide a systematic and detailed analysis of the top 31 EV charging mobile applications. We are the first to address the lack of a comprehensive analysis of the EVCS mobile application, which plays an integral role in the EVCS ecosystem. It is worth noting that the mobile applications used to manage EVCSs are widely distributed and easily accessible, which simplifies and makes it easier for the attacker to acquire, analyze, and compromise, compared to firmware or cloud management systems.

Moreover, the communication links have been also studied in the literature. Alcaraz et al. [9] examined the communication protocol and presented a vulnerability in the OCPP that allows for MitM attacks, thus interfering in the communication between the EVCS and the EV resource reservation service. Moreover, in [18] they address the vulnerabilities presented in [9] and provide countermeasures. The security of the OCPP has been improved; various security measures have been implemented to harden the communication protocol such as the addition of secure firmware updates, security logging, and event notification and security profiles for authentication (key management for client-side certificates) and secure communication (TLS) [13].

2.3.2 Infrastructure. Sayed et al. [6] demonstrated the impact of compromising EVCSs on the power grid and launched attacks against it. They discussed the non-linear nature of the EV charging load and simulated multiple attacks that can be launched against the power grid using these EVs. While the grid was able to recover after a 48 MW attack utilizing traditional residential loads, a smaller 30 MW EV load attack was able to completely destabilize the grid. Moreover, in [19], the authors study how a botnet of compromised EVs and fast-charging direct current stations can be utilized to launch cyber attacks on the power grid and its implications on the transmission and distribution networks. Moreover, in [20–24], they study the implications of EV charging on the grid and discuss some mitigation techniques. In [25], they discuss the use of SMS phishing as a social-based attack, where an attacker can send spoofed text messages to the users advertising a discount (20% off when the users charge their vehicle at noon). Consequently, they studied the impact of such an attack on the grid. However, mobile phishing attacks require knowledge of the mobile phone numbers of EV users in a certain target area, which affects the feasibility of acquiring such information. Furthermore, in [25], the attack depends on the susceptibility of users to the demand-and-response phishing attack. In our study, we demonstrate how and when an attacker can get information to perform an orchestrated attack that might impact the stakeholders and study the feasibility of acquiring such information. Moreover, we demonstrate vulnerabilities that allow us to exploit the communication between the mobile application and the charging station. The discovered design flaws allow the adversary to charge any vehicle connected to an EVCS that could be used later in load-altering attacks.

2.3.3 Difference over Related Work. Zhou et al. [26] studied home IoT communication with the cloud to detect device interaction issues. They performed traffic analysis by intercepting mobile application traffic with the cloud and the IoT devices' interaction with the cloud. However, due to the limited access to the public EVCS infrastructure, we use functional analysis to monitor the EVCS state changes based on the mobile applications' user behavior. Moreover, we record user input in the application (e.g., user information, credit card, vehicle information) to understand the semantics of the traffic being sent. Utilizing traffic analysis similar to [26], the origin and the meaning of the traffic being sent would be lacking. The adversary in [26] needs to have access to the charging infrastructure to detect vulnerabilities; however, in our analysis, we only utilize the mobile application as a means to get information about the whole ecosystem. Furthermore, in [7], the authors focused on static analysis to analyze the vulnerabilities in mobile applications, which limits their ability to analyze the interactions of the ecosystem while utilizing the mobile application as an attack vector. To the best of our knowledge, analysis of the EVCS mobile application and its interaction with other components (CMS and EVCS) has not been done before.

2.4 Threat Model

To model the threat we considered different items to analyze and understand potential threats to the ecosystem. The items that were taken into account are:

- Assets: We consider the remote adversary with access to one or more mobile applications distributed on the Google Play Store and Apple store. Along with that, the adversary has access to online resources that aid in understanding the system architecture. This helps in understanding the value and criticality of the assets to help prioritize and scope the adversary's search for vulnerabilities.
- Entry points: The adversary then determines the entry points to the ecosystem. The attack surface considered in this work is the mobile application. Other attack surfaces that could be identified from the different assets are considered out of scope.

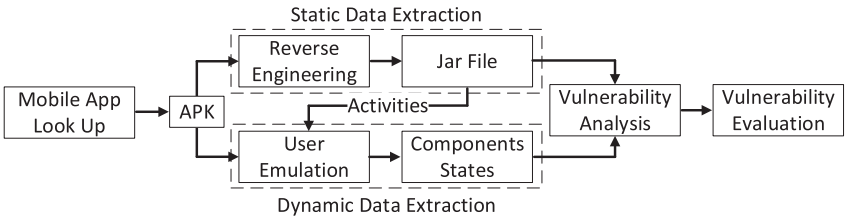


Fig. 2. The overall mobile application lookup and vulnerability analysis methodology.

- Trust boundaries: Identify the trust boundaries within the system, where different levels of access or privilege exist. Understanding the boundaries helps in evaluating potential threats associated with unauthorized access, the elevation of privileges, or trust exploitation.
- Threat actor goals: The attacker aims to utilize mobile applications as an entry point to target EVCSs with connected vehicles. The adversary's goal is to exploit design flaws in the interactions among the different entities (e.g., EVCS, CMS, mobile application) to hijack or initiate an unauthorized charging session remotely without compromising legitimate user accounts. Moreover, the adversary's ultimate goal is to leverage illegal charging sessions to perform large-scale, coordinated botnet attacks against the underlying critical infrastructure (e.g., the power grid) and the EV users.
- Threat actor: We consider the remote hacker that is able to create an account on EVCS charging mobile applications. Taking into consideration the threat actor goal, such large-scale attacks have been increasing lately where likewise state-backed attackers targeted critical infrastructure.

Additionally, similar to [26], we do not assume any forms of software bugs or protocol vulnerabilities. The adversary relies on understanding the interactions between the components by utilizing various analysis methods to identify vulnerabilities and understand the interactions between the mobile application and the CMS. The analysis methods range from reverse engineering and white-box testing to functionality analysis, system fuzzing, and black-box analysis.

3 METHODOLOGY

In this section, we elaborate on the analysis methodology to identify vulnerabilities that allow adversarial accounts to control charging sessions for vehicles they do not own. As shown in Figure 2, we combine static (reverse engineering and code review) and dynamic analysis techniques (functionality and traffic) to perform vulnerability analysis and assessment of the identified mobile applications. First, we start by fetching EV charging mobile applications, and then, for each product, we extract data for analysis by applying reverse engineering techniques on their binaries. Second, we extract network traffic during the functionality analysis while emulating the user behavior of the application; consequently, we analyze the application states and behavioral changes to abstract the system interactions and then evaluate it to find flaws. We provide details on the proposed methodology components below.

3.1 Mobile Application Lookup

In this section, we discuss our selection strategy for mobile applications. According to Statista [27], Android maintained its position as the leading mobile operating system for mobile phones with about 73% market share. We look for EV charging mobile applications on the Google Play store, which is the main platform used by Android users to download applications. Furthermore, we automatically fetch 50 mobile applications from the Google Play store. Then, we choose the mobile

Table 1. Types of EV Charging Mobile Applications Based on Their Abilities

Description	Type	Application Names
Remote control of charging sessions and system overview	1	**Remote Start Charging:** ChargeHub[a,b] - Electrify Canada[c] - PodPoint[a,b] - Electrify America[a,b] - EVDC[a,b] - Semma Connect[a,b] - eCharge Network[a,b] - Tata Power EZ Charge[c] - Flo EV Charging[a,b] - BC Hydro EV[a,b] - Circuit Électrique[a,b] - PlugShare[a,b] **Remote Start and Stop Charging:** Petro-Canada EV[a,b] - ChargePoint[a,b] - vend-electric[a,b] - Anywhere Charging[a,b] - Electromaps[a,b] - Ionity[a,b] - Volta Charging[a,b] - Charge Assist[a,b] - Virta[a,b] - Global Charge[a,b] - EV Charging by NewMotion[a,b] - EV Match[a,b] - EcoFactor Network[a,b] - FastNed[a,b] - EVgo[a,b] - Greenlots[a,b] - EVduty[a,b] - EV Connect[a,b] - NextCharge[a,b]
System overview	2	Zap-Map - Charge Map - EVMap - Kazam EV - Chargeway - Charge Finder - Open Charge Map - EV Stations
Home charging optimization	3	EV Energy - OptiWatt - Monta EV Charging

[a]Indicates the mobile application operators that possess Flaw 1 (Unverified Ownership). [b]Indicates the mobile application operators that possess Flaw 2 (Improper authorization for a critical function). [c]Indicates the mobile application operators that possess Flaw 1 and 2 but mitigate them by requiring information only found physically on the EVCS HMI.

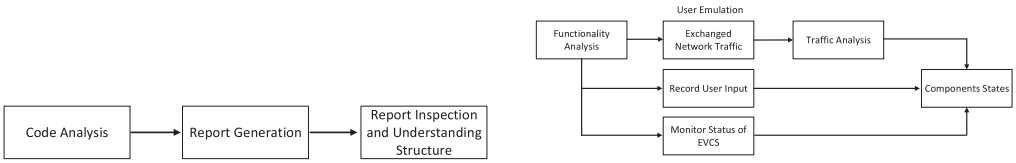
applications and filter them based on the features they possess by automatically searching the description for keywords such as start/stop charging. After further analysis, we discarded eight mobile applications that are either EV charging calculators or not related. Our analysis focuses on mobile applications that provide remote control functionalities to control public EVCSs; consequently, we analyze the applications manually to ensure the existence of these functionalities as they pose a real danger to the power grid when compromised at scale [6].

Based on the prior differentiation between the applications according to their capabilities, we identify and classify the applications into three types, as shown in Table 1. Type 1 applications allow users to have an overview of the ecosystem and control the charging session, while Type 2 applications are used to perform reconnaissance activities and can only show an overview of the system. Type 3 mobile applications are developed to target home EVCS owners or businesses with private EVCSs, limiting their impact, especially from a power grid perspective. Type 3 applications could possess vulnerabilities; however, it is considered out of the scope of this work as we focus on mobile applications that provide access to public EVCSs, thus increasing the impact of an attack on the power grid. Namely, we focus on Type 1 applications that can be used to perform attacks on the power grid because of their special capability to control the EVCS and its operations. Additionally, Type 1 and 2 applications can be used by an adversary to prepare for their attacks by analyzing the availability of EVCSs and their usage trends, as discussed in Section 4.4.

Finally, we fetch the remaining mobile applications and download/install their APKs. It is crucial to highlight that any security concern discovered in the communication between the mobile application and the CMS applies to both Android applications and iOS since they rely on the same back-end that handles their requests in most cases. Therefore, we assume that our analysis methodology and results can be generalized to both platforms, respectively. However, confirming this will be considered for future work.

3.2 Static Analysis

We aim at documenting and understanding the functionality of mobile applications and their utilized libraries. We used static analysis to understand the security measures implemented by EV charging mobile application vendors to thwart automated bot attacks by examining libraries and artifacts found in the binary files, along with understanding the structure of the mobile applications (Figure 3(a)) to perform a systematic functionality analysis. Thus, we utilize reverse engineering of the APK, which is an archive package that contains a manifest file with the package name, activity names, hardware features support, permission, and other configurations. The APK also contains



(a) Static (b) Dynamic

Fig. 3. Overview of the static and dynamic analysis methodologies.

the certificates for the application, a `lib` directory holding compiled libraries used by the application, and a file with compiled application code in `dex` file format, which can be interpreted by the **Dalvik Virtual Machine (DVM)** and the Android runtime environment.

We extract all the files using apktool [28], which disassembles all the resources and extracts the application Manifest and the `dex` files. Consequently, we use the extracted dex files and convert them to a `JAR` file using dex2jar [29] utility. We then input the file into jd-gui [30] to browse the underlying Java source code. We then analyze the extracted jar files using white-box analysis techniques to check the application resources (e.g., libraries and their functionality, certificate signing techniques used by the application developer, etc.). Further, we extract resources from the generated reports (e.g., the activities used in the mobile application and the flow of activities), which allow us to perform a detailed functionality analysis and identify libraries used in EV charging mobile applications by using MobSF [31] and LiteRadar [32]. Consequently, we check the functionality of the libraries used (e.g., Google reCAPTCHA, hCAPTCHA, Anti-location Spoofing). The understanding created by studying the information obtained through static analysis is used to systematically guide the following step, which is the dynamic analysis.

3.3 Dynamic Analysis

We rely on dynamic analysis (Figure 3(b)) to complement our static analysis method and provide a holistic and comprehensive assessment of all the interactions and functionalities provided by the mobile application. The dynamic analysis is performed through functionality analysis, recording user input, traffic analysis, and monitoring system state changes.

Functionality Analysis. We perform functionality analysis to collect data and identify system states to understand the communication between different entities. Specifically, we seek to answer the question of "How can the adversary utilize the interaction vulnerabilities and weaknesses to connect to a remote EVCS and control its operations?"

Guided by the static analysis that allowed us to understand the structure of each mobile application by mapping its activity flow, we attempt to systematically cover all the possible scenarios to systematically perform a detailed functionality analysis [33, 34]. We analyze each mobile application by manually mimicking regular users' behavior and operations and triggering every functionality possible in the mobile application. Throughout our analysis, we discover that some functionalities are strictly prohibited based on the location of the device. For example, Petro-Canada EV prevents initiating charging requests if the users' location is not in the vicinity of the EVCS. However, to mitigate that during our functionality analysis, we spoof the location of the device by broadcasting a location close to the EVCS. We utilized GPS JoyStick ADB Shell [35] to spoof the device's location. It is worth mentioning that other applications (e.g., Electric Circuit) notify the user that they are far away from the location of the charging station; however, this does not restrict the user from initiating a charging request. Furthermore, some applications (e.g., electromaps) detect that the user is far from the EVCS even if the location was spoofed. This is attributed to the IP-Geolocation services used to detect the location of the originating IP, which

can be circumvented using a VPN that routes our traffic through a private tunnel that appears to originate from the same country as the charging station.

Traffic Analysis. While emulating user behavior and performing user actions, we capture the traffic generated by the mobile application to understand the information that is being sent and the interactions between the mobile application and the CMS similar to [26]. We trigger important functionalities of the application such as sign-up/in and start/stop charging. However, most applications use trusted **Certificate Authorities (CAs)** to protect user privacy by encrypting the communication between the mobile application and the CMSs [26]. To decrypt the communication we utilize an un-rooted device with the Android 7 operating system.

Moreover, since Android OS with version ($\geq$ 7.0) does not trust user-installed certificates by design, to run applications on an un-rooted device with user-installed certificates, we create a virtual space on the phone that allows running Android APKs as plugins by utilizing VirtualXposed [36]. Consequently, we perform API hooking to bypass certificate pinning/verification by using Inspeckage Package Inspector [37]. Moreover, in some applications we bypass certificate verificationby reverse engineering the application using APKTool [28], followed by injecting code into the application. We then repack and sign the application before installing it. Consequently, we utilize Burpsuite [38], which operates as a proxy server between the mobile application and the target server to intercept, inspect, and modify raw traffic passing in both directions. Our analysis was not intrusive; however, it allowed us to unravel and understand the communication between the mobile application and the CMS.

4 RESULTS

We utilize different methods to infer and analyze the interaction of the main components within the EV charging ecosystem during user and device registration and when initiating charging requests. We extract the capabilities instilled in each mobile application. The capabilities recorded for each mobile application include remote start charging and remote stop charging. Moreover, we record the remote control restrictions that are implemented by the mobile application vendors to hinder the illegal or abnormal usage of a charging station. The vendors limit the flexibility for the user to initiate charging requests based on (1) location proximity of the users (i.e., must be near the target charging station to initiate requests), (2) IP Geolocation info (i.e., charging requests should originate from the same country/area as the target charging station), and (3) user-entered charging station ID that is found on the target device.

Our preliminary analysis shows that all applications provide their users with a remote start charging service. Moreover, only 13 (e.g., Flo EV Charging, Plugshare, etc.) mobile applications do not provide stop charging functionality, forcing users to remove their cars when they finish charging. It is important to note that only two applications (Petro Canada EV and Electromaps) check the integrity of the users by validating the location of the device, whereas only one application (Electromaps) checks the locations of the originating IP of the charging request. Finally, two applications force the user to input a station ID or scan a QR code to initiate charging (Tata Power EZ Charge and Electrify Canada).

4.1 Inferred Interactions

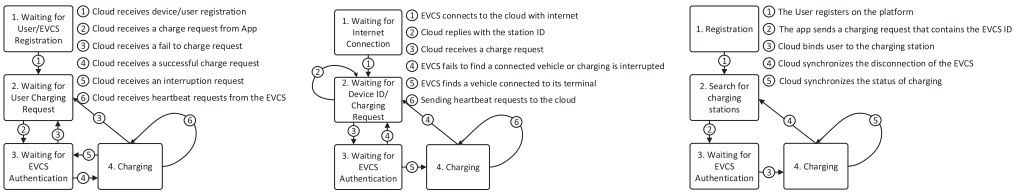
As described in Section 3.3, we leveraged dynamic traffic analysis to capture and infer the interactions between the mobile application and the CMS. As an adversary, we consider the communication between the EVCS and the CMS as a black box. However, the communication between the other entities can be inferred from the different states that the mobile application goes through while performing different actions. Moreover, while previous works presented in [7, 9, 13, 14] complement our analysis of the communication between the EV, EVCS, and CMS, they provide

us with some additional insights about such communication and interactions. Specifically, it has been shown that the communication between the EV and the EVCS lacks proper security measures, which renders the underlying equipment vulnerable to remote attacks. For instance, Baker et al. [14] were able to eavesdrop on the **Pulse-Width Modulation (PWM)** communication, which is utilized by IEC 61851 [39, 40] for safety-related signaling mechanisms between EVs and EVCSs. Furthermore, despite the added security features in the ISO/IEC 15118 (e.g., Signal-level Attenuation Characterization and TLS encryption), the works in [14, 39, 40] highlighted the improper deployment of these features in practice. Moreover, as highlighted in [6], most of these security features are optional and are commonly ignored by manufacturers, thus rendering devices vulnerable in real life.

User and EVCS Registration. To this end, we analyzed the EVCS charging applications listed in Table 1 of Type 1 to infer the interaction between the different entities upon user registration, EVCS registration, and sending a charging request. We identify the main interactions with the CMS during the registration of a new user and an EVCS. During user registration, each user is assigned a unique identifier, which allows the user to log into the platform and use existing functionalities. There are several options for the user identifiers such as email/password combination or authentication tokens to name a few. The unique user identifier is transferred to the CMS upon user registration on a given platform and then used later for authentication purposes. On the other hand, when an EVCS is installed and made available for the public, the operator needs to register it with the CMS by sending/registering its unique identifier. This information gets saved in the CMS and used by the mobile application to identify a charging station.

Initiating Charging Requests. After the user connects the vehicle to the charging station using one of the available connectors, the user must initiate a charging request using the mobile application by selecting the desired charging station (e.g., using a map view). The application embeds the unique user identifier along with the selected station's ID in the message sent to the CMS, respectively. The CMS then sends a start charge request to the charging station with the respective ID/info. Consequently, the EVCS checks for any connected vehicle before initiating the charging session. Note that in case no vehicle was connected at the time when the user initiates a charging request, the EVCS will wait for a grace period (e.g., 5 minutes) to provide the EV owner with sufficient time to plug the charger into the vehicle. Otherwise, if a car was found to be connected to the EVCS, it will initiate the charging session by sending a confirmation message to the CMS; this is inferred by monitoring the changes on the mobile application user interface. Once the CMS receives the confirmation, a correlation between the user identifier and the EVCS identifier is established. Finally, the CMS relays the charging confirmation sent by the EVCS to the mobile application.

To put this in a better context, we describe the state transitions inferred from the analysis of different platforms provided by the various EVCS operators in the industry. We validated the interactions that helped us derive the states of the components on a real-time co-simulation test bed and two EVCSs acquired from one of the biggest North American operators. Any action triggered on the mobile application has a cascading effect on the other components and will alter the state of the other components. For example, whenever a user initiates a charging session from the mobile application and transitions from S2 to S3 as illustrated in Figure 4(c), the EVCS and the CMS will transition from S2 to S3 and eventually S4; the state of each component is dependent on the actions done by the other components. The different components making up the EVCS ecosystem are tightly coupled. The transition of one entity from one state to the other would change the current state of the ecosystem. An ideal system must strictly maintain the three-entity state machine. The legitimate states of the EVCS ecosystem are depicted as a 3-tuple combination. The CMS, EVCS, and mobile application must strictly maintain the following 3-tuple states at all times to avoid potential attacks. For example, if an attacker is able to induce the cloud and the EVCS to transition



(a) Cloud management system (CMS) (b) EV charging station (EVCS) (c) Mobile application

Fig. 4. High-level state machines for the three interacting components within the ecosystem.

to state S3 while the legitimate user is still in state S2, this shows that the charging session was hijacked by a third party (adversary). Moreover, another 3-tuple state if triggered can also lead to similar consequences by forcing the cloud and the EVCS to transition to state S4, whereas the legitimate user mobile application is in state S1 or S2. Whenever a legitimate user is looking to charge his or her vehicle using a certain EVCS, the cloud state and the EVCS state should allow future transitions rather than being blocked. If user B is charging, user A is not allowed to access the EVCS remotely. Note that all components have the same numeric state at all times, except at S1, where the CMS can be in S1 while the other components could be in either S1 or S2. Consequently, through this work we aim at dissecting the intricate interactions of the components to trigger illegal states. The state machine shows the general operation of the EVCS ecosystem and shows the type of information that is shared during the process. Thus, through our analysis, we identify several vulnerabilities related to the trust model adopted in these ecosystems. Moreover, a deep understanding of a system allows us to identify flaws in the ecosystem design that are related to how the components interact with each other. Thus, we performed a systematic unraveling of the different components and their interactions.

4.2 Identified Vulnerabilities

In what follows, we present examples of the identified vulnerabilities that can be exploited to perform remote attacks against the EV charging ecosystem and the various involved stakeholders (e.g., EV consumers and the power grid). We leveraged the described analysis methodology in Section 3 along with the inferred traffic/interactions in the previous sub-section to identify vulnerabilities.

Specifically, we discovered three major security weaknesses that are inherited from design and implementation flaws in the studied EV charging mobile application (Type 1). The EVCS charging platform does not strictly comply with the legitimate 3-tuple states. We found that the three entities stay in multiple unexpected 3-tuple states. The first unexpected state is (S4, S4, S1/S2); the CMS and the EVCS transition to the charging state, whereas the mobile application user is either still in registration or EVCS discovering state. This illegal state combination when exploited by an adversary could allow for remote charging/discharging session hijacking.

In what follows, we elaborate further on the root cause of such behavior, the identified vulnerabilities, and their implications. It is worth noting that some mobile applications (e.g., EVMatch) mitigate the first unexpected state by forcing the user to reserve a spot beforehand. Other applications (e.g., Tata Power EZ Charge) hinder remote hijacking by forcing users to scan a QR code on the EVCS. However, when statically analyzing the mobile applications, QR codes are saved in a temporary file in the external SD card, which allows an on-device attacker to get access to that information to hijack the charging sessions. It is worth noting that some applications hide the access to charging behind a payment gateway (e.g., buying store credit). However, adversaries can overcome this by buying store credit, which will provide access to the charging infrastructure. In what follows we focus only on the flaws that can be used to manipulate the EVCSs for attacking the grid.

Flaw 1 (F1): Unverified Ownership. Ideally, an EV user should be the owner or authorized user of the vehicle. Thus, the EV user should be the sole entity to authorize any form of control or action on the vehicle. Interestingly, our static and dynamic analysis results indicate that the mobile applications do not verify user ownership over target vehicles when initiating charging requests. In other words, an EV charging mobile application user can initiate a charging request to any vehicle connected to the network since neither the mobile application nor the CMS has a mechanism to bind the application user to the target vehicle. Given that all communications of the mobile application go through the CMS, it is imperative to have the CMS verify critical operations such as vehicle identification and ownership management. On the other hand, access to vehicles from an unauthorized user is not verified within the EV charging mobile application platforms, therefore rendering the EV exposed to any user who can claim ownership and control over its charging functions when connected to the EVCS. This leads to unexpected behaviors and potentially exploitable states.

Flaw 2 (F2): Improper Authorization for a Critical Function. Starting and stopping charging operations on a given EVCS are considered examples of critical functions that could be abused by adversaries (unauthorized users) to destabilize the operations of the EVCS and the connected power grid. This was clearly demonstrated in [6], where the authors measured the impact of mass charging and stopping operations on the power grid. Ideally, an EV owner/operator should be the only user authorized to perform critical functions on the vehicle. Exposing critical functionalities essentially allows adversaries to control charging sessions, which is considered as the first step toward initiating mass charging attacks on the grid. Nevertheless, our analysis results indicate that there is a lack of authorization, which allows any actor to perform critical functionalities on the connected EVs. This is closely related to our first finding (F1); it is mainly due to the fact that the binding step happens only based on the user and charging station IDs without further verification of the EV ownership or binding to specific mobile users. Therefore, an adversary can utilize fake accounts to hijack sessions.

4.3 Attack Scenarios

An attacker can leverage the discussed vulnerabilities to launch various malicious activities against the EVCS and its operations such as remote charging session hijacking. To do this, attackers need to control a number of adversarial accounts (i.e., bots), which provide access to existing charging services through mobile applications. Note that adversaries do not need to exploit or hijack user accounts to create the required botnet. The attackers can easily create their own botnet of legitimate mobile application accounts. The only security measures in place rely on SMS or email authentication/verification during account creation (e.g., one-time password and email verification). In practice, an attacker could rent service from online SMS and e-mail providers such as Twilio [41], which provide communication APIs to handle the verification processes. Additionally, attackers can create as many fake email accounts as needed for the verification process.

Remote Charging Session Hijacking. After analyzing the interactions of the mobile application with the different entities, we developed an understanding of how the mobile application could be used as an attack surface against the power grid. We found that the studied platforms are vulnerable to session hijacking. By utilizing these vulnerabilities, attackers can initiate unauthorized EV charging sessions with the aim to impact the power grid. Ideally, the CMS should only allow a charging request if the request is issued from the account owner that is bound with the EV. However, we found that the CMS does not perform any account-based authorization or check during charging. In other words, there is a decoupling between the user account and the EV that is connected to the EVCS. The user is coupled with the EVCS ID only. Thus, an EV connected to an EVCS can be charged remotely regardless of whether the user initiating the request is the

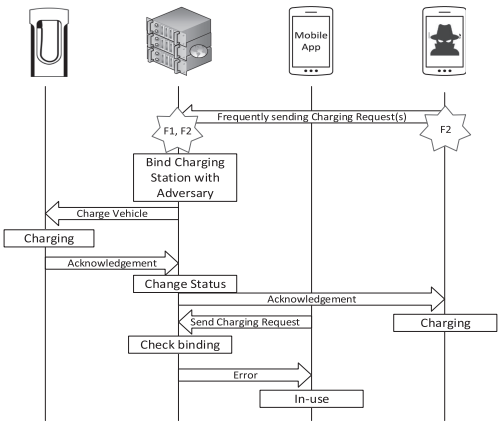


Fig. 5. Sequence diagram depicting remote charging/discharging scenario.

legitimate owner of the EV. The CMS does not perform the necessary checks to validate whether a user is allowed to perform this action or if the user is the actual owner of the EV, thus allowing adversarial accounts to unlawfully hijack charging sessions.

As illustrated in Figure 5, the adversary can leverage the combination of F1 and F2 to initiate unauthorized charging requests to take control over the charging session that should have been initiated by the actual EV owner. When the attacker uses an adversarial account to start charging requests, the CMS will establish a connection with the adversary. It is worth mentioning that the actions performed by the adversary are fully legitimate and within the scope of the permitted functionalities of the ecosystem. Consequently, legitimate EV owners can no longer control their charging session. The only way for the user to stop the charging is by physically unplugging the EV. The adversary utilizes the mobile application as an attack surface against the ecosystem. The adversary creates legitimate accounts and lurks in the system, monitoring the behavior of victim EVCSs to be targeted. The adversary studies the behavior of the EVCS through its status that is shown through the application. The behavior allows the adversary to understand peak hours when the EVCS is mostly utilized and discover when users usually use the EVCS. The attacker can then stage their attack by targeting peak hours to increase the success rate of the attack as described in Section 4.4. The adversary leverages the unverified ownership of a user over the vehicle they are trying to charge and the lack of authentication on this critical function to send charging requests.

After the user's charging session has been hijacked, the user can no longer control the EVCS. While this could raise an alert for a security-savvy user, other users may simply disregard this behavior as long as they see that their EV is charging. It is worth mentioning that even the security-savvy users will not notice the attacker's actions unless they regularly check their mobile applications during the charging process. Additionally, even when the attack is noticed by these users, the root of the problem cannot be traced back to the adversary. Only the CMS has the knowledge to trace back the attack's origin. However, due to F1 and F2, the attacker's actions are considered legitimate.

We describe in Table 1 the possible attack scenarios based on the functionalities instilled in the mobile application, showing that 29 out of 31 mobile applications are vulnerable to remote mass charging attacks. We exclude Tata Power EZ Charge and Electrify America as they require inputting the EVCS ID number that is physically placed on the charging station HMI. Moreover, there are only 19 out of the 31 mobile applications that provide remote start and stop of

charging and allow adversaries to launch oscillatory load attacks with their advanced control on the stopping of charging.

Remote Discharging. Vehicle-to-Grid (V2G) capabilities are one of the attractive features of EVs that can one day transform the EV battery into a distributed storage to support the power grid operation. Willing EV owners would register themselves as users willing to contribute to supporting the grid during peak hours for an incentive (e.g., financial incentives larger than the cost of charging) through utilizing their mobile application. However, as demonstrated above, the current system architecture lacks traceability and end-to-end authentication. An adversary can hijack a session, as explained above, and register themselves as a legitimate user that is willing to contribute to such a V2G scheme. This allows the adversary to gain monetary compensation by discharging other users' vehicles. We acknowledge that such a class of attacks is not currently feasible due to the shy adoption of V2G capabilities and the absence of wide-scale compensation programs. However, the advancement toward such capabilities being instilled in the ecosystem requires improving the current ecosystem architecture and strict access control mechanisms for safe and secure operation.

4.4 Attack Feasibility

In this section, we study the feasibility of launching wide-scale coordinated charging/discharging attacks. In these attacks, we assume that users connect their EVs to EVCSs before starting a charging session. We also assume that the vehicle remains connected for a period of time after the end of the charging. In the aims to understand user behavior and predict it, the authors in [42] highlight that EV owners do not necessarily start charging right after plugging in. Additionally, a time window exists between plugging in and charging an EV, which is the time a user needs to pull out the phone to start a charging session. Moreover, according to Almaghrebi et al. [43], another time window exists where customers leave their vehicles for an extended time when parking at the workplace or overnight beyond finishing charging. Some users even leave their vehicles for longer than 24 hours. This attack window is only applicable to mobile applications that provide remote start and stop services. These time windows are exploitable by the adversary that can utilize them to launch remote session hijacking.

In [44], the authors utilize a multistep hybrid LSTM neural network to predict EVCS occupancy. They base their analysis on public charging data from the city of Dundee, UK, in 2018. The number of charging stations plugged in simultaneously during the day fluctuates, reaching 300 charging sessions at 10:00 AM during the weekend and 400 charging sessions during the weekday. The number of charging sessions starting at peak times during the day is expected to increase as more customers adopt electric vehicles as a means of transportation with the rapid and increased deployment of public charging stations. Thus, the feasibility of remote charging session hijacking at scale increases.

4.5 Attack Demonstration and Verification

To execute an attack by exploiting these vulnerabilities, an adversary needs information about the user's behavior. By understanding user behavior, the adversary can time and coordinate the attack to increase its success rate. The attacker can extract information from the mobile applications (Type 1 and Type 2) similar to [45], where the authors predicted user behavior based on arrival time, duration, departure time, and so forth. An adversary can utilize the online interface (mobile application/web portals) to gather information. The information we gathered through a tool we devised is the start charge time (arrival time) and departure time if a vehicle connected within the attack windows. We used Appium [46] to automate mobile application scraping, which can be used for web application scraping. We then identify target EVCSs and monitor their utilization

and status. We collect information about the EVCSs that are in use, allowing us to track arrival and departure times. Moreover, whenever the station's status changes from "in use" to "available," we send a probe charging request to check if there is an EV connected to the EVCS. The collected data is used to model user behavior and allow us and the attacker to target peak EV connectivity hours to hijack charging sessions at scale [44].

In this section, we evaluate and verify our observations, inferences, and conclusions by using a real-time co-simulation test bed. We create a replica of the real EVCS ecosystem by integrating real charging station hardware with a production-grade CMS. The EVCS hardware utilizes OCPP v1.6 and communicates with the CMS backend to perform device registration initially. The EVCS and the CMS then continuously communicate with each other over WebSockets to ensure that the EVCS is alive by either sending a heartbeat notification or through the WebSockets ping-pong request/response. Indeed, during device registration, the EVCS will send an HTTP request to the CMS backend, which gets upgraded to a WebSocket connection. The HTTP header includes the EVCS identification number, which could be the serial number or an operator-defined ID as demonstrated in Figure 6. Consequently, we were able to confirm and verify our observations and inferences regarding the interactions of the different components discussed previously.

Accordingly, we aim to demonstrate our new attack vector to verify our conclusions on the vulnerabilities that exist in the EVCS mobile operators. To leverage the aforementioned design flaws that rely on the authentication scheme adopted by the different EVCS mobile operators, we first identify two EVCSs mainly in heavily populated areas. We monitor these EVCSs and record their utilization by gathering information from the mobile interface of the applications for 3 days starting on November 8, 2022, till November 10, 2022. During that time we performed reconnaissance to understand the utilization of the EVCS. $EVCS_1$ showed heavy arrival during evening hours (between 6:30 PM and 7:30 PM), whereas $EVCS_2$ showed heavy arrival between 4:00 PM and 5:00 PM. To identify the utilization and arrival, we note the change in the state of the EVCS. When charging an EV, the EVCS shows an "in-use" state, rendering it unavailable to other users. Consequently, on November 11, 2022, we execute our attack as a proof of concept on the EVCSs. We leverage the lack of rate limiting to send multiple charging requests every 3 to 4 minutes from the adversarial account we created using the legitimate mobile application channels. After sustaining the attack for almost 30 minutes starting at 6:30 PM for $EVCS_1$ and starting at 4:00 PM for $EVCS_2$, consequently, at 7:01 PM and at 4:18 PM we were able to successfully hijack the charging session of the $EVCS_1$ and $EVCS_2$, respectively. We show in Figure 7 the confirmation of a successful charging of a vehicle that does not belong to us for almost 3 hours. Moreover, the attack was demonstrated and verified on two different mobile applications, i.e., one that only allows a start charge functionality and another that allows remote start and stops charging functionality. We also note that vulnerabilities that allow the adversary to remotely start and stop a session could be used in combination with other high-wattage IoTs to impact the power grid. In [47], the authors demonstrate the impact of controlling a large botnet swarm of high-wattage IoTs that could be used to impact the power grid by launching load-altering attacks. Load-altering attacks impact the power grid by inducing grid instability (e.g., frequency instability). Finally, the same attack workflow could be launched to impact the power grid using the other mobile applications as they follow the same procedures and policies to authenticate users.

We illustrate in Figure 8 the components of the co-simulation platform we have developed. The co-simulation platform is composed of cyber-components connected to the physical counterpart resembllling Hypersim, which simulates the power grid for physical impact studies. Moreover, real EVCSs are included in the simulation and are connected to the actual power grid to evaluate and verify our cyber layer observations, inferences, and conclusions. The mobile application is an essential component for the commercialization of the EVCS ecosystem. It provides different remote

```
GET /ocpp/[redacted] HTTP/1.1
Host: [redacted]
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: yc00jOdENekBP7wdb6P1lg==
Sec-WebSocket-Version: 13
Sec-WebSocket-Extensions: permessage-deflate; client_max_window_bits
Sec-WebSocket-Protocol: ocpp1.6
User-Agent: Python/3.10 websockets/10.3

HTTP/1.1 101 Switching Protocols
Date: Thu, 08 Dec 2022 18:46:11 GMT
Connection: upgrade
Upgrade: websocket
Sec-WebSocket-Accept: SX/d29VizNfPapmpmC4xjyxGHQI=
Sec-WebSocket-Protocol: ocpp1.6
```

Fig. 6. EVCS device registration with the CMS using our real-time co-simulation test bed.

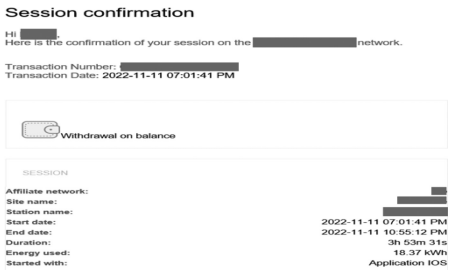


Fig. 7. Session confirmation showing the success of our attack by hijacking the charging of an idle vehicle.

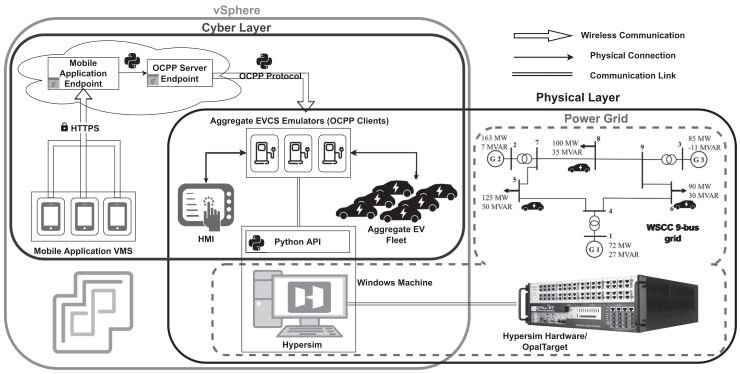
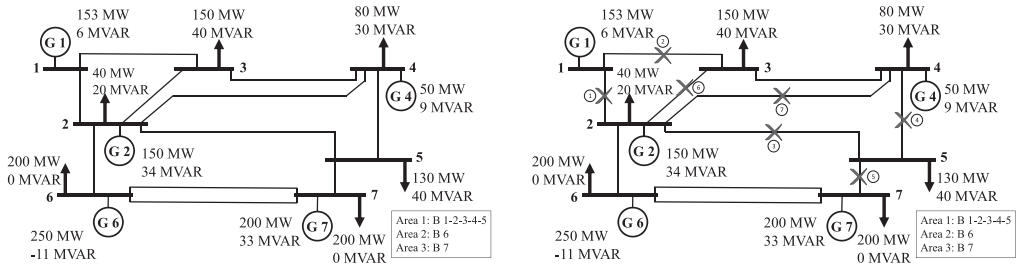


Fig. 8. Co-simulation architecture [48].

Table 2. Specifications of the Real-time Co-simulation Test Bed

Technology	Specification
vSphere ESXi	Version 6.0.0
Hypersim	Version 2022.1
Hypersim Simulation Step Size	25 µs
OpalTarget	OP5707XG - RCP/HIL Virtex-7 FPGA-based Real-time Simulator
EVCS VM	1 GB 1 CPU
CMS VM	1 GB 1 CPU
Mobile App VM	1 GB 1 CPU
Python Interface	Python 3.7

capabilities of starting, stopping, and scheduling charging. For the current implementation, we focus on starting and stopping functionalities. Moreover, as for the EVCSs, we use both simulated and real hardware that are part of our co-simulation environment. The EVCS VM runs a client-side OCPP developed using Python along with the CMS we developed that runs a server-side OCPP, which is used to manage the EVCSs remotely. The OCPP implementation has been validated with a production-grade EVCS and CMS provided by our partner Hydro-Quebec as part of a legal agreement and research collaboration. The CMS also provides a mobile application endpoint that is responsible for sending and receiving requests from the mobile application VMs. The specifications of the co-simulation platform are summarized in Table 2. Further details of the co-simulation platform can be found in [48].



(a) Glover book 7-bus grid (b) Line tripping attack impact

Fig. 9. Overview of the (a) Glover book seven-bus grid and (b) the impact of the line tripping attack scenario.

5 ATTACK IMPLICATIONS

In what follows, we discuss the remote charging session hijacking attack scenario along with its implications on the power grid and EV users, respectively.

5.1 Attack Implications on the Power Grid

As demonstrated in the analysis, attackers can leverage the identified vulnerabilities to compromise user accounts and perform synchronized large-scale cyber-attacks against the integrated infrastructure [5, 7, 49, 50]. With the EV charging ecosystem being a new and wide attack surface, it is an attractive target for exploitation by organizations with enough resources to conduct large-scale attacks against the power grid by utilizing the mobile application to perform stealthy attacks.

Consequently, an adversary could initiate a distributed botnet attack utilizing thousands of malicious accounts to send charging requests and hijack as many sessions as possible to amplify the attack. The behavior of arrival and departure at charging stations almost coincides with the demand behavior of the power grid [45], as demonstrated in the utility demand curve of California, United States [51], and New South Wales (NSW), Australia [52]. Additionally, by exploiting the identified vulnerabilities and initiating the remote charging session hijacking attack (Section 4.3) during the described attack windows (Section 4.4), an adversary can remotely orchestrate hijacked charging sessions to synchronize a wide-scale attack that can disrupt the power grid operations.

In this section, we study the impact of synchronized mass charging attacks on power system economics (i.e., generation cost and transmission line losses). We then examine how adversaries with some knowledge of the grid topology can craft targeted mass charging attacks in order to overload and trip transmission lines. Finally, we study the power grid stability subject to oscillatory load attacks that can cause violation of the safe frequency operation limits and load shedding. Oscillatory load attacks can be performed using 19 of the applications that provide on-and-off remote control capabilities without requiring the user to scan a QR code.

To amplify the attack impact on the grid, an adversary with knowledge of the grid can craft targeted and smarter attacks. A small number of compromised charging sessions with enough knowledge of weak buses allow the adversary to disrupt the power grid operations. Power grid information can be estimated through monitoring the measurements of the power grid to estimate the topology, using MILP programming, machine learning, and voltage and load monitoring [6, 53–59]. Various stability techniques and strategies could then be used by adversaries to locate the most sensitive/vulnerable buses, such as PV and QV curves [6, 60].

We demonstrate the impact of the attacks on the seven-bus test case introduced by Glover et al. [61] (Figure 9(a)), which is commonly used for research purposes [6]. We utilize this grid due to its built-in optimal power dispatching capabilities, unlike the work in [7]. Moreover, this seven-bus test case provides the generation cost formulas that will allow us to study the economic

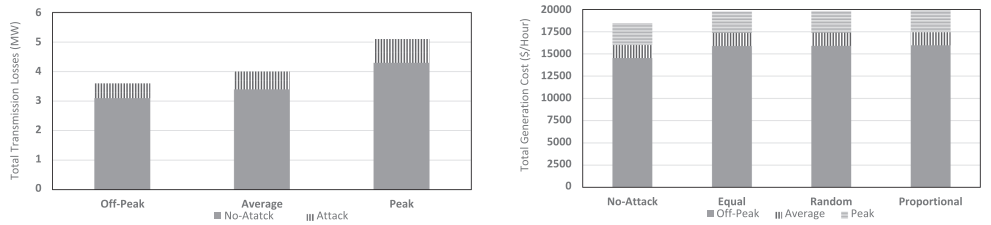
impacts on the utility. To achieve a close-to-realistic simulation of the power grid behavior during peak and off-peak demand hours, we scaled the grid loads based on the NSW [52] power grid load profile using Equation (1):

$$HourlyLoad_{Scaled} = \frac{GloverLoad \times HourlyLoad_{NSW}}{AverageLoad_{NSW}}. \quad (1)$$

To this end, we use PowerWorld [62], which is a power simulator that allows us to analyze the steady-state power flow, transient stability, generation costs, and other power system operations. Additionally, the stability studies are initially performed in PowerWorld and then validated using our real-time co-simulation test bed depicted in Figure 8. Unlike [6, 47], here we study the economical aspects of an attack such as generation cost and line losses, respectively. Along with that, we also take into account the load shedding mechanism that is used by the utility to regulate power generation in case of a sudden drop in frequency below certain thresholds [63] to demonstrate more realistic attack implications. The different attack simulations and results are demonstrated below. In what follows, the attack is initiated by compromising 84 MW of EV load, which is equivalent to 7,636 EVs charging at the 11 kW Level 2 chargers. The current numbers of EVCSs and EVs are not enough to mount such attacks; however, the growth in the EV numbers will soon provide a large enough surface to make it possible [6]. It is worth noting that mobile applications allow cross-product communication and control, thus increasing the scale of the attack as more vendors join these platforms. Moreover, as the EVCS market move toward wide adoption of Level 3 chargers, the higher power entails higher risk.

Economical Impact. The attacker can cause the power utility to incur economic losses by launching EV attacks against the power grid. To study the economic impacts of a mass charging attack on the grid, we examine the transmission line losses and the power generation cost during different loading conditions and under different attack scenarios. To perform mass charging attacks, 29 applications allow us to perform such an attack, whereas the rest prevent remote mass charging by forcing the adversary to scan a QR code. We used the scaled load profile to demonstrate the incurred cost and losses at different grid loading conditions. Namely, we focused on the peak load (943 MW), the average load (800 MW), and the minimum load (677 MW) conditions that we will refer to later as off-peak load. We simulate three different attack scenarios against the test grid by (1) distributing the attack load randomly, (2) distributing the attack load equally among the six load buses, and (3) distributing the attack load proportionally among the different load buses. It is worth highlighting that Scenario (1) represents a random distribution of the EV charging attack load to simulate an adversary with no knowledge of the grid topology. Scenarios (2) and (3) represent attacks by an adversary with limited knowledge of the grid and geographical knowledge of load size and EV distribution, respectively.

Generally speaking, the attacks will increase the transmission losses under all loading conditions. However, under higher loading conditions, the same attack will cause more incremental losses due to the increased power flow in the lines. Line losses are calculated as $P_{loss} = R_{line} \times I^2$; thus, at higher loading conditions, the same attack load will result in more losses. It is worth noting that under the different attack scenarios, the total incremental line losses were almost equal. This is due to the fact that the total load of the different attack scenarios is the same and that we do not have any extra long transmission lines that will have significantly different losses under different attack load distributions. The normal and incremental losses are demonstrated in Figure 10(a). The no-attack losses under the different loading conditions were 3.1 MW (off-peak), 3.4 MW (average), and 4.3 MW (peak), which led to an increase of 16.13% at off-peak loading conditions, 17.65% during average loading conditions, and 18.6% during peak conditions. Thus, this simulation clearly demonstrates that the attack impact on system losses is amplified when the power demand is the



(a) Transmission losses (b) Attack-related costs

Fig. 10. Incurred (a) transmission losses and (b) costs due to various attack scenarios.

highest, which also coincides with the time during which EV connection to the chargers is the highest.

In the case of attacks against generation cost, each attack scenario differs based on the optimal power dispatch performed by the utility to reduce the overall cost. Figure 10(b) presents the total generation cost of the system when no attack occurs and during the three attack scenarios mentioned above. As Figure 10(b) demonstrates, the total added cost due to the attack is higher during peak loading conditions across all attacks. More importantly, the proportional attack scenario caused the highest extra cost. To put things into context, the no-attack cost at off-peak, average, and peak loading conditions was $14,545.28/hour, $16,009.39/hour, and $18,438.10/hour, respectively. The added cost due to the proportional attack was $1,423.83/hour under off-peak conditions, $1,426.45/hour under average conditions, and $1,451.95/hour under peak conditions. This demonstrates how an attacker can force the utility to increase its generation and incur extra costs.

One aspect not present in the simulation was the usage of peak generation units. This was left out due to the absence of these units in the Glover grid in Figure 9(a). These units are usually fast-ramping units used by utilities and power grid operators during peak hours when the large baseline generation units do not have sufficient capacity to supply all the load. These peak generation units are usually operated for a few hours a day only due to their high operation cost. This means that if the attack occurs at a time when peak generators are being utilized, the extra cost would be higher. Another aspect of repeated long-term attack worth mentioning is that mass charging attacks, especially at peak hours, will cause extra transformer loading. This extra loading would reduce its lifetime and would require more frequent maintenance intervals, causing extra maintenance costs.

An attacker with a long-term target of causing the utility to incur extreme losses can repeat the hijacking of charging sessions over long periods of time. For instance, launching the above attack for 1 hour during peak times every day for an entire year will create an extra generation cost totaling $529,962 for the utility based on the Glover grid and the generators' cost functions. To put things into a better context, scaling this attack up to the NSW grid will cause a $4,615,967 extra cost for the utility per year. To this end, an attacker might choose to compromise a smaller number of EV charging sessions and choose different sets of EVs every day to remain stealthier and still cause millions of dollars of losses to the utility in extra generation costs.

Mass Drop in Load. The attacker can also initiate a different type of attack consisting of a mass drop in load. This attack can be initiated through the 19 mobile applications that allow the attacker to control both the start and stop of the charging session. The attacker would first start by gradually increasing the EV charging load in the grid in a manner that does not cause any stability issues to the grid. Once the attackers achieve the desired botnet size that is concurrently charging, they will initiate the mass drop in load. Figure 11 demonstrates the impact of a 40 MW drop in EV load equivalent to 3,636 EVs charging at 11 kW EVCSs that stopped charging simultaneously. It can be observed how the frequency initially spikes to 61 Hz before the grid eventually regulates back

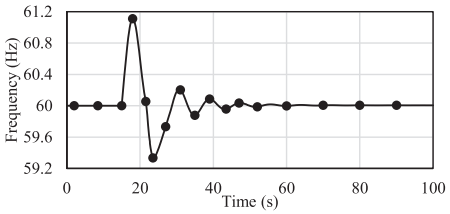


Fig. 11. Mass drop in load attack impact.

to the normal operating point of 60 Hz after 30 s of oscillations. A larger drop in load would result in larger-frequency deviations, but eventually the system would be able to recover from such an event.

Overloading and Tripping Transmission Lines. Another type of impact that might be desired by the attacker is causing line overloading and tripping by crafting a targeted attack against the grid. This attack has more severe and immediate consequences since it can leave consumers without electricity. In the previous set of attacks, some lines got highly loaded, but none of them reached an overloaded state. The same EV load, however, can be used by attackers with topology knowledge to target certain lines in order to cause cascading line failure. The attacker will only require knowledge of the topology and estimate values of the loads and power flows but not the line parameters. This information can be found online and in multiple public access databases and websites.

To simulate such attacker behavior, we targeted bus 4 and bus 5 with a synchronized 20 MW and 64 MW EV charging attack, respectively. This attack overloaded and tripped the line connecting bus 1 and bus 2, after which multiple lines would be overloaded and tripped. In total, seven lines would trip successively in the order shown in Figure 9(b). The successive line tripping would lead to islanding each of buses 1, 3, 4, and 5. While the load at bus 4 will be supplied by power from the generator at the same bus, the loads at bus 3 and 5 will lose their power supply and thus the grid will lose a total of 280 MW, which represents a loss of electricity to 35% of the consumers.

Power Grid Instability. Another attack that takes advantage of load manipulation is an oscillatory load attack that can impact the frequency stability of the power grid. This attack revolves around the concept of creating a demand surge to cause a frequency drop on the grid followed by a drop in demand to cause the frequency to overshoot. In the first step, the attacker will use the compromised accounts and hijack charging sessions to initiate mass charging to increase the power load. This extra power load would create an imbalance between the increased load and the generated power, causing the generators to slow down, resulting in a frequency drop. The second step of this attack happens when the system starts its recovery. The attacker would switch off the compromised charging sessions, initiated in the first step, to cause a frequency increase that is amplified by the operator's effort to increase the speed in response to the initial step. The attacker would then alternate between these steps for the desired duration. The impact can be amplified by launching the attack when the system has lower inertia due to the presence of a high share of renewable energy resources.

Given the dependence of the grid's transient behavior on the generator and turbine models, we utilized the automatic control models common to studies similar to ours. It is important to note that the utilization of different control models will change the exact values of the simulation, but the general shape and behavior remain the same. This demonstrates that the attacks can be successful under different conditions, but their magnitudes might need to be scaled based on the different conditions to achieve the desired impact. In our study, we used the machine model "GENSAL," the exciter model "IEEE T1," and the turbine governor model "IEEE G2."

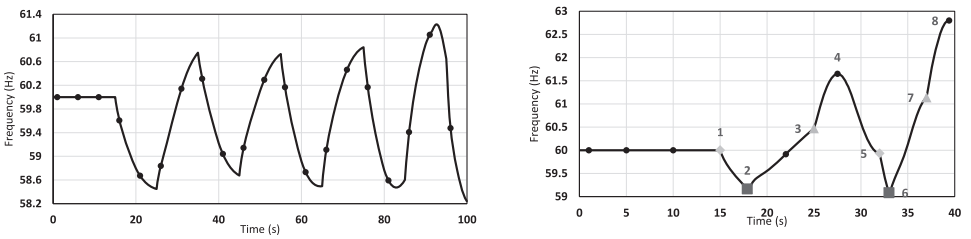


Fig. 12. Frequency behavior over time (a) without load shedding and (b) with load shedding.

The oscillatory load attack is simulated against the grid in Figure 9(a) by initiating the oscillatory load behavior described above on buses 3 and 5. The attack is initiated by starting a mass charging session equivalent to 20 MW (at time t = 15 s) and stopping it at t = 25 s after the system starts to increase generator speed to compensate and the frequency starts to rise. This charging and stopping behavior is repeated periodically every 10 seconds while increasing the attack load at each bus by 5.5 MW every cycle. The frequency behavior of the grid that results after such an attack is demonstrated in Figure 12(a), where we can see the frequency fluctuation due to the oscillatory load behavior. The importance of this attack is that it does not require huge loads to cause the frequency fluctuations depicted in Figure 12(a). Even when the compromised EV numbers are much less than in the example above, a sustained oscillation will hinder the system's return-to-normal operation. Sustaining this attack would damage the turbines due to the constant acceleration and deceleration.

In the previous example, we assumed no grid protection mechanism was used by the utility and that the attacker followed a semi-naive approach in which the attack period is predetermined and does not change as a result of actual conditions on the grid. In this iteration of the attack, we assume the utility will utilize load shedding when the frequency drops below preset thresholds. The threshold that is violated in the attack is the 59.3 Hz threshold, after which the utility will immediately disconnect 5% of the total load in order to compensate for the fast dropping frequency [63]. This utility behavior is depicted in Figure 12(b) by the squares at time t = 17.9s and t = 33s. The behavior depicted in Figure 12(b) is a response to a more advanced oscillatory load attack requiring the attacker to know and observe the actual grid response to tune the attack load and period. The attacker and utility interaction at every step is summarized in Table 3. Another important aspect of such an attack is the attackers' ability to cause devastating impacts on the grid and very large-frequency deviation with only a small portion of the load required in all of the previous attack scenarios. As a comparison, this oscillatory load attack was able to force the grid to shed part of its load and eventually reach a frequency of 62.8 Hz using 20 MW compared to the 61 Hz frequency reached after the 40 MW drop in load attack.

An extension of the oscillatory load attack can be achieved by utilizing the reverse power discharge through the V2G functionality similar to work performed in [64]. By initiating this V2G at the instance we stop the mass charging, the attacker can cause a larger-frequency spike. It is worth mentioning that by instilling V2G capabilities in mobile applications, the adversary can then utilize it to increase the oscillatory attack effect on the grid.

6 MITIGATION AND RECOMMENDED COUNTERMEASURES

The following section provides recommendations for hardening the EVCS ecosystem by reducing the attack surface and addressing the discovered vulnerabilities. First, we provide suggestions

Table 3. Attack Scenario Description and Impact

#	Time(s)	System State	Action	Action By	Impact (State Change)
1	15	System is operating normally	Total attack load of 40 MW initiated	Attacker	System frequency starts dropping
2	17.9	Frequency drops below 59.3	5% load shedding	Utility	5% of total consumers lose electricity. System frequency starts rising
3	25	The frequency peaks and is regulated by the automatic generator control	Turning off all compromised charging sessions	Attacker	System frequency spikes
4	27.5	The frequency starts dropping due to the automatic generation control	Automatic action of generation control system, "no human intervention"	Automatic	System frequency is being reduced to stabilize the system
5	32	The frequency was reduced by the automatic generation control	Total attack load of 80 MW initiated	Attacker	System frequency starts dropping faster than step 1 due to the larger attack load and the reduced generation after load shedding
6	33	Frequency drops below 59.3	5% load shedding	Utility	Additional 5% of total consumers lose electricity (10% total). System frequency starts rising
7	37	The frequency peaks and is regulated by the automatic generator control	Turning off all compromised charging sessions	Attacker	Causes a larger spike in frequency than step 3 since the EV load that was turned off is larger than that of step 3
8	39.4	Frequency exceeds 61.8 Hz [63]	Generators should be tripped instantaneously	Utility	Sequential generator tripping until system frequency stabilizes
>8	>39.4	Utility trips generators immediately. The system inertia drops	The attack impact is larger, causing more tripping	Attacker	As more generators are tripped, the system reaches a state of blackout

based on industry best practices for securing mobile applications. We then provide suggestions based on the unique properties found in the EV ecosystem.

To assist in mitigating automated attacks, mobile platforms should try to detect bot behavior. To keep malicious software from engaging in abusive behaviors, charging platforms should utilize reCAPTCHA [65] or other similar services that use an advanced risk analysis engine and adaptive challenges. The implementation of reCAPTCHA helps in detecting automated scraping, the creation of synthetic compromised accounts, and automated behaviors. This will not only hinder the attacker from performing the attack but also hinder the attacker from utilizing mobile applications to perform reconnaissance on the EVCSs and their behavior, which is an integral step in preparation for a wide-scale attack against the grid.

While this could hinder the attacker, it does not prevent the adversary from utilizing the lack of end-to-end authentication. Thus, to mitigate the consequences of such vulnerability, we recommend implementing a mechanism for verifying user ownership over the EV. Users should be coupled with their EVs from the beginning, making them the central authority that controls any action performed on their specific EV. Using this, we would add security by design in the interactions between the entities in the EV ecosystem [66]. Each EV has a unique **Vehicle Identification Number (VIN)** that can be used to identify during the creation of the mobile application account in order to verify that this mobile application user actually has a physical EV. The mobile application should later ask the users to enter the VIN of their EV when they attempt to initiate a charging session. This will ensure that only authorized EVs are charged. This allows the creation of a one-one relationship between the vehicle and the owner. This would require collaboration between different counterparts such as charging station vendors and vehicle manufacturers.

To ensure that the decision-making is distributed and more robust against cyber tampering, we recommend that the EVCS needs to check if a connected vehicle has the same VIN as the one that sent in the charging request. If the identifiers match, then the EVCS will start charging after verifying the end-to-end authentication. If no match was found, then it will return an error to the user. The adversary can no longer register to the platform without registering with a valid VIN. Furthermore, even if the adversary creates an account using a legitimate VIN, the adversary will not be able to know the specific VIN of the EV connected at the targeted EVCS. Thus, by enabling a strict end-to-end authentication, the attacker cannot be associated with someone else's EV. This

would prevent the adversary from launching large-scale attacks against the grid. Additionally, EV charging can be restricted to users who are in the vicinity of their EVs and the EVCS. Some EVs have NFC chips installed that can be used to initiate charging. The mobile application can use NFC to communicate with the EV and verify that the user is in close proximity to their EV. The mobile application can use GPS-based location information to confirm that the user is in proximity to the EVCS. This will ensure that only authorized users who are physically present near the EV and EVCS can initiate charging. We understand that such restrictions might affect the usability and commercialization of the mobile application. For example, when Alice lends her car to her son, he should be allowed to charge the EV. Thus, we suggest creating an authorization list where Alice can add a list of verified mobile application accounts that are authorized to control the charging of the EV. By implementing the suggested mitigation methods, the security of the EV charging mobile application will be hardened and the attacks discussed above will be rendered extremely difficult to perform.

7 CONCLUSION

In this work, we explored the security of the EV charging ecosystem by focusing on the understudied EV charging mobile application as a main attack surface. We studied the interactions of the mobile application with other components to understand its remote control functionality over the charging stations. Our analysis of the identified interactions and communications unveiled critical vulnerabilities that allow remote adversaries to gain control over the charging operations and perform DDoS attacks by preventing legitimate users from using the charging equipment. Moreover, while we demonstrate the feasibility of exploiting existing EV charging mobile applications' vulnerabilities to hijack charging sessions, we discuss the implications of such attacks on various stakeholders within the EV charging ecosystem. Specifically, we discuss the impact of wide-scale remote attacks on the underlying critical infrastructure (i.e., the power grid) and show that an attacker can utilize a botnet of adversarial accounts on those vulnerable mobile applications to cripple the operations of the power grid. Finally, while we discuss attack implications against EV consumers, we also recommend countermeasures to secure the infrastructure and impede adversaries from performing reconnaissance and launching remote attacks using compromised accounts. In future work, we aim at studying different distributed blockchain solutions to mitigate these design flaws while minimizing integration overhead for the operators.

REFERENCES

[1] Helen Regan. 2020. China pledges to go carbon neutral by 2060. https://www.cnn.com/2020/09/22/china/xi-jinping-carbon-neutral-2060-intl-hnk/index.html

[2] Linda Gyulai. 2020. Montreal's climate plan includes ban on non-electric cars downtown by 2030. https://montrealgazette.com/news/local-news/montreal-releases-climate-plan-including-ban-on-non-electric-cars-downtown-by-2030

[3] Charles Riley. 2021. Europe aims to kill gasoline and diesel cars by 2035. https://edition.cnn.com/2021/07/14/business/eu-emissions-climate-cars/index.html

[4] Natural Resources Canada. 2021. Government of Canada. https://www.nrcan.gc.ca/energy-efficiency/transportation-alternative-fuels/zero-emission-vehicle-infrastructure-program/21876

[5] Samrat Acharya, Yury Dvorkin, Hrvoje Pandžić, and Ramesh Karri. 2020. Cybersecurity of smart electric vehicle charging: A power grid perspective. *IEEE Access* 8 (2020), 214434–214453.

[6] Mohammad Ali Sayed, Ribal Atallah, Chadi Assi, and Mourad Debbabi. 2021. Electric vehicle attack impact on power grid operation. *International Journal of Electrical Power & Energy Systems* 112 (2021), 107784. DOI: http://dx.doi.org/10.1016/j.ijepes.2021.107784

[7] Tony Nasr, Sadegh Torabi, Elias Bou-Harb, Claude Fachkha, and Chadi Assi. 2021. Power jacking your station: In-depth security analysis of electric vehicle charging station management systems. *Computers & Security* (2021), 102511.

[8] R. Akhras, W. El-Hajj, M. Majdalani, H. Hajj, R. Jabr, and K. Shaban. 2020. Securing smart grid communication using ethereum smart contracts. *International Wireless Communications and Mobile Computing (IWCMC'20)*. Limassol, Cyprus, 1672–1678. DOI: 10.1109/IWCMC48107.2020.9148345

[9] Cristina Alcaraz, Javier Lopez, and Stephen Wolthusen. 2017. OCPP protocol: Security threats and challenges. *IEEE Transactions on Smart Grid* 8, 5 (2017), 2452–2459.
[10] Juan E. Rubio, Cristina Alcaraz, and Javier Lopez. 2018. Addressing security in OCPP: Protection against man-in-the-middle attacks. In *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS'18)*. IEEE, 1–5.
[11] Hossam ElHussini, Chadi Assi, Bassam Moussa, Ribal Atallah, and Ali Ghrayeb. 2021. A tale of two entities: Contextualizing the security of electric vehicle charging stations on the power grid. *ACM Transactions on Internet of Things* 2, 2 (2021), 1–21.
[12] Joseph Antoun, Mohammad Ekramul Kabir, Bassam Moussa, Ribal Atallah, and Chadi Assi. 2020. A detailed security assessment of the EV charging ecosystem. *IEEE Network* 34, 3 (2020), 200–207.
[13] Open Charge Alliance. 2021. OCPP 2.0.1, protocols, home. https://www.openchargealliance.org/protocols/ocpp-201/
[14] Richard Baker and Ivan Martinovic. 2019. Losing the car keys: Wireless phy-layer insecurity in EV charging. In *28th USENIX Security Symposium (USENIX Security'19)*. 407–424.
[15] Y. S. Wong, K. T. Chau, and C. C. Chan. 2006. Battery sizing for plug-in hybrid electric vehicles. *Journal of Asian Electric Vehicles* 4, 2 (2006), 899–904.
[16] Kaspersky Lab. 2021. How do fuel cell electric vehicles work Using hydrogen? https://afdc.energy.gov/vehicles/how-do-fuel-cell-electric-cars-work
[17] Sklyar Dmitry. 2018. ChargePoint Home Security Research. https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2018/12/13084354/ChargePoint-Home-security-research_final.pdf
[18] Juan E. Rubio, Cristina Alcaraz, and Javier Lopez. 2018. Addressing security in OCPP: Protection against man-in-the-middle attacks. In *2018 9th IFIP International Conference on New Technologies, Mobility and Security*. 1–5.
[19] Omniyah Gul M. Khan, Ehab El-Saadany, Amr Youssef, and Mostafa Shaaban. 2019. Impact of electric vehicles botnets on the power grid. In *2019 IEEE Electrical Power and Energy Conference (EPEC'19)*. IEEE, 1–5.
[20] Kristien Clement-Nyns, Edwin Haesen, and Johan Driesen. 2009. The impact of charging plug-in hybrid electric vehicles on a residential distribution grid. *IEEE Transactions on Power Systems* 25, 1 (2009), 371–380.
[21] Niels Leemput, Frederik Geth, Juan Van Roy, Annelies Delnooz, Jeroen Büscher, and Johan Driesen. 2014. Impact of electric vehicle on-board single-phase charging strategies on a flemish residential grid. *IEEE Transactions on Smart Grid* 5, 4 (2014), 1815–1822.
[22] Anamika Dubey and Surya Santoso. 2015. Electric vehicle charging on residential distribution systems: Impacts and mitigations. *IEEE Access* 3 (2015), 1871–1893.
[23] Hugo Morais, Tiago Sousa, Zita Vale, and Pedro Faria. 2014. Evaluation of the electric vehicle impact in the power demand curve in a smart grid environment. *Energy Conversion and Management* 82 (2014), 268–282.
[24] Soroush Shafiee, Mahmud Fotuhi-Firuzabad, and Mohammad Rastegar. 2013. Investigating the impacts of plug-in hybrid electric vehicles on power distribution systems. *IEEE Transactions on Smart Grid* 4, 3 (2013), 1351–1360.
[25] Elif Ustundag Soykan, Mustafa Bagriyanik, and Gurkan Soykan. 2021. Disrupting the power grid via EV charging: The impact of the SMS Phishing attacks. *Sustainable Energy, Grids and Networks* 26 (2021), 100477.
[26] Wei Zhou, Yan Jia, Yao Yao, Lipeng Zhu, Le Guan, Yuhang Mao, Peng Liu, and Yuqing Zhang. 2019. Discovering and understanding the security hazards in the interactions between IoT devices, mobile apps, and clouds on smart home platforms. In *28th USENIX Security Symposium (USENIX Security'19)*. 1133–1150.
[27] S. O'Dea. 2021. Mobile OS market share 2021. https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/
[28] 2021. apktool - a tool for reverse engineering 3rd party, closed, binary android apps. https://ibotpeaches.github.io/Apktool/
[29] KALI. 2021. Dex2jar: Kali linux tools. https://www.kali.org/tools/dex2jar/
[30] KALI. 2021. JD-Gui: Kali linux tools. https://www.kali.org/tools/jd-gui/
[31] MobSF. 2021. Mobile security framework (mobsf). https://github.com/MobSF/Mobile-Security-Framework-MobSF
[32] Pkumza. 2021. pkumza/LiteRadar: Lite version of LibRadar. https://github.com/pkumza/LiteRadar
[33] Shengqian Yang, Dacong Yan, Haowei Wu, Yan Wang, and Atanas Rountev. 2015. Static control-flow analysis of user-driven callbacks in Android applications. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Vol. 1. IEEE, 89–99.
[34] Tanzirul Azim and Iulian Neamtiu. 2013. Targeted and depth-first exploration for systematic testing of Android apps. In *Proceedings of the 2013 ACM SIGPLAN International Conference on Object Oriented Programming Systems Languages & Applications*. 641–660.
[35] GPS Joystick Guide-the app ninjas. n. d. http://gpsjoystick.theappninjas.com/
[36] SHIVAM says. 2021. Virtualxposed APK 0.20.3 download latest in 2021 [official]. https://virtualxposed.com/
[37] Ac-Pm. n.d. AC-PM/inspeckage: Android Package Inspector - Dynamic Analysis with API hooks, start unexported activities and more. (Xposed module). https://github.com/ac-pm/Inspeckage

[38] Jens Schmutzler, Claus Amtrup Andersen, and Christian Wietfeld. 2021. Burp Suite - Application Security Testing Software. https://portswigger.net/burp
[39] Jens Schmutzler, Claus Amtrup Andersen, and Christian Wietfeld. 2013. Evaluation of OCPP and IEC 61850 for smart charging electric vehicles. *World Electric Vehicle Journal* 6, 4 (2013), 863–874.
[40] Jens Schmutzler, Christian Wietfeld, and Claus Amtrup Andersen. 2012. Distributed energy resource management for electric vehicles using IEC 61850 and ISO/IEC 15118. In *2012 IEEE Vehicle Power and Propulsion Conference*. IEEE, 1457–1462.
[41] Communication apis for SMS, Voice, Video & Authentication. [n. d.] https://www.twilio.com/
[42] Yu-Wei Chung, Behnam Khaki, Tianyi Li, Chicheng Chu, and Rajit Gadh. 2019. Ensemble machine learning-based algorithm for electric vehicle user behavior prediction. *Applied Energy* 254 (2019), 113732.
[43] Ahmad Almaghrebi, Subhaditya Shom, Fares Al Juheshi, Kevin James, and Mahmoud Alahmad. 2019. Analysis of user charging behavior at public charging stations. In *2019 IEEE Transportation Electrification Conference and Expo (ITEC'19)*. IEEE, 1–6.
[44] Tai-Yu Ma and Sébastien Faye. 2022. Multistep electric vehicle charging station occupancy prediction using hybrid LSTM neural networks. *Energy* vol 244, part B (2022), 123217.
[45] Zachary J. Lee, Tongxin Li, and Steven H. Low. 2019. ACN-data: Analysis and applications of an open EV charging dataset. In *Proceedings of the 10th ACM International Conference on Future Energy Systems*. 139–149.
[46] Appium. 2021. Automation for Apps. https://appium.io/
[47] Saleh Soltan, Prateek Mittal, and H. Vincent Poor. 2018. BlackIoT: IoT botnet of high wattage devices can disrupt the power grid. In *27th USENIX Security Symposium (USENIX Security'18)*. 15–32.
[48] K. Sarieddine, M. A. Sayed, D. Jafarigiv, R. Atallah, M. Debbabi, and C. Assi. 2023. A real-time cosimulation testbed for electric vehicle charging and smart grid security. In *IEEE Security & Privacy* 21, 4 (2023), 74–83. DOI: 10.1109/MSEC.2023.3247374
[49] Yosra Fraiji, Lamia Ben Azzouz, Wassim Trojet, and Leila Azouz Saidane. 2018. Cyber security issues of Internet of electric vehicles. In *2018 IEEE Wireless Communications and Networking Conference (WCNC'18)*. IEEE, 1–6.
[50] Richard M. Pratt and Thomas E. Carroll. 2019. Vehicle charging infrastructure security. In *2019 IEEE International Conference on Consumer Electronics (ICCE'19)*. IEEE, 1–5.
[51] California ISO. 2021. Demand Trend. https://www.caiso.com/TodaysOutlook/Pages/default.aspx
[52] Australian Energy Market Operator (AEMOO). 2021. Demand Trend. https://aemo.com.au/en
[53] Guido Cavraro, Andrey Bernstein, Vassilis Kekatos, and Yingchen Zhang. 2019. Real-time identifiability of power distribution network topologies with limited monitoring. *IEEE Control Systems Letters* 4, 2 (2019), 325–330.
[54] Seyed Iman Taheri, M. B. C. Salles, and N. Kagan. 2019. A new modified TLBO algorithm for placement of AVRs in distribution system. In *2019 IEEE PES Innovative Smart Grid Technologies Conference-Latin America*. IEEE, 1–6.
[55] Guido Cavraro and Vassilis Kekatos. 2019. Inverter probing for power distribution network topology processing. *IEEE Transactions on Control of Network Systems* 6, 3 (2019), 980–992.
[56] Keith Moffat, Mohini Bariya, and Alexandra Von Meier. 2019. Unsupervised impedance and topology estimation of distribution networks—limitations and tools. *IEEE Transactions on Smart Grid* 11, 1 (2019), 846–856.
[57] Anandini Gandluru, Shiva Poudel, and Anamika Dubey. 2019. Joint estimation of operational topology and outages for unbalanced power distribution systems. *IEEE Transactions on Power Systems* 35, 1 (2019), 605–617.
[58] Deepjyoti Deka, Michael Chertkov, and Scott Backhaus. 2019. Topology estimation using graphical models in multi-phase power distribution grids. *IEEE Transactions on Power Systems* 35, 3 (2019), 1663–1673.
[59] Keith Moffat, Mohini Bariya, and Alexandra Von Meier. 2020. Real time effective impedance estimation for power system state estimation. In *2020 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT'20)*. IEEE, 1–5.
[60] Prabha Kundur. 2007. Power system stability. *Power System Stability and Control*, (3rd edition), (2007), 7–1.
[61] J. Duncan Glover, Mulukutla S. Sarma, and Thomas Overbye. 2012. *Power System Analysis & Design, SI Version*. Cengage Learning.
[62] PowerWorld. 2021. https://www.powerworld.com/
[63] Bing Huang, Alvaro A. Cardenas, and Ross Baldick. 2019. Not everything is dark and gloomy: Power grid protections against IoT demand attacks. In *28th USENIX Security Symposium (USENIX Security'19)*. 1115–1132.
[64] M. E. Kabir, M. Ghafouri, B. Moussa, and C. Assi. 2021. A two-stage protection method for detection and mitigation of coordinated EVSE switching attacks. In *IEEE Transactions on Smart Grid* 12, 5 (2021), 4377–4388. DOI: 10.1109/TSG.2021.3083696
[65] ReCAPTCHA. 2021. https://www.google.com/recaptcha/about/
[66] Smartcar Âů API platform for Connected Car Data. n.d. https://smartcar.com/

Received 26 February 2023; revised 24 May 2023; accepted 1 July 2023

HPRoP: Hierarchical Privacy-preserving Route Planning for Smart Cities

FRANCIS TIAUSAS, Nara Institute of Science and Technology (NAIST), Japan
KEIICHI YASUMOTO, Nara Institute of Science and Technology (NAIST), Japan and RIKEN Center for Advanced Intelligence Project (AIP), Japan
JOSE PAOLO TALUSAN, Vanderbilt University, USA
HAYATO YAMANA, Waseda University, Japan
HIROZUMI YAMAGUCHI, Osaka University, Japan
SHAMEEK BHATTACHARJEE, Western Michigan University, USA
ABHISHEK DUBEY, Vanderbilt University, USA
SAJAL K. DAS, Missouri University of Science and Technology, USA

Route Planning Systems (RPS) are a core component of autonomous personal transport systems essential for safe and efficient navigation of dynamic urban environments with the support of edge-based smart city infrastructure, but they also raise concerns about user route privacy in the context of both privately owned and commercial vehicles. Numerous high-profile data breaches in recent years have fortunately motivated research on privacy-preserving RPS, but most of them are rendered impractical by greatly increased communication and processing overhead. We address this by proposing an approach called Hierarchical Privacy-Preserving Route Planning (HPRoP), which divides and distributes the route-planning task across multiple levels and protects locations along the entire route. This is done by combining Inertial Flow partitioning, Private Information Retrieval (PIR), and Edge Computing techniques with our novel route-planning heuristic algorithm. Normalized metrics were also formulated to quantify the privacy of the source/destination points (*endpoint location privacy*) and the route itself (*route privacy*). Evaluation on a simulated road network showed that HPRoP reliably produces routes differing only by $\leq 20\%$ in length from optimal shortest paths, with completion times within ~ 25 seconds, which is reasonable for a PIR-based approach. On top of this, more than half of the produced routes achieved near-optimal endpoint location privacy (~ 1.0) and good route privacy (≥ 0.8).

This work was supported by R&D for Trustworthy Networking for Smart and Connected Communities, Commissioned Research of National Institute of Information and Communications Technology (NICT), JSPS KAKENHI Grant Numbers JP21H03431 and JP19H05665, and National Science Foundation through award numbers 1647015, 1818901, CNS-2030611, CNS-1818942, SaTC-2030624, SaTC-2030611, and ECCS-2319995.
Authors' addresses: F. Tiausas, Nara Institute of Science and Technology (NAIST), Ikoma, 630-0192, Nara, Japan; e-mail: tiausas.francis_jerome.ta5@is.naist.jp; K. Yasumoto, Nara Institute of Science and Technology (NAIST), Ikoma, 630-0192, Nara, Japan, and RIKEN Center for Advanced Intelligence Project (AIP), Tokyo, 103-0027, Japan; e-mail: yasumoto@is.naist.jp; J. P. Talusan and A. Dubey, Vanderbilt University, 2201 West End, Nashville, TN 37235-1829 USA; e-mails: {jose.paolo.talusan, abhishek.dubey}@vanderbilt.edu; H. Yamana, Waseda University, 1-104 Totsukamachi, Shinjuku, Tokyo, 169-8050, Japan; e-mails: yamana@waseda.jp; H. Yamaguchi, Osaka University, 1-1 Yamadaoka, Suita, Osaka, 565-0871, Japan; e-mail: h-yamagu@ist.osaka-u.ac.jp; S. Bhattacharjee, Western Michigan University, 1903 W Michigan Ave., Kalamazoo, MI 40998-5466, USA; e-mail: shameek.bhattacharjee@wmich.edu; S. K. Das, Missouri University of Science and Technology, 500 W. 15th Street, Rolla, MO 65409, USA; e-mail: sdas@mst.edu.

2378-962X/2023/10-ART27 $15.00
https://doi.org/10.1145/3616874

CCS Concepts: • **Security and privacy** → *Privacy-preserving protocols*; *Domain-specific security and privacy architectures;*

Additional Key Words and Phrases: Route planning services, location privacy, route-planning algorithms

ACM Reference format:
Francis Tiausas, Keiichi Yasumoto, Jose Paolo Talusan, Hayato Yamana, Hirozumi Yamaguchi, Shameek Bhattacharjee, Abhishek Dubey, and Sajal K. Das. 2023. HPRoP: Hierarchical Privacy-preserving Route Planning for Smart Cities. *ACM Trans. Cyber-Phys. Syst.* 7, 4, Article 27 (October 2023), 25 pages.
https://doi.org/10.1145/3616874

1 INTRODUCTION

Route Planning Services (RPS) are web-based applications that can calculate routes between two chosen points in a road network and are indispensable navigational aids for commuters, vehicle operators, autonomous vehicles, and so on. However, these same routes can reveal *points of interest* [15] that may contain users' places of work and residence, in addition to highly sensitive information about their political, sexual, or religious tendencies [30]. Furthermore, this exposes users to risks of targeted criminal acts, mass surveillance, discrimination, and so on. The rise in high-profile data breaches over the past decade has highlighted the importance of protecting user location (and route) data and has spurred many recent works focusing on adding privacy-preserving mechanisms to RPS.

In the RPS context, privacy primarily entails keeping users' origin, destination, and route information from being acquired by untrusted entities. It exists side-by-side with other important **Quality of Service (QoS)** metrics such as Utility (e.g., the accuracy of the routes) and Performance (e.g., the response time of the RPS), which affect whether or not an RPS would gain widespread adoption. Finding a good balance between these metrics is crucial for any privacy-preserving RPS. For instance, privacy-preserving protocols for querying road traffic data have been developed for **Vehicle Ad-hoc Networks (VANETs)**, but these either offload the computation cost of route planning onto the vehicle itself [24, 25, 39] or an external trusted entity [3, 4]. The latter approaches are already done by modern RPS, while the former are not web-based RPS at all. Structured encryption schemes [17, 26, 37] provide privacy-protection for both user queries and road network data while also being relatively lightweight and efficient but at the cost of inherently leaking some query-related information that are detrimental to the privacy of user routes.

Aside from these, there are also works that utilize **Private Information Retrieval (PIR)** [7], since this protocol has strong privacy guarantees. In an RPS, their efficiency depends mostly on the routing algorithm but are often considered too computationally heavy to be used on very large databases—such as the road network graph of a large city. As such, only a few examples of PIR-based RPS have been developed over the past decade. One approach [35] compresses road network graphs in a novel PIR-queryable manner but results in longer pre-processing and query response times. The other approach [28] partitions the road network into disjoint subgraphs and these are individually retrieved via PIR to inform a local routing algorithm on the user's device. This results in longer route completion times, since many PIR queries are needed to complete the route. Both approaches clearly entail a significant degradation of QoS, which dissuades most mainstream RPS from experimenting with and adopting them.

Our approach aims to create a RPS that addresses the aforementioned issues by fulfilling the following three objectives: (1) produce close-to-optimal users' routes, (2) provide strong privacy guarantees for users' route data, and (3) maintain an adequate level of performance by minimizing processing and communication overhead as much as possible. The approach involves two phases.

In the *pre-processing phase*, a graph partitioning technique is used to hierarchically divide the road network into balanced partitions. Routes within each partition are then pre-computed and stored in separate databases, and the same is done for an additional set of routes between neighboring partitions. In the *routing phase*, a novel hierarchical heuristic algorithm on the user's device privately obtains partial routes from the different partition databases using PIR and iteratively combines them to create a progressively finer route. We named the proposed approach, ***Hierarchical Privacy-Preserving Route Planning*** **(HPRoP)**, which makes the following key contributions:

- An RPS that uses a hierarchical partitioning of the road network with a novel hierarchical route-planning algorithm to produce routes with good *optimal route approximation* while maintaining strong privacy guarantees and low route completion times,
- A pair of privacy metrics—*endpoint location privacy* and *route privacy*—that aim to be general enough to be applicable to other privacy-preserving RPS while also accounting for specific characteristics of PIR-based RPS (such as providing strong privacy guarantees between routes in the same database), and
- A comprehensive evaluation of the Utility, Privacy, and Performance of the proposed RPS on a simulated road network based in Osaka City against two PIR-based baseline approaches.

Note that the baseline approaches mentioned above were also developed solely for this article to address the lack of recent PIR-based route-planning approaches that HPRoP can be compared against. However, we do not count them as separate contributions, since they are primarily used as an evaluation tool. The rest of this article is organized as follows: Section 2 presents a summary of prior work related to privacy-preserving route planning. Section 3 discusses the mathematical models, assumptions, and other preliminaries. Section 4 presents the concept and intuition behind route planning under the constraints of PIR. Section 5 defines and discusses the privacy metrics used to evaluate the RPS. Section 6 presents the key ideas behind the approach itself and the details of the heuristic algorithm. Section 7 discusses the evaluation framework and the results. Section 8 concludes with a brief summary of the article and some notes on potential future work.

2 RELATED WORK

Most algorithms for calculating exact shortest paths require *knowledge* of the exact source and destination locations. Classical algorithms such as Dijkstra's and Bellman-Ford [10] calculate routes by repeatedly scanning connected vertices from some source point and assigning them weights until a path to the destination point is found. Modern algorithms improve upon this by leveraging unique properties of road networks. ALT [18] pre-computes distances to fixed landmarks and uses them as lower bounds to inform a bidirectional A* search [21]. *Contraction Hierarchies* [16] pre-processes the network graph to establish "shortcut edges," facilitating faster route calculation between distant points. *Customizable Route Planning* [11] uses the network graph's topology in their metric-independent hierarchical routing method. Regardless, deploying these on the server-side inevitably means that the user's origin and destination must be divulged so the final route can be computed. Meanwhile, deploying these on the client side means downloading large amounts of road network data and computing routes on more resource-constrained machines. In other words, the first case compromises privacy while the second degrades functionality.

Alternatively, algorithms for finding *approximate shortest paths* also exist, focusing on quickly obtaining short routes rather than finding the exact shortest ones. Point-to-point variants of these [8, 22, 23] were developed at a time when mobile computational power was very limited and have been outclassed by modern exact shortest path algorithms. Yet, these remain useful in **All-Pairs Shortest Path (APSP)** *distance oracles* [1, 34] for speeding up goal-directed route calculations.

Recent research on privacy-preserving route-planning techniques generally fall under three categories: (1) *Structured Encryption*-based, (2) *PIR*-based, and (3) *Other encryption-based* schemes. In addition, a number of schemes for privately querying road traffic information with applications to vehicle navigation systems also exist, but, since these either perform route planning on the vehicle itself [24, 25, 39] or an external trusted entity [3, 4], these works have been excluded here.

Structured Encryption [6] refers to techniques that allow data structures to be encrypted such that these can be queried later in a privacy-preserving manner. They are typically more efficient in terms of computation time and communication overhead at the cost of leaking a small amount of information about the data and the queries. The approaches in References [26, 37] use structured encryption to find the shortest distance between vertex-pairs in encrypted graphs. These, however, are only able to find shortest distance values instead of the actual shortest paths and are vulnerable to collusion between the storage and computing servers—which are essentially the same entity in the RPS context. In contrast, Reference [17] is able to retrieve the actual shortest paths on a single-server setup, which effectively eliminates the aforementioned issues. However, pre-computing the encrypted database for large sparse graphs (i.e., with $|V| \geq 10{,}000$ and $|E| \geq 30{,}000$) took upwards of 16.5 hours and produced very large files (around 4.4 GB), rendering it impractical for dynamic scenarios such as real-time route planning. Additionally, while all three have heavily constrained leakage profiles, some of the information they inherently leak may be detrimental to route privacy. For instance, the number of potential query elements (i.e., the database size) can be used to determine the specific subgraph of the road network graph being used. Query repetitions and total queries for route completion can be jointly analyzed across multiple sessions to deduce the actual route. Different approaches may also leak other information in addition to the ones mentioned here.

PIR [7] is a technique that allows remote databases to be queried in such a way that the retrieved element would not be revealed to the service provider or any third-party entity. PIR-based schemes, therefore, have slightly stronger privacy guarantees in that repeated queries and underlying database sizes are not leaked but have the disadvantage of much higher communication overhead. While most modern implementations [2, 19, 27, 29] have become very communication-efficient (i.e., up to $O(\sqrt{N})$), they remain impractical for accessing a database of APSP in a large city. This is because PIR schemes need to go through each individual element to avoid leaking information about the element being retrieved [5]. The approach in Reference [35] describes a method for compressing road network graphs via sign-decomposition combined with Yao's garbled circuits [36] and PIR to protect both user queries and said graph, giving it strong end-to-end privacy guarantees. However, it also has relatively long pre-processing and query response times, since the protocol must operate on compressed and encrypted data at all times. The approach in Reference [28] partitions the network graph into disjoint sections (i.e., each consisting of a separate subgraph) that can then be retrieved via PIR during local computation of a shortest path during the routing phase. While this requires minimal pre-processing time, the total route completion time remains rather long, since the locally run routing algorithm would need multiple PIR queries to complete a single route.

Other encryption-based schemes with much stronger privacy guarantees also exist, but they are also much less efficient than the previous two. For instance, Reference [38] allows users to request routes between arbitrary source and destination partitions, as well as within said partitions in a privacy-preserving manner using 1-of-n Oblivious Transfer [31]. However, the scheme needs to compute **All-Pairs of Shortest Paths (APSP)** for the aforementioned partitions during the routing phase, drastically slowing down query response times. Similarly, the work in Reference [14] uses Paillier's Encryption to privately query outgoing edge weights from vertices in the road network graph that, in turn, is used to inform a route-planning algorithm running locally on the

Table 1. Summary of Symbols Used in Section 3

Symbol	Description
$G = (V, E)$	Road network graph with road segments E and intersections V
$l_G(u, v)$	Length of a road segment having endpoints (u, v)
s, d	Source (s) and Destination (d) vertices
$r_G(s, d)$	Path/route between s and d
$L_G(r_G(s, d))$	Total length of path/route $r_G(s, d)$
$R_G(s, d)$	All possible paths between s and d
$\rho_G(s, d)$	*Shortest path* between s and d
$G_P = (P, C)$	Partition graph derived from G with the set of all partitions P and the connections between them C
$p = (V_p, E_p)$	A partition (i.e., a subgraph of G) consisting of road segments E_p and endpoints V_p within it
NB_{p_u}	Set of all neighboring partitions for partition p_u
ℓ	Partition level
$\mathcal{L}$	Maximum partition level ($0 \leq \ell \leq \mathcal{L}$)
P^ℓ	Subset of partitions in P at level ℓ
p_i^ℓ	A partition belonging under P^ℓ with a given index i
x_u	Representative vertex for the partition p_u
$c_{p_u p_v}$	A connection between partitions p_u and p_v through shortest path $\rho_G(x_u, x_v)$
$dist_G(u, v)$	*Shortest path* distance between u and v in G
$\mathcal{D}(p)$	Database of shortest paths for partition p
$C(\rho_0, \ldots, \rho_n)$	Arbitrary route combination heuristic
$r^*_G(u, v)$	Approximate shortest path between u and v
$\alpha(r^*_G(u, v))$	Optimal route approximation metric

user's own device. The scheme, unfortunately, has a very high communication overhead, since it has to make a separate query for every vertex that needs to be "scanned" by the routing algorithm.

As this work aims to achieve strong privacy guarantees for users' routes while also meeting utility and performance targets, PIR was chosen as the core privacy-preservation mechanism for the proposed approach. Unlike structured encryption, it does not leak information about query repetitions, which can potentially be analyzed by an adversary to distinguish between different route requests by the same user.

3 MODELS AND ASSUMPTIONS

This section presents the assumptions and mathematical models related to the road network graph, its corresponding partition graph, and the "approximate" routes that can be derived from the latter. Table 1 provides a summary of the symbols and notation used in this section.

3.1 Road Network Partitioning Model

A road network can be modelled as a directed graph $G = (V, E)$, where the edges E represent road segments, and the vertices V represent either road intersections or terminals. Each road segment $e \in E$ is a directed edge between two vertices $u, v \in V$ such that $e = (u, v)$, with parallel opposing lanes being represented by two directed edges going in opposite directions. The traversal cost for e is given by its length, $l_G(u, v)$. A *route* or *path* between two vertices $s, d \in V$ is defined as a sequence of vertices $r_G(s, d) = (v_1^{sd}, v_2^{sd}, \ldots, v_{n-1}^{sd}, v_n^{sd})$ where the first vertex $v_1^{sd} = s$ and the last vertex $v_n^{sd} = d$. The total length of $r_G(s, d)$ is given by $L_G(r_G(s, d)) = \sum_{i=1}^{|r_G(s,d)|-1} l_G(v_i^{sd}, v_{i+1}^{sd})$. Denoting *all* possible paths (between s and d) as $R_G(s, d)$, the *Shortest Path* is then:

$$\rho_G(s, d) = arg\ min_{r \in R_G(s,d)}\ L_G(r). \tag{1}$$

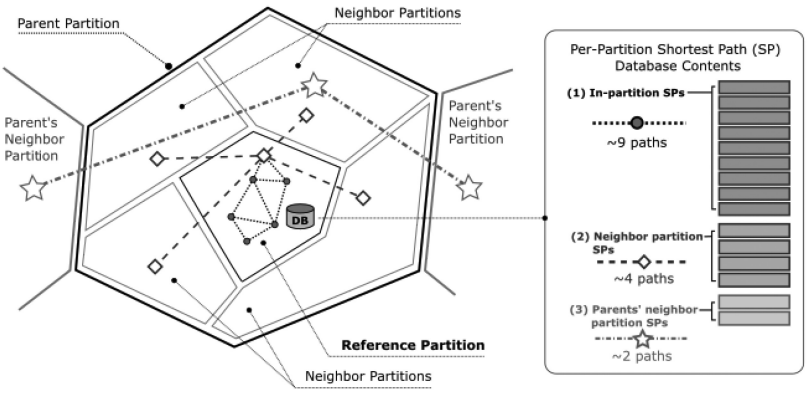


Fig. 1. Types of shortest paths stored by each partition.

Most modern route-planning algorithms can already deal with very large road networks, but typically do not incorporate route privacy protection mechanisms out of the box, as they tend to significantly increase processing and communication overhead, as mentioned in Section 2. However, if the route-planning task can be divided, then the additional processing cost incurred by the privacy mechanism can be distributed across multiple devices instead and ultimately improve RPS performance. This is done by first dividing the road network into different areas called *partitions*.

An arbitrary partitioning of the road network graph G is represented by a separate *partition graph* $G_P = (P, C)$, where the vertices P represent partitions and the edges C represent connections between them. Each *partition* $p \in P$ is a subgraph $p = (V_p, E_p)$ such that $V_p \subseteq V$ and $E_p \subseteq E$. Each connection $c_{p_u p_v} \in C$ is a shortest path $\rho_G(x_u, x_v)$ between the *representative vertices* $x_u, x_v \in V$ of any two neighboring partitions $p_u, p_v \in P$ where $x_u \in V_{p_u} \wedge x_v \in V_{p_v}$. Two partitions are considered "neighbors" if $\exists\ e = (u, v)$ s.t. $e \in E \wedge u \in V_{p_u} \wedge v \in V_{p_v}$. Finally, the set of all *neighboring* partitions for p_u is defined as $NB_{p_u} = \{p_v | p_v \in P \wedge (c_{p_u p_v} \in C \vee c_{p_v p_u} \in C)\}$.

Representative vertices are ideally chosen to minimize the shortest path distance to all other vertices in the partition. Vertices with high graph centrality are good candidates, but our preliminary experiments show that it is viable to simply choose the vertex closest to the geospatial average of the coordinates of each partition's vertices with no impact on routing performance. Thus, representative vertices were chosen via this method. It then follows that $l_{G_P}(p_u, p_v) = L_G(\rho_G(x_u, x_v))$. For simplicity, the shortest path distance between any $u, v \in G$ is represented by $dist_G(u, v)$, while $dist_{G_P}(p_u, p_v)$ refers to the shortest path distance between their *containing partitions*, $p_u, p_v \in P$.

Partitions are defined in a hierarchical manner with the maximum partition level being $\mathcal{L}$ and the subset of partitions for each level $\ell < \mathcal{L}$ denoted as $P^\ell \subseteq P$ such that $P^\ell \subset P^{\ell-1}$ for $\ell > 0$. A partition may also be defined as p_i^ℓ, where ℓ is the partition's level and i is the partition's index at that level given $P^\ell = \{p_0^\ell, p_1^\ell, \ldots, p_n^\ell\}$. The base level partition set P^0 contains only one partition/subgraph $p_0^0 = G$ at $\ell = 0$. Conversely, p_0^0 might be composed of several smaller partitions p_0^1, p_1^1, p_2^1, and p_3^1 at $\ell = 1$ such that all of them are subgraphs of p_0^0 and so on. For clarity, partition levels will henceforth be referred to by their position (i.e., higher or lower) instead of their subgraph's size.

Each partition p^ℓ has three sets of shortest path data, as depicted in Figure 1: (1) the shortest paths *within* the partition p^ℓ (black edges), (2) the shortest paths to its *neighboring* partitions, NB_{p^ℓ} (blue edges), and (3) the shortest paths between the *containing* partition, $p^{\ell-1}$, to its own neighbors, $NB_{p^{\ell-1}}$ (green edges). This database of shortest paths is represented by $\mathcal{D}(p^\ell)$.

3.2 Approximate Shortest Path Model

Our approach relaxes the shortest path problem by accepting approximate shortest paths between two areas (in this case, partitions) containing s and d in place of the exact shortest path between the two points. This, in turn, reduces the number of queries required to obtain a route (hence, faster route completion times) at the cost of potentially having slightly longer paths. These are formally defined here as follows: Given an arbitrary $s, d \in V$, the exact shortest path $\rho_G(s, d)$ rarely coincides with the shortest path $\rho_{G_P}(x_s, x_d)$, where x_s and x_d are the representative vertices in the same partitions as s and d, respectively. This is because only routes between representative vertices of partitions in P can be produced from G_P, and it is highly likely that $s \neq x_s$ or $d \neq x_d$. However, it is still possible to "complete" the route by adding in the missing start and end sections. Letting $C(\rho_1, \ldots, \rho_n)$ be a route combination heuristic, the simplest "completed" route would be:

$$r^*_G(s, d) = C(\rho_G(s, x_s), \rho_{G_P}(x_s, x_d), \rho_G(x_d, d)). \tag{2}$$

This can be generalized further by replacing $\rho_{G_P}(x_s, x_d)$ with $C(\rho_{G_P}(x_1, x_2), \ldots, \rho_{G_P}(x_{n-1}, x_n))$:

$$r^*_G(s, d) = C(\rho_G(s, x_s), \rho_{G_P}(x_1, x_2), \ldots, \rho_{G_P}(x_{n-1}, x_n), \rho_G(x_d, d)), \tag{3}$$

where $x_1 = x_s$ and $x_n = x_d$. In this work, these kinds of combined paths are designated as *approximate shortest paths*, and their quality is measured based on how well they approximate their counterpart exact shortest paths. This metric is defined as the *optimal route approximation*:

$$\alpha(r^*_G(s, d)) = \frac{l_G(r^*_G(s, d))}{dist^*_G(s, d)}, \tag{4}$$

where $l_G(r^*_G(s, d))$ is the length of the approximate shortest path, and $dist_G(s, d)$ is the length of the exact shortest path in G.

3.3 Assumptions

Having presented the mathematical models relevant to route planning, we now state the core assumptions in this work as follows: Foremost is that the RPS operates over a particular service region and is primarily used by the general public for their day-to-day activities. The *service region* in this case is assumed to be a geographical area of arbitrary shape and size that has fixed bounds. This area is assumed to have comprehensive road network data available such that an RPS can be used to calculate routes within it. This road network is assumed to be represented as a graph that can be divided multiple times to produce subgraphs representing partitions as per Section 3.1. Each highest-level partition is assumed to be handled by a distinct physical or virtual device for the purpose of the RPS. Additionally, it is assumed that each partition can calculate, build, and maintain its own database of shortest paths $\mathcal{D}(p)$ independent of its other tasks. This per-partition database is then assumed to be queryable by users in a privacy-preserving manner through PIR.

We additionally assume that all entities other than the user are potential threats—henceforth, simply called "adversaries"—interested in gaining access to the user's route information. Note that no distinction is made between the service providers themselves and malicious third-parties. The kinds of information that can be leaked include the user's: (1) exact origin, (2) exact destination, and (3) the calculated "route" between them.

4 ROUTE PLANNING WITH PIR

PIR can be used by RPS to provide strong privacy guarantees by protecting the database representation of the road network graph used to calculate routes. In the simplest case, consider a database containing APSP for the whole road network graph, indexed by pairs of origin and destination vertices, (s, d). PIR can then be used to retrieve any route between any two locations on the road

Table 2. Summary of Symbols Used in Section 4

Symbol	Description
R_G^*	Set of shortest paths between all possible pairs of vertices in V
L_G^{max}	Length of the longest path in the set, R_G^*
C_{pir}	Constant representing the impact of database sizes on record retrieval times for PIR
$\mathcal{N}_p$	Average number of vertices in a partition (across all $p \in P$)
V_p^{adj}	Set of vertices in partition p that are adjacent to other partitions
$\mathcal{N}_p^{adj}$	Average number of vertices adjacent to other partitions (i.e., V_p^{adj}) (across all $p \in P$)
$\mathcal{N}_c$	Average number of external connections (across all $p \in P$)

network using a single query (i.e., $O(1)$) with very high privacy. This is the *naive* approach, which is not feasible in practice for two reasons: (1) it requires a prohibitively large amount of storage space, and (2) pre-computing APSP information for large road network graphs takes a very long time. As the following sections will discuss space and time complexity in great detail, a summary of the symbols and notation for this section is provided in Table 2.

Assuming a graph with 10,000 vertices, a longest path length of 20 vertices, and a 1-byte representation for vertex data, the total PIR database size is already around 2 GB. This is even larger for city-sized road networks such as Osaka City's, which has $\sim$ 99,000 vertices ($\sim$ 200.8 GB keeping all other parameters same). In short, letting $R_G^* = \{r_G^*(u,v) | u, v \in V\}$ and $L_G^{max} = L_G(\arg\max_{r \in R_G^*} |r|)$, the space complexity for such an approach is $O(|V|^2 \cdot L_G^{max})$. This issue is made even worse by PIR, since individual record retrieval times become slower with larger database sizes. This has been verified through preliminary experiments where data retrieval times were observed to scale linearly with database sizes by a constant factor, C_{pir}, such that accessing a single route would have a time complexity of $O(C_{pir} \cdot |V|^2) \approx O(|V|^2)$ instead of the expected $O(1)$.

Since time complexity is heavily dependent on space-complexity for PIR-based approaches, existing works [28, 35] have focused on tackling the space complexity problem, since pre-processing is assumed to be done offline only once. This is not the case for real-world road networks, however, where traffic conditions can change very quickly. One way to solve this is by distributing route data among several edge servers such that each one only manages vertices for a *distinct* partition. This reduces the time complexity to $O(|V| \cdot \mathcal{N}_p)$, where $\mathcal{N}_p = 1/|P| \cdot \sum_{p \in P} |V_p|$ is the average vertex count per partition, while the per-partition space complexity becomes $O(\mathcal{N}_p \cdot |V| \cdot L_G^{max})$. These databases still need to be kept updated, but it is much easier to do so in a distributed manner. However, dividing the data comes at the cost of weaker privacy, since the set of route data stored on the *accessed* edge server is assumed known to the adversary. This is analyzed further in Section 5.

4.1 Exact Partial Region Dijkstra's Algorithm (EPR-D)

The space complexity problem can be mitigated further by storing only edge weights between adjacent vertex pairs, as this is the minimum information needed by Dijkstra's algorithm. This is also known as the *adjacency matrix* representation, which readily maps into a database that can then be used with *any* PIR scheme. We designated this PIR-adapted approach as **E**xact **P**artial **R**egion **D**ijkstra's Algorithm (**EPR-D**), since it simply partitions and distributes graph data across several edge servers and produces exact shortest paths. A notable disadvantage is its use of separate PIR queries to retrieve information for each vertex, since the original algorithm tends to scan a lot of vertices, which can make route completion times very long. It is also possible to reidentify routes based on the sequence of edge servers queried by the user. This is examined further in Section 5. This is reflected in its average time complexity of $O(\mathcal{N}_p \cdot [\mathcal{N}_p + \mathcal{N}_p^{adj}] \cdot [|V| + |E| log |V|])$, where $\mathcal{N}_p^{adj} = 1/|P| \cdot \sum_{p \in P} |V_p^{adj}|$ is the average number of vertices adjacent to *other* partitions, in turn,

Table 3. Summary of Time and Space Complexity for Section 4

	Space Complexity	Time Complexity
Naive	$O(\|V\|^2 \cdot L_G^{max})$	$O(\|V\|^2)$
EPR-D	$O(\mathcal{N}_p \cdot [\mathcal{N}_p + \mathcal{N}_p^{adj}])$	$O(\mathcal{N}_p \cdot [\mathcal{N}_p + \mathcal{N}_p^{adj}] \cdot [\|V\| + \|E\| log \|V\|])$
APR-D	$O([\mathcal{N}_p^2 + \mathcal{N}_c] \cdot R_G^{*,max})$	$O([\mathcal{N}_p^2 + \mathcal{N}_c] \cdot [(\|P\| + \|C\| log \|P\|) + 2)])$

Table 4. Summary of Symbols Used in Section 5

Symbol	Description
$\Omega(s, d)$	Endpoint location privacy metric
$\mathcal{R}_G(u, v)$	Arbitrary routing mechanism operating on some graph G
$Q^{s,d}$	Query sequence used to obtain a route from s to d
Q^*	An arbitrary query sequence for no specific route
$\Phi(Q^*)$	Route privacy metric for some candidate query sequence Q^*
$P_{s,d}$	Partition sequence derived from some query sequence $Q^{s,d}$
$k(Q_{s,d}, Q^*)$	Indicator function for checking if $Q*$ can replace $Q_{s,d}$ and vice versa
V_X	A subset of V containing only the representative vertices

represented by $V_p^{adj} = \{v | v \notin V_p \wedge [(u, v) \in E \wedge u \in V_p]\}$. Meanwhile, its average per-partition space complexity is $O(\mathcal{N}_p \cdot [\mathcal{N}_p + \mathcal{N}_p^{adj}])$. Table 3 summarizes the complexity of EPR-D.

4.2 Approximate Partial Region Dijkstra's Algorithm (APR-D)

A possible solution to the time complexity issue is by allowing the RPS to produce *approximate shortest paths* instead of exact shortest paths. This can be done as follows: (1) calculate an approximate route between source and destination partitions p_s and p_d using Dijkstra's algorithm over partition graph G_P, (2) retrieve *subpaths* to both s and d from *within* their respective partitions via database lookup, and (3) merge all obtained paths to form the final route. We designated this approach as **A**pproximate **P**artial **R**egion **D**ijkstra's Algorithm (**APR-D**), since it produces approximate routes instead of exact ones. While it is significantly faster than EPR-D, it requires more space (since full routes are stored). It also has weaker privacy guarantees, since routes between distant areas tend to follow the same intermediate route, as will be expanded upon in Section 5. Since the database has to store complete routes, the space complexity is larger than EPR-D's being at $O([\mathcal{N}_p^2 + \mathcal{N}_c] \cdot R_G^{*,max})$, where $\mathcal{N}_c = 1/|P| \cdot \sum_{p \in P} |C_p|$ is the average number of external connections from each partition. The different stages have different time complexities, with the first stage having $O([\mathcal{N}_p^2 + \mathcal{N}_c] \cdot [|P| + |C| log |P|])$ and the second stage having $O(2 \cdot \mathcal{N}_p^2 + \mathcal{N}_c)$. The combined time complexity is then $O([\mathcal{N}_p^2 + \mathcal{N}_c] \cdot [(|P| + |C| log |P|) + 2)])$, which is significantly less than that of EPR-D, since $|P| << |V|$. The complexity of APR-D is also summarized in Table 3.

5 PROPOSED PRIVACY METRICS

To evaluate the effectiveness of a privacy-preserving PIR-based RPS, objectively quantifiable privacy metrics must first be established in the RPS context. Thus, the two models presented in this section jointly characterize the privacy of the different kinds of information that can be leaked by an RPS, as described in Section 3.3. Table 4 provides a summary of the symbols and notation used in this section.

5.1 Endpoint Location Privacy Model

As mentioned in Section 3.1, the partition database is used to store the shortest paths for each partition and is queried privately using PIR in our approach. The queried partitions are assumed

to be *knowable* by adversaries, but strong privacy is still guaranteed for exact locations *within* each partition. Let $\mathcal{R}_{G_P}(s,d)$ be a *routing mechanism* that returns the approximate shortest path $\rho_{G_P}(s,d)$, and suppose that the origin location s is replaced with a nearby location s'. If s' is still in the *same* partition as s (i.e., $s' \in V_{p_s}$), then $\mathcal{R}_{G_P}(s',d) = \mathcal{R}_{G_P}(s,d)$. The same applies replacing d with any $d' \in V_{p_d}$. An adversary knowing only $\mathcal{R}_{G_P}(s,d)$ would be unable to distinguish s,d from all other possible s',d' as long as $s' \in V_{p_s}$ and $d' \in V_{p_d}$. Thus, location privacy is guaranteed for s and d within p_s and p_d, respectively.

The privacy for any s,d pair is then proportional to the number of possible s',d' pairs that can be drawn between V_{p_s} and V_{p_d}. This is designated as the *endpoint location privacy* metric:

$$\Omega(s,d) = 1 - \frac{1}{|V_{p_s}| \cdot |V_{p_d}|}, \tag{5}$$

where V_{p_s} and V_{p_d} give the sets of all vertices in p_s and p_d, respectively. Note that while having more vertices per partition (i.e., larger $|V_{p_s}|$ and $|V_{p_d}|$) is advantageous for endpoint location privacy, this also means higher computation overhead for PIR. Extensive preliminary experiments with the PIR scheme used by our approach showed that the retrieval times scaled almost linearly with the database size at a rate of roughly $\frac{1}{45,000} \approx 2.22 \times 10^{-5}$ seconds per record. That is, a database with $|V_p|^2 \approx$ 10,000 routes was found to have an average retrieval time of $\sim$ 0.25 second, while another with $|V_p|^2 \approx$ 100,000 routes took $\sim$ 2 seconds.

5.2 Route Privacy Model

Endpoint location privacy assumes that a route's origin and destination partition are already known and thus quantifies only the privacy of the *exact* origin and destination points. This section focuses on the privacy of the *routes themselves*. As with endpoint location privacy, it is assumed that the queried partitions are knowable by adversaries. It is also assumed that they have in-depth knowledge about the algorithms used by the RPS.

Let $Q_{s,d} = \{q_0, \ldots, q_n\}$ be the *query sequence* that a user must perform to obtain a route from s to d. This sequence can be transformed into a *partition sequence* $P_{s,d} = \{p_0, \ldots, p_n\}$ (where $p_0 = p_s$ and $p_n = p_d$) using a function $f_{qp} : Q \to P$ that maps every element of Q to its handling partition in P. If the partitioning is hierarchical, then only the highest-level partitions are considered, since they already contain the shortest path data of lower-level partitions, as stated in Section 3.1.

Note that partition sequences are not simply partitions along the final route. For instance, in the case of Dijkstra's algorithm, they can be thought of as the *entire* sequence of "scanned" vertices, as depicted in Figure 2. It is therefore possible for several routes to share the same partition sequence (though unlikely in the case of Dijkstra's). This is modeled as an indicator function that identifies whether or not a candidate Q^* can replace $Q_{s,d}$ for calculating $r_{G_P}(s,d)$ and vice versa:

$$k(Q_{s,d}, Q^*) = \begin{cases} 1 & \text{if } f_{qp}(Q_{s,d}) = f_{qp}(Q^*) \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

With this, the lower bound for the total number of *distinct routes* that share Q^* is:

$$\sum_{s',d' \in V_X} k(Q_{s',d'}, Q^*), \tag{7}$$

where $V_X \subseteq V$ contains *only* the representative vertices (e.g., x_u, x_v) associated with the partition graph G_P, as described in Section 3.1. This ensures that routes where $p_{s'} \neq p_s$ and $p_{d'} \neq p_d$ are counted with equal importance as the route where $p_{s'} = p_s \wedge p_{d'} = p_d$—hence, the focus on *distinct*

Fig. 2. Queried partitions using Dijkstra's algorithm to calculate two routes with the same origin but different destinations (~ 1 km away from each other). The numbers indicate how many queries were handled by each partition.

routes. Finally, the *route privacy* can then be quantified for any Q^* as follows:

$$\Phi(Q^*) = 1 - \frac{1}{\sum_{s',d' \in V_X} k(Q_{s',d'}, Q^*)}. \tag{8}$$

6 HIERARCHICAL PRIVACY-PRESERVING ROUTE PLANNING

Our proposed approach, HPRoP, is built up from several key design choices to meet particular requirements. That is, HPRoP should be able to:

- REQ1: Compose feasible approximate shortest paths by carefully choosing its component routes from the appropriate partitions,
- REQ2: Privately retrieve component route information from said partitions with minimal processing overhead,
- REQ3: Produce an approximate shortest path with good *optimal route approximation* values (i.e., $\alpha(r^*_G(s,d)) \rightarrow 1.0$),
- REQ4: Ensure a good level of privacy protection for the user's exact origin and destination points and intermediate route,
- REQ5: Reflect dynamic and up-to-date road conditions, and
- REQ6: Scale reasonably well with changes in client demand and computational resource availability over time and per area.

All these are brought together by a novel hierarchical route-planning heuristic presented in the latter half of this section, along with other improvements to privacy and routing.

6.1 Private Information Retrieval (PIR)

HPRoP uses PIR as its core route privacy-preservation mechanism. The choice of implementation was the SealPIR [2] library configured to use **Brakerski/Fan-Vercauteren (BFV) Homomorphic Encryption (HE)** [13] with a database upper bound of $N = 216$, a plaintext modulus of $log(t) = 12$, and a dimensionality factor, $d = 2$. This implementation was chosen specifically for its significantly reduced processing and communication overhead compared to other HE-based ones, making it ideal for an RPS. This along with the hierarchical route-planning heuristic drastically reduces the number of queries and, in effect, the route completion time.

Fig. 3. Road network graph of Osaka City, Japan, hierarchically partitioned using the Inertial Flow algorithm.

6.2 Inertial Flow Partitioning

Optimal route approximation and endpoint location privacy are highly dependent on how the service region is partitioned. Straightforward methods such as grid partitioning are simple but often result in disjoint partition subgraphs in the presence of natural barriers such as rivers and so on. To mitigate this, one way would be to ensure that each partition subgraph is a strongly connected component, ensuring high internal connectivity and reachability. Examples of road network graph aware partitioning methods include PUNCH [12], *Buffoon* [32], and Inertial Flow [33].

Inertial Flow was chosen for HPRoP, as it is a relatively simple algorithm based on maximum flow that results in balanced partitions and also preserves internal connectivity. A vertex threshold of around 300 nodes per partition was chosen instead of an area-based threshold to guarantee an *Endpoint Location Privacy* of $\Omega(s, d) \approx 99.998\%$ (i.e., $< 0.002\%$ probability). Moreover, based on preliminary experiments with SealPIR, a partition database with $300^2 \approx 90,000$ routes is expected to have retrieval times between $1.5 - 2.5$ seconds (1.75 seconds, on average), which is viable when combined with HPRoP's reduced query counts. A queue is initialized by adding the entire road network graph to it. A graph from this queue is then used as input to the Inertial Flow algorithm to produce two balanced partition subgraphs. The simple iterative technique in Reference [1] was used alongside Inertial Flow to find and apply optimal cuts during this step. If any of the subgraphs do not yet satisfy vertex threshold, they are simply added back to the queue. This entire procedure is then repeated until the queue is empty. Afterwards, every two consecutive cuts was then retroactively denoted as a separate *partition level,* as shown in Figure 3, and the partitions under each are then tagged accordingly. Due to the vertex threshold, the highest-level partition may vary greatly from area to area. Some routes therefore require more queries to complete over other routes, which increases profiling risk. A partition level threshold $\ell_{threshold} = 3$ was therefore imposed such that higher level ones were reassigned to $\ell = \ell_{threshold}$.

6.3 Distributed Architecture

HPRoP leverages the hierarchically partitioned road network by delegating each partition to a different entity. In the cloud-based scenario, these entities would be server instances; while, in the edge-based scenario, these would be edge-servers throughout the smart city. The edge-based architecture presents several advantages. First, it allows PIR queries to be directed only to partitions that have the information necessary to answer them, effectively distributing the computational load of using PIR. Second, it allows the system to better scale based on the number of users, availability of computing resources, and so on, which can vary greatly at different times across different parts of the city, as shown in Figure 4. Finally, it also makes the pre-computation of shortest paths to neighboring partitions more efficient, since it can be done independently by every partition after an initial exchange of road condition information with said neighbors. This is useful for reflecting dynamic road conditions bound to some local area.

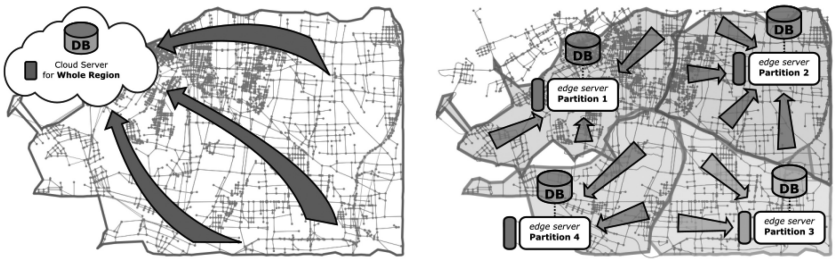


Fig. 4. Cloud-based architecture (left) vs. distributed/edge-based architecture (right).

ALGORITHM 1: Hierarchical Route Planning Heuristic

Input: Source node s, Destination node d, Current level l_c
Output: The final route r_*

1 **begin**
2 $l_o \leftarrow FindBaseLevel(s, d)$;
3 $r_* \leftarrow RetrievePath(p_s^{l_o}, p_d^{l_o}$, "from source");
4 Initialize $l_c \leftarrow l_o + 1$;
5 **while** $l_c \leq l_{threshold}$ **do**
6 Initialize $r_s^{l_c}, r_d^{l_c} \leftarrow []$;
7 **if** $l_c < (l_{threshold} - 1)$ **then**
8 $r_s^{l_c} \leftarrow GetSubroute(s, l_c, r_*$, "from source");
9 $r_d^{l_c} \leftarrow GetSubroute(d, l_c, r_*$, "from destination");
10 **else**
11 $r_s^{l_c} \leftarrow GetEndSubroute(s, l_c, r_*$, "from source");
12 $r_d^{l_c} \leftarrow GetEndSubroute(d, l_c, r_*$, "from destination");
13 $r_* \leftarrow MergeRoutes(r_*, r_s^{l_c}, r_d^{l_c})$;
14 $l_c \leftarrow l_c + 1$;
15 **return** r_*;

6.4 Heuristic Algorithm

HPRoP uses a simple heuristic algorithm to arrive at an approximate shortest path as follows:

(1) *Initialization*: Find an initial basis route between the lowest-level partitions containing source and destination, then move up one level.
(2) *Subroute Connection*: Connect the source and destination partitions at the current level to the basis route using subroutes.
(3) *Basis Route Merging*: Merge the subroutes into the basis route.
(4) Repeat steps (2) to (3) until the highest partition level is reached

Algorithm 1 runs exclusively on the client-side, sending PIR queries to edge servers handling specific partitions. Route information is obtained solely through these PIR queries, and, thus, no information is leaked by the queries themselves. However, the number of queries, their timestamps, and the partitions they were sent to are still assumed to be known to the adversary.

The algorithm starts with an *Initialization* step (lines 2–4 in Algorithm 1), which finds an approximate shortest path between the lowest-level partitions containing s and d separately as shown in Figure 5. This level is denoted as the base level $l_o = \arg\min_l (p_s^l \neq p_d^l)$. For simplicity, this is just denoted as $FindBaseLevel(s, d)$ in Algorithm 1. The client then sends a PIR query to partition $p_s^{l_o}$,

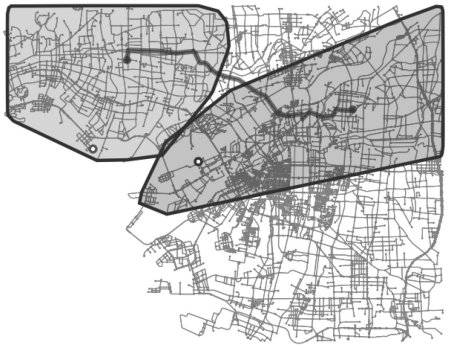


Fig. 5. Initialization: Find a shortest path between the lowest-level partitions containing s (blue circle) and d (green) separately.

Fig. 6. Initialization: Use discovered path as basis route (purple line) for the next level.

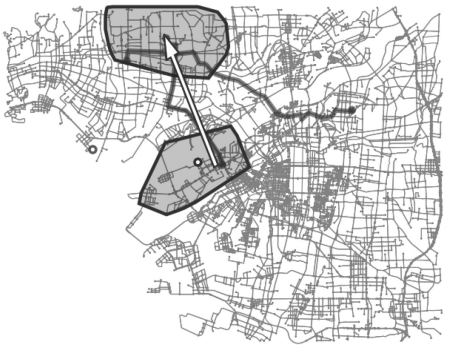


Fig. 7. Source Subroute Connection: Find and connect source subroute (blue line) to basis route (purple line).

Fig. 8. Dest. Subroute Connection: Find and connect dest. subroute (green line) to basis route (purple line).

retrieving a route to partition $p_d^{l_o}$. This corresponds to $RetrievePath(p_u, p_v, \mathfrak{D})$, which retrieves the shortest path between two partitions taking into account some direction flag $\mathfrak{D}$. This flag simply indicates whether the path is being calculated from the source or the destination, which will be relevant later. The retrieved route is then denoted as the initial *basis route* r_* shown in Figure 6. At the end of this step, the current level variable l_c is also initialized (line 4).

The main loop starts from the *Subroute Connection* steps (lines 6–9 in Algorithm 1). The algorithm for Subroute Connection itself is described in Algorithm 2. A *subroute* $r_x^{l_c}$ is defined as a path that connects a *basis partition* $p_x^{l_c}$ to the current basis route r_*. The basis partition given by $p_x^{l_c}$ is always the source or destination partition at level l_c containing some vertex x and is used to obtain the source or destination sub-partitions (depending on the direction $\mathfrak{D}$) at line 3 of Algorithm 2. For instance, the *source subroute* is obtained by finding a sequence of shortest paths from $p_s^{l_c}$ that connects to the basis route, as shown in Figure 7. This step also uses several important functions, such as: (1) $FindRoutePartitions(r_*, l)$, which gives the sequence of partitions at level l along r_*, and (2) $DoesNotIntersect(r_x^{l_c}, r_*)$, which is "True" if $r_x^{l_c}$ and r_* have no common vertices. Starting from $p_x^{l_c}$, it builds $r_x^{l_c}$ by retrieving a path to nearby connected partitions (line 9) and then connecting them to the subroute (lines 10–13). Since $r_x^{l_c}$ might not yet intersect r_* during the initial iteration, this is repeated with an updated reference partition (lines 14–15) until an intersection

Fig. 9. Basis Route Merging: Merge routes and use as new basis route (purple line).

is found. Computing the *destination subroute* follows the same steps but first has to *reverse* the aforementioned sequence (lines 5–7) so the algorithm can begin from the last route partition. This step is shown in Figure 8.

The *Basis Route Merging* step (line 13 in Algorithm 1) is performed once the source and destination subroutes are found. The basic idea is to find the "best" point at which the subroutes intersect with the basis route and join them there. For the source subroute, the "best" point is as far as possible from the start of the basis route; while for the destination subroute, this is as far as possible from the end of the basis route. This is illustrated in Figure 9. This merged route is then used as the new basis route. The current level l_c is then updated (line 15), and the loop is restarted. The loop is terminated once l_c reaches $l_{threshold}$.

The time complexity of this algorithm depends on the required number of PIR database lookups. Finding the initial basis route and calculating the final routes at p_s and p_d always require a single lookup each. Meanwhile, the number of lookups at each level depends on the maximum length of

ALGORITHM 2: Subroute Connection

Input: Basis node x, Current level l_c, Basis route r_*, Direction $\mathfrak{D}$
Output: Subroute at the current level $r_x^{l_c}$

1 **begin**
2 Initialize $r_x^{l_c} \leftarrow []$;
3 $p_c \leftarrow p_x^{l_c}$ //This retrieves either source or destination sub-partition depending on direction $\mathfrak{D}$
$RP^{l_c} \leftarrow FindRoutePartitions(r_*, l_c)$
4 **if** $\mathfrak{D}$ ***is*** *"from destination"* **then**
5 $Reverse(RP^{l_c})$;
6 $i \leftarrow 0$;
7 **while** $DoesNotIntersect(r_x^{l_c}, r_*)$ ***and*** $i < |RP^{l_c}|$ **do**
8 $r_{part} \leftarrow RetrievePath(p_c, RP^{l_c}[i], \mathfrak{D})$;
9 **if** $\mathfrak{D}$ ***is*** *"from source"* **then**
10 $r_x^{l_c} \leftarrow r_x^{l_c} + r_{part}$;
11 **else if** $\mathfrak{D}$ ***is*** *"from destination"* **then**
12 $r_x^{l_c} \leftarrow r_{part} + r_x^{l_c}$;
13 $p_c \leftarrow RP^{l_c}[i]$;
14 $i \leftarrow i + 1$;
15 **return** $r_x^{l_c}$;

the shortest path in G_P *at that level*, which is given by $max_{p_u^\ell, p_v^\ell \in P^\ell} |r^*_{G_P^\ell}(p_u^\ell, p_v^\ell)|$, where $G_P^\ell \subset G_P$ containing only that level's partitions P^ℓ and connections C^ℓ. The average time complexity is then:

$$O\left(\left[\mathcal{N}_p^2 + \sum_{\ell \in L} \mathcal{N}_c^\ell\right] \cdot \left[3 + \sum_{\ell \in L} 2 \cdot \max_{p_u^\ell, p_v^\ell \in P^\ell} |r^*_{G_P^\ell}(p_u^\ell, p_v^\ell)|\right]\right), \tag{9}$$

where $\mathcal{N}_c^\ell$ is the average number of connections from each partition *at level* ℓ. Since HPRoP precomputes and stores shortest path data for multiple levels per partition, it is necessary to account for $\mathcal{N}_c^\ell$ in HPRoP's space complexity:

$$O\left(\left[\mathcal{N}_p^2 + \sum_{\ell \in L} \mathcal{N}_c^\ell\right] \cdot R_G^{*,max}\right). \tag{10}$$

6.4.1 Route Privacy Mechanism. Route privacy, as defined in Section 5.2, quantifies the privacy based on how many other possible routes have a query sequence matching that of a given route, where more matches mean better privacy. That is, an adequate route privacy mechanism should: (1) maximize the matching of query sequences between all possible routes and (2) minimize the information gain from the order of queries in the sequence itself. HPRoP already achieves the latter via hierarchical execution, which can somewhat obfuscate the actual query sequence but does not necessarily strengthen the former. To address this, the algorithm is extended to *pad* the query sequence with *dummy queries*. The basic idea is to query multiple other partitions instead of just p_s and p_d to ensure that the actual origin and destination partitions are "hidden" among them. In theory, querying more partitions would mean better route privacy at the cost of longer route completion times. Achieving full route privacy, however, would require querying all level partitions at every iteration of the algorithm at least once, which would require prohibitively long route completion times. For example, a service region with a total of 463 partitions would require roughly 810 seconds (13.5 minutes), on average, to complete a single route. However, simply querying a small random subset of the aforementioned partitions will not be enough to ensure a certain level of route privacy, since an adversary can simply use the hierarchy of partitions to check for inconsistencies in the set of queried partitions and easily identify the dummy ones. Instead, we chose to limit HPRoP to querying all other partitions *under the same parent* as the highest-level partitions containing s and d. This selection method is straightforward and ensures that none of the queried partitions can easily be identified as dummy partitions. Additionally, this ensures that route privacy will be around $\Phi(Q^*) \approx 1 - 1/jk$, where j and k are the total number of partitions under the same parent partitions as p_s and p_d, respectively.

This is implemented through the *Subroute End Connection* steps (lines 10–12 in Algorithm 1), while the procedure itself is presented in Algorithm 3. Instead of stopping when the subroute and basis route first intersect, this algorithm continues until all other partitions sharing the same parent $p_x^{l_c - 1}$ as the basis partition have been queried. This is done by repeatedly drawing a partition p_{sub} from a queue of these same-parent sub-partitions SP^{l_c} (line 10). If p_{sub} is the same as the current partition p_c, then a part of the subroute is retrieved as normal (lines 11–18). If it is a route partition, then it is instead pushed back to the subpartition queue (lines 19–22). If both prior conditions are not satisfied, then a dummy query is simply sent to p_{sub} (line 24). This ensures that important queries are mixed in with dummy queries, making it more difficult to determine which ones are relevant.

6.5 Shortcut Connections

A simple strategy for improving algorithm performance is through pre-computing and storing shortest path data to partitions *beyond* just the adjacent ones. This reduces the number of queries

ALGORITHM 3: Subroute End Connection with Dummies

Input: Source or Destination node x, Current level l_c, Basis route r_*
Output: Source or Destination subroute at the current level $r_x^{l_c}$

begin
  Initialize $r_x^{l_c} \leftarrow []$;
  $p_c \leftarrow p_x^{l_c}$;
  $RP^{l_c} \leftarrow FindRoutePartitions(r_*, l_c)$
  if $\mathfrak{D}$ ***is*** *"from destination"* **then**
    $Reverse(RP^{l_c})$;
  $SP^{l_c} \leftarrow FindSubPartitions(p_x^{l_c-1})$ **as** $Queue$
  $j \leftarrow 0$;
  while $|SP_x^{l_c-1}| > 0$ **do**
    $p_{sub} \leftarrow Pop(SP^{l_c})$;
    if $p_{sub} = p_c$ **and** $DoesNotIntersect(r_d^{l_c}, r_*)$ **then**
      $r_{part} \leftarrow RetrievePath(p_{sub}, RP^{l_c}[j])$;
      if $\mathfrak{D}$ ***is*** *"from source"* **then**
        $r_x^{l_c} \leftarrow r_x^{l_c} + r_{part}$;
      else if $\mathfrak{D}$ ***is*** *"from destination"* **then**
        $r_x^{l_c} \leftarrow r_{part} + r_x^{l_c}$;
      $p_c \leftarrow RP^{l_c}[j]$;
      $j \leftarrow j + 1$;
    else if $p_{sub} \in RP^{l_c}$ **and** $DoesNotIntersect(r_d^{l_c}, r_*)$ **then**
      if $Index(p_{sub}, RP^{l_c}) > j$ **then**
        $Push(p_{sub}, SP^{l_c})$
        $Shuffle(SP^{l_c})$
    else
      $SendDummyQuery(p_{sub})$;
  return $r_x^{l_c}$;

to complete a route while also improving optimal route approximation for paths to further away partitions. These paths to non-adjacent partitions were therefore denoted as *Shortcut Connections*, represented as additional edges in the partition graph.

These shortcut connections, however, also increase the pre-computation time for each partition relative to how many of them need to be made. In addition, the extent of the road network graph needed for pre-computation also increases based on the distance to the partitions being connected. It is therefore more useful to limit the number of partitions to connect to and how far those partitions can be. HPRoP considers two methods for determining shortcut connections: (1) the Same Parent Shortcuts method and (2) the *N-hop Neighbor Shortcuts* method. *Same Parent Shortcuts* simply connects each partition to all other partitions under the same parent partition, as shown in Figure 10 (Middle). *N-hop Neighbor Shortcuts* pre-computes shortcuts to other N-hop away partitions, as shown in Figure 10 (Right). Both would theoretically improve the *optimal route approximation* when calculating subroutes between non-adjacent partitions and greatly reduce the possibility of broken routes. HPRoP currently has no mechanisms to handle these other than deferring to the user's device to perform local route calculation to bridge the final gap. The usefulness of these methods is shown through the results in Section 7.3.1.

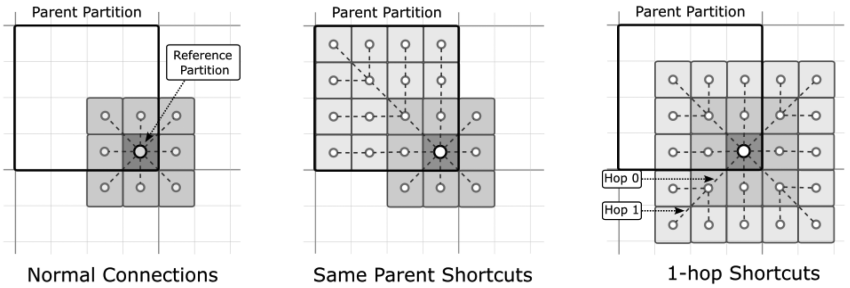


Fig. 10. Demonstration of different shortcut connection strategies for improving the heuristic algorithm's performance against the base case (Left), where the dark red shape represents the starting partition, and the lighter red shapes represent the partitions it connects to. The black outline represents the starting partition's parent. (Middle) uses Same Parent Shortcuts, while (Right) uses 1-hop Neighbor Shortcuts.

7 EVALUATION

In this section, the details of the evaluation framework for HPRoP are first presented prior to showing the actual evaluation results and their subsequent analysis.

7.1 Environment

HPRoP was implemented in Python on a Jupyter notebook for ease of testing and visualization, with the notebook itself encapsulated in a Docker container for portability. The execution environment was a dedicated Linux server running Ubuntu 20.04.1 SMP equipped with a AMD Ryzen Threadripper 3970X 32-Core processor and 256 GB RAM in total.

The service region was a rectangular geographical area of roughly 546 km^2 (i.e., 26 km in width, 21 km in height) encompassing the entire road network of Osaka City, Japan, and a portion of the immediately outlying areas. Its road network graph consists of $|V|$ = 99,734 vertices and $|E|$ = 269,614 edges. The region is hierarchically partitioned using the Inertial Flow algorithm, as shown in Figure 3 based on the parameters in Section 6.2.

7.2 Methodology

Evaluation was done through several metrics under the following categories: (1) Utility, (2) Privacy, and (3) Performance. The Utility category pertains to the usefulness of the service, with the *Optimal Route Approximation* metric falling under this category. Since the base algorithm in Section 6.4 cannot guarantee complete routes, *Route Errors* are also included here as a metric. Route errors are then defined as the occurrence count of broken routes during testing. In turn, a *broken route* is defined as a route where a subroute connection *cannot be established* to the highest-level partition containing either s or d, and thereby results in a route that cannot be completed by HPRoP's algorithm. The Privacy category is composed of the *Endpoint Location Privacy* and *Route Privacy* metrics, described in Section 5. The Performance category pertains to how well the service can deal with higher client demand, dynamic road conditions, and so on, without service quality degradation. The *Memory Usage*, *Route Completion Time*, and *Pre-processing Time* metrics fall under this category.

Evaluation was done by comparing HPRoP to the two baseline PIR-based approaches—EPR-D and APR-D—previously presented in Section 4. For the *Utility* category, *Privacy* category, and *Route Completion Time* metrics, all three approaches were evaluated by calculating routes for randomly generated s, d pairs until 4,000 successful routes have been completed. This termination threshold was chosen to be sufficiently high enough to capture any route errors that might occur for HPRoP.

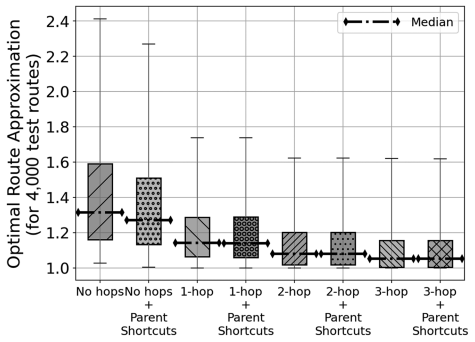


Fig. 11. Distribution of *Optimal Route Approximation* results for different Shortcut Connection methods.

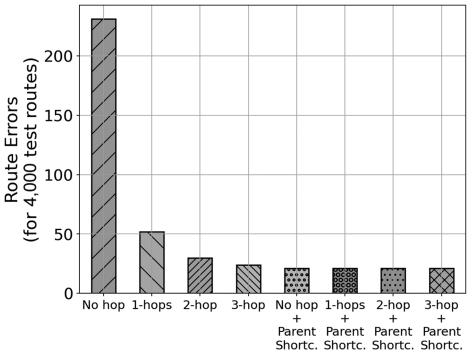


Fig. 12. Comparison of total Route Errors for different Shortcut Connection methods.

Note that both APR-D and EPR-D are guaranteed to produce complete routes, so this metric is no longer evaluated for them. *Memory Usage* was evaluated by calculating the projected size of the per-partition databases for the highest-level partitions. Finally, *Pre-processing Time* was evaluated by measuring the time to build each partition's database.

7.3 Results

7.3.1 Effect of Shortcut Connections. The base performance of HPRoP under different shortcut connection methods in Section 6.5 is first characterized with the goal of finding the method that maximizes optimal route approximation while minimizing both route completion time and route errors. Figure 11 shows the resulting distribution of optimal route approximation values for 4,000 successful test routes under different methods. Surprisingly, *Same Parent Shortcuts* have an almost negligible effect on optimal route approximation, with *N-hop Neighbor Shortcuts* being a more effective way to increase the said metric. However, *Same Parent Shortcuts* were highly effective in preventing the occurrence of broken routes, reducing the error count to 21, as shown in Figure 12. Further investigation of these remaining errors showed that they were caused by choosing the same vertex as s and d, which results in failure, as no routing can be done by the algorithm. In short, for all 4,000 test routes, *Same Parent Shortcuts* seem to eliminate all occurrences of true broken routes—i.e., where the highest-level partitions containing s or d could not be reached. This also validates the hypothesis that broken routes are caused by a lack of reachability at the final partition level that can contain more than 4 child partitions due to the deliberate choice to set $\ell_{threshold} = 3$ mentioned in Section 6.2.

Meanwhile, using *1-hop Neighbor Shortcuts* drastically improves the results of the 75th percentile from $\alpha(r_*) \approx 1.54$ to $\alpha(r_*) \approx 1.28$, but anything beyond *2-hop Neighbor Shortcuts* is seen to have gradually diminishing returns. Finally, the overhead caused by the additional shortcut connections is evaluated based on the pre-processing time metric in Section 7.2. Figure 13 shows that both methods increase the per-partition pre-processing time to about ~ 1.5 seconds once *1-Hop Neighbor Shortcuts* are introduced but stabilizes around this value even as the number of hops is further increased. In contrast, the effect of *Same Parent Shortcuts* on pre-processing time is minimal, amounting to an increase of ~ 0.1 second, on average. Thus, both methods are equally viable in terms of this metric.

Table 5 summarizes the results discussed so far. Based on these, it was decided to use *2-hop Neighbor Shortcuts* with *Same Parent Shortcuts* as the representative configuration for HPRoP, as it offers good *optimal route approximation* and short pre-computation times with minimal errors.

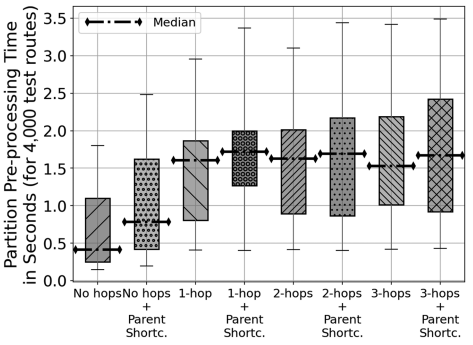


Fig. 13. Distribution of Pre-computation Times for different Shortcut Connection methods.

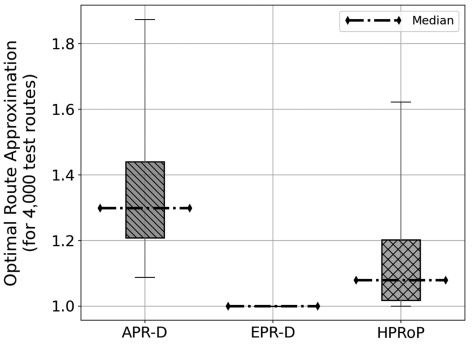


Fig. 14. Distribution of Optimal Route Approximation results using APR-D, EPR-D, and HPRoP.

Table 5. Summary of Results for Different Shortcut Connection Configurations

	Normal				**With Same Parent Shortcuts**			
	0-hop	1-hop	2-hop	3-hop	0-hop	1-hop	**2-hop**	3-hop
Optimal Route Approximation								
Min	1.0	1.0	1.0	1.0	1.0	1.0	**1.0**	1.0
25%	1.160	1.062	1.018	1.003	1.133	1.060	**1.017**	1.003
50%	1.314	1.141	1.080	1.052	1.270	1.138	**1.079**	1.052
75%	1.590	1.287	1.202	1.156	1.510	1.288	**1.202**	1.156
Max	16.256	5.210	5.371	5.371	15.424	5.210	**5.371**	5.371
Per-partition Routes Calculation Time (in seconds)								
Min	0.04	0.11	0.19	0.19	0.08	0.11	**0.19**	0.19
25%	0.16	0.58	1.88	3.33	0.24	0.75	**1.92**	3.48
50%	0.20	1.43	2.49	4.41	0.47	1.51	**2.90**	4.75
75%	0.28	1.60	3.36	5.70	1.30	1.74	**3.47**	6.03
Max	1.27	3.17	5.78	10.55	3.15	3.37	**6.75**	11.44
Route Errors (per 4,000 successful routes)								
Total	231.0	52.0	30.0	24.0	21.0	21.0	**21.0**	21.0

7.3.2 Optimal Route Approximation. The performance of HPRoP, APR-D, and EPR-D in terms of *optimal route approximation* is shown in Figure 14. As expected, EPR-D achieves a constant *optimal route approximation* value of $\alpha(r_*) = 1.0$, since it always produces exact shortest paths. APR-D produced routes with $\alpha(r_*) \approx 1.44$ for the 75th percentile despite operating only over the partition graph. HPRoP produced even better routes with $\alpha(r_*) \approx 1.20$ for the 75th percentile, whereas APR-D was only able to achieve this for the 25th percentile of all results. Additionally, HPRoP achieves much better worst-case routes than APR-D.

7.3.3 Endpoint Location Privacy. Endpoint Location Privacy $\Omega(s, d)$ describes how well the exact s, d is kept private, as described in Section 5.1. Since Inertial Flow partitioning was used, evaluation results showed that all three approaches were able to achieve an average endpoint location privacy of $\Omega(s, d) = 0.999982$. Thus, each route is *indistinguishable* from approximately 99% of all other routes between the same partitions, making all three approaches equally viable.

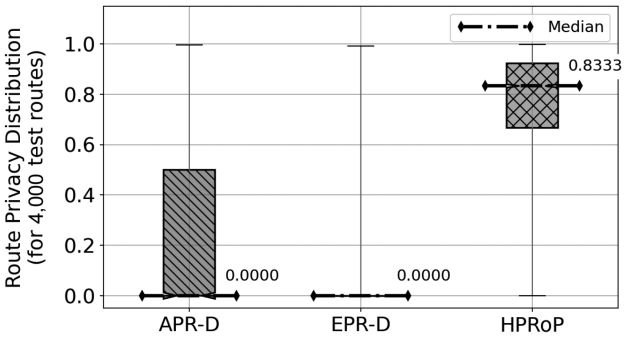


Fig. 15. Distribution of Route Privacy $\Phi(Q^*)$ results for APR-D, EPR-D, and HPRoP.

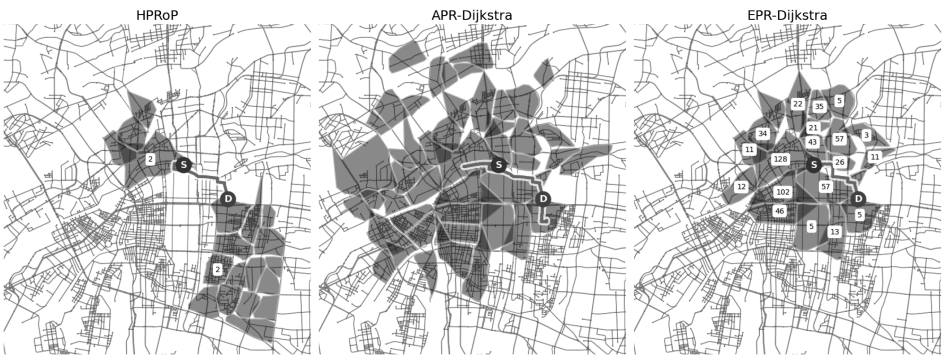


Fig. 16. Visualization of queried partitions for APR-Dijkstra (center), EPR-Dijkstra (right), and HPRoP (left) for the same route. Numbers indicate partitions that were queried multiple times.

7.3.4 Route Privacy. Route Privacy $\Phi(Q^*)$ describes how well a route is kept private based on how many other routes share a query sequence similar to its own, as described in Section 5.2. To evaluate this, a lookup table for the routes between each $s, d \in V_x$ such that $s \neq d$ was done separately for EPR-D, APR-D, and HPRoP to account for their unique querying behaviors, as shown in Figure 16.

Figure 15 shows a comparison of the route privacy distributions across all three approaches. EPR-D showed the worst route privacy, achieving $\Phi(Q^*) > 0$ only for the 25th percentile of all results. APR-D performed only slightly better with $\Phi(Q^*) > 0$ for about 50% of all results, while achieving $\Phi(Q^*) \geq 0.5$ for only 25% of them. This is expected, since both have no inherent privacy mechanisms, yet this does illustrate that APR-D is still better than EPR-D in terms of route privacy. HPRoP, in contrast, has route privacy of $\Phi(Q^*) \geq 0.80$ for the 50th percentile of all results, while also achieving $\Phi(Q^*) \geq 0.50$ for the 75th percentile, surpassing both EPR-D and APR-D. However, further analysis of the routes showed that the worst-case route privacy (i.e., $\Phi(Q^*) = 0.0$) still happens for $\sim$ 12.2% of all test routes. This suggests that HPRoP does not fully guarantee route privacy for all cases, although this can be increased further by adding more dummy queries.

Additionally, the relationship between optimal route approximation $\alpha(r_*)$ and route privacy metric $\Phi(Q^*)$ was analyzed for HPRoP. This was done to determine whether some tradeoff exists between the two metrics. The results in Figure 17 show that the majority of the routes have $\alpha(r_{s,d}) < 1.2$ across widely varying levels of route privacy. This suggests that the two metrics have little effect on one another, and performing a simple Pearson correlation confirms that there is only a weak positive correlation ($\rho \approx 0.104$) between the two.

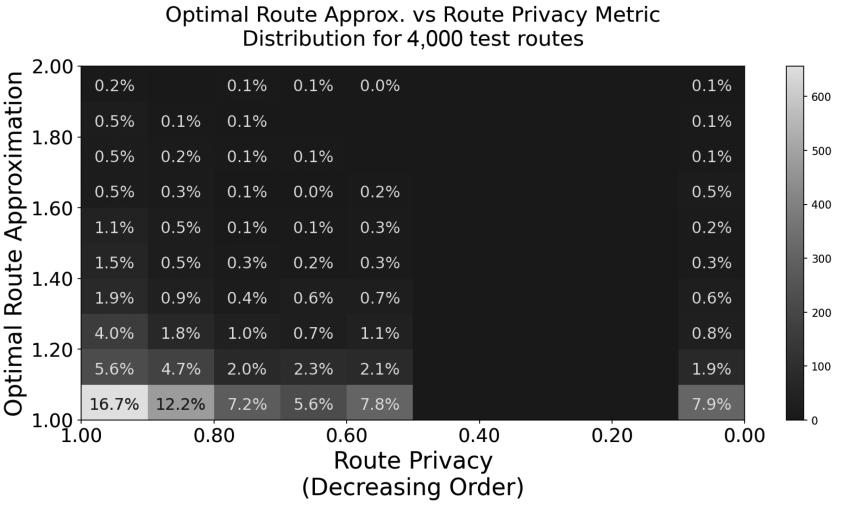


Fig. 17. Distribution of Optimal Route Approximation and Route Privacy results using HPRoP for 4,000 test routes. Note that both axes were reoriented to show the best results on the lower left.

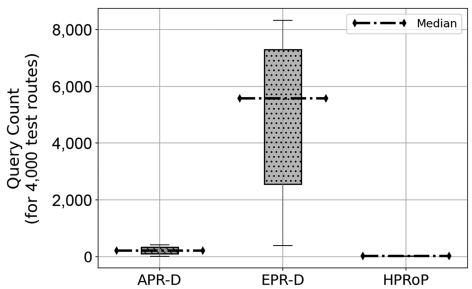


Fig. 18. Distribution of the required number of queries for route completion.

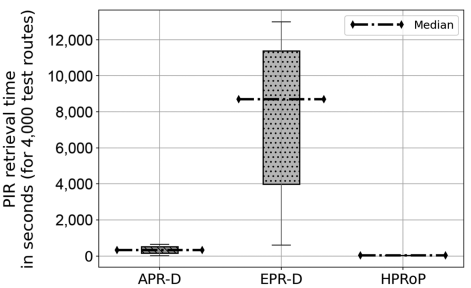


Fig. 19. Distribution of total PIR retrieval times in seconds.

7.3.5 Route Completion Time. Obtaining a route using any of the three algorithms (APR-D, EPR-D, and HPRoP) requires the client to make multiple PIR queries to the RPS. Thus, route completion time is highly dependent on how many such queries need to be made as shown in Figure 18, and is equivalent to the total processing overhead for an RPS. In this work, it is calculated based on projections derived from the preliminary experiments on SealPIR database retrieval times mentioned at the end of Section 5.1. The empirically derived value of roughly $\frac{1}{45,000} \approx 2.22 \times 10^{-5}$ seconds per record is then used to calculate the route completion times, which are shown in Figure 19. Note that we opted to use projections here instead of simulating the actual results, as the latter would take a prohibitively long time to conclude in the case of EPR-D (and, to a lesser extent, APR-D). For instance, APR-D requires $\sim$ 214 queries, on average, which is expected to take 5.55 minutes (333.26 seconds) per route. EPR-D is even worse, requiring $\sim$ 4,958 queries, on average, which would take at least 2.15 hours (7,731.47 seconds) just to complete a single route. Both are clearly impractical from the perspective of any modern RPS. This is in contrast to HPRoP, which requires significantly less queries, on average (at $\sim$ 25 queries), and takes only 23.55 seconds per route. This is because HPRoP's algorithm bypasses the need to explore large sections of the partition and road network graph, as it already starts with a very coarse route between the origin and destination partitions

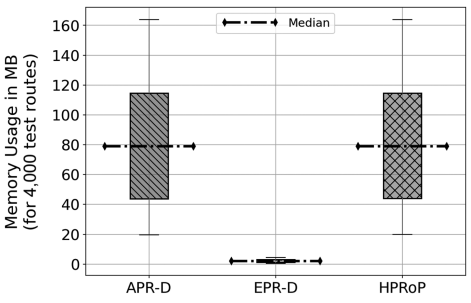


Fig. 20. Distribution of per-partition memory usage (in MB) for the different approaches.

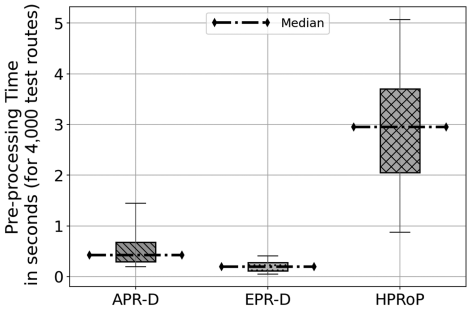


Fig. 21. Distribution of per-partition pre-processing time results (in seconds) for the different approaches.

at the lowest level to guide its route search. APR-D and EPR-D, in contrast, explore outwards from the origin, checking every unexplored partition or vertex on the way, as per Dijkstra's algorithm.

7.3.6 Memory Usage. Memory usage depends on the size of the route databases maintained by each partition, which is very important in a distributed environment with resource-constrained devices. Figure 20 shows the distribution of per-partition memory usage across the three approaches. APR-D and HPRoP calculate and store a similar number of routes, thus requiring an allocation of around 20–160 MB per partition. EPR-D has a *significantly* smaller memory footprint at around 0.5–4 MB per partition, because it essentially stores a next-hop matrix instead. In practice, however, the memory usage of all three approaches is well within the capabilities of standard edge servers.

7.3.7 Pre-processing Time. The pre-processing time metric is total time needed to build each partition's database during the pre-processing phase, which determines how often it can be updated during operation. As shown in Figure 21, EPR-D requires less than ≤ 0.5 second per partition, on average, since it only needs to calculate the routes *inside* each partition, while APR-D requires ≤ 1.5 seconds, since it also needs to calculate routes to immediately neighboring partitions in addition. Meanwhile, HPRoP needs to calculate routes inside each partition, routes to neighboring partitions, and routes to other partitions under the same parent. This results in slightly longer pre-processing times at 1–5 seconds per partition. Regardless, the pre-processing time for all three approaches is clearly fast enough to accommodate frequent updates even under dynamic road conditions.

8 CONCLUSION

In this work, the **Hierarchical Privacy-Preserving Route Planning (HPRoP)** approach was proposed, which combines Inertial Flow partitioning, **Private Information Retrieval (PIR)**, and Edge Computing techniques along with a novel hierarchical route-planning heuristic algorithm to produce routes that can adequately approximate the actual shortest paths while also providing *endpoint location privacy* and *route privacy*. HPRoP reliably produced routes with an *optimal route approximation* of $\alpha(r_*) \leq 1.2$, while also achieving near-optimal endpoint location privacy at $\Omega(s, d) \approx 1.0$ and good route privacy at $\Phi(Q^*) \geq 0.5$. In terms of performance, HPRoP has a route completion time of around 23.55 seconds, on average, which is reasonable for a privacy-preserving RPS. Its viability for deployment in a distributed/edge-based smart city context were also shown through its relatively small memory footprint (20–160 MB for each partition's database) and short pre-processing times (2–5 seconds per partition), which are well within the capabilities of conventional edge servers.

In addition, although most modern route-planning algorithms can also use PIR, we did not use them as the basis of our approach due to several reasons. For instance, *bounded-hop techniques* such as two-hop labelling [9] are the fastest known class of routing algorithms but they require computing and storing prohibitively large indices for city-sized road networks, which becomes even worse in combination with PIR. *Separator-based techniques* such as **Customizable Route Planning (CRP)** [11] and *goal-directed techniques* such as ALT (based on A*) [18] and Arc Flags [20] have very long precomputation times, making them infeasible to use with dynamic road networks having edge weights that need to be updated frequently. The *hierarchical technique* called **Contraction Hierarchies (CH)** [16] has none of the aforementioned problems but still presents a potential route privacy risk, since it must explicitly query the partitions along the actual shortest path multiple times to obtain the final path. Nevertheless, it is a good routing algorithm to consider for future PIR-based RPS work.

Aside from the experiments presented in Section 7, we also conducted tests on how well HPRoP generalizes to other road networks, but the results are omitted for conciseness. In particular, it was tested on three other large cities—New York, Shanghai, and Tokyo—by obtaining 4,000 test routes and examining the optimal route approximation distribution. For 75% of all routes, the optimal route approximation was at $\alpha(r_*) \leq 1.20$ for New York, $\alpha(r_*) \leq 1.23$ for Shanghai, and $\alpha(r_*) \leq 1.24$ for Tokyo. This indicates that HPRoP generalizes somewhat, but further research is still required.

In the future, we also plan to refine HPRoP's route-planning algorithm to further improve optimal route approximation and reduce route completion times by exploring more efficient data structures for route information and improve route privacy by considering out-of-order query execution.

REFERENCES

[1] Gaurav Aggarwal, Sreenivas Gollapudi, and Ali Kemal Sinop. 2021. Sketch-based algorithms for approximate shortest paths in road networks. In *Proceedings of the Web Conference*. 3918–3929.

[2] Sebastian Angel, Hao Chen, Kim Laine, and Srinath Setty. 2018. PIR with compressed queries and amortized query processing. In *Proceedings of the IEEE Symposium on Security and Privacy (SP'18)*. IEEE, 962–979.

[3] Ugur Ilker Atmaca, Carsten Maple, Gregory Epiphaniou, and Mehrdad Dianati. 2021. A privacy-preserving route planning scheme for the internet of vehicles. *Ad Hoc Netw.* 123 (2021), 102680.

[4] Barnana Baruah and Subhasish Dhal. 2022. A security and privacy preserved intelligent vehicle navigation system. *IEEE Trans. Depend. Sec. Comput.* 20, 2 (2022).

[5] Amos Beimel, Yuval Ishai, and Tal Malkin. 2000. Reducing the servers computation in private information retrieval: PIR with preprocessing. In *Proceedings of the 20th Annual International Cryptology Conference*. Springer, 55–73.

[6] Melissa Chase and Seny Kamara. 2010. Structured encryption and controlled disclosure. In *Proceedings of the 16th International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 577–594.

[7] Benny Chor and Niv Gilboa. 1997. Computationally private information retrieval. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*. 304–313.

[8] Yu-Li Chou, H. Edwin Romeijn, and Robert L. Smith. 1998. Approximating shortest paths in large-scale networks with an application to intelligent transportation systems. *INFORMS J. Comput.* 10, 2 (1998), 163–179.

[9] Edith Cohen, Eran Halperin, Haim Kaplan, and Uri Zwick. 2003. Reachability and distance queries via 2-hop labels. *SIAM J. Comput.* 32, 5 (2003), 1338–1355.

[10] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. *Introduction to Algorithms*. MIT Press.

[11] Daniel Delling, Andrew V. Goldberg, Thomas Pajor, and Renato F. Werneck. 2017. Customizable route planning in road networks. *Transport. Sci.* 51, 2 (2017), 566–591.

[12] Daniel Delling, Andrew V. Goldberg, Ilya Razenshteyn, and Renato F. Werneck. 2011. Graph partitioning with natural cuts. In *Proceedings of the IEEE International Parallel & Distributed Processing Symposium*. IEEE, 1135–1146.

[13] Junfeng Fan and Frederik Vercauteren. 2012. Somewhat practical fully homomorphic encryption. *Cryptology ePrint Archive* (2012). https://eprint.iacr.org/2012/144

[14] Farhad Farokhi, Iman Shames, and Karl H. Johansson. 2020. Private routing and ride-sharing using homomorphic encryption. *IET Cyber-phys. Syst.: Theor. Applic.* 5, 4 (2020), 311–320.

[15] Sébastien Gambs, Marc-Olivier Killijian, and Miguel Núñez del Prado Cortez. 2010. Show me how you move and I will tell you who you are. In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS*. 34–41.

[16] Robert Geisberger, Peter Sanders, Dominik Schultes, and Christian Vetter. 2012. Exact routing in large road networks using contraction hierarchies. *Transport. Sci.* 46, 3 (2012), 388–404.

[17] Esha Ghosh, Seny Kamara, and Roberto Tamassia. 2021. Efficient graph encryption scheme for shortest path queries. In *Proceedings of the ACM Asia Conference on Computer and Communications Security*. 516–525.

[18] Andrew V. Goldberg and Chris Harrelson. 2005. Computing the shortest path: A search meets graph theory. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA'05)*, Vol. 5. Citeseer, 156–165.

[19] Alexandra Henzinger, Matthew M. Hong, Henry Corrigan-Gibbs, Sarah Meiklejohn, and Vinod Vaikuntanathan. 2022. One server for the price of two: Simple and fast single-server private information retrieval. *Cryptology ePrint Archive* (2022). https://eprint.iacr.org/2022/949

[20] Moritz Hilger, Ekkehard Köhler, Rolf H. Möhring, and Heiko Schilling. 2009. Fast point-to-point shortest path computations with arc-flags. *Short. Path Prob.: Ninth DIMACS Implement. Chall.* 74 (2009), 41–72.

[21] Takahiro Ikeda, Min-Yao Hsu, Hiroshi Imai, Shigeki Nishimura, Hiroshi Shimoura, Takeo Hashimoto, Kenji Tenmoku, and Kunihiko Mitoh. 1994. A fast algorithm for finding better routes by AI search techniques. In *Proceedings of the Vehicle Navigation and Information Systems Conference (VNIS'94)*. IEEE, 291–296.

[22] George Rosario Jagadeesh, Thambipillai Srikanthan, and K. H. Quek. 2002. Heuristic techniques for accelerating hierarchical routing on road networks. *IEEE Trans. Intell. Transport. Syst.* 3, 4 (2002), 301–309.

[23] Sungwon Jung and Sakti Pramanik. 2002. An efficient path computation model for hierarchically structured topographical road maps. *IEEE Trans. Knowl. Data Eng.* 14, 5 (2002), 1029–1046.

[24] Meng Li, Yifei Chen, Shuli Zheng, Donghui Hu, Chhagan Lal, and Mauro Conti. 2020. Privacy-preserving navigation supporting similar queries in vehicular networks. *IEEE Trans. Depend. Sec. Comput.* 19, 2 (2020), 1133–1148.

[25] Yangfan Liang, Yining Liu, and Brij B. Gupta. 2022. PPRP: Preserving-privacy route planning scheme in VANETs. *ACM Trans. Internet Technol.* 22, 4 (2022), 1–18.

[26] Chang Liu, Liehuang Zhu, Xiangjian He, and Jinjun Chen. 2018. Enabling privacy-preserving shortest distance queries on encrypted graph data. *IEEE Trans. Depend. Sec. Comput.* 18, 1 (2018), 192–204.

[27] Samir Jordan Menon and David J. Wu. 2022. Spiral: Fast, high-rate single-server PIR via FHE composition. In *Proceedings of the IEEE Symposium on Security and Privacy (SP'22)*. IEEE, 930–947.

[28] Kyriakos Mouratidis. 2013. Strong location privacy: A case study on shortest path queries. In *Proceedings of the IEEE 29th International Conference on Data Engineering Workshops (ICDEW'13)*. IEEE, 136–143.

[29] Muhammad Haris Mughees, Hao Chen, and Ling Ren. 2021. OnionPIR: Response efficient single-server PIR. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*. 2292–2306.

[30] Vincent Primault, Antoine Boutet, Sonia Ben Mokhtar, and Lionel Brunie. 2018. The long road to computational location privacy: A survey. *IEEE Commun. Surv. Tutor.* 21, 3 (2018), 2772–2793.

[31] Michael O. Rabin. 2005. How to exchange secrets with oblivious transfer. *Cryptology ePrint Archive* (2005). https://eprint.iacr.org/2005/187

[32] Peter Sanders and Christian Schulz. 2012. Distributed evolutionary graph partitioning. In *Proceedings of the 14th Workshop on Algorithm Engineering and Experiments (ALENEX'12)*. SIAM, 16–29.

[33] Aaron Schild and Christian Sommer. 2015. On balanced separators in road networks. In *Proceedings of the International Symposium on Experimental Algorithms*. Springer, 286–297.

[34] Christian Sommer. 2016. All-pairs approximate shortest paths and distance oracle preprocessing. In *Proceedings of the 43rd International Colloquium on Automata, Languages, and Programming (ICALP'16)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[35] David J. Wu, Joe Zimmerman, Jérémy Planul, and John C. Mitchell. 2016. Privacy-preserving shortest path computation. *arXiv preprint arXiv:1601.02281* (2016).

[36] Andrew Chi-Chih Yao. 1986. How to generate and exchange secrets. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science (SFCS'86)*. IEEE, 162–167.

[37] Can Zhang, Liehuang Zhu, Chang Xu, Kashif Sharif, Chuan Zhang, and Ximeng Liu. 2020. PGAS: Privacy-preserving graph encryption for accurate constrained shortest distance queries. *Inf. Sci.* 506 (2020), 325–345.

[38] Lei Zhang, Jing Li, Songtao Yang, and Bin Wang. 2017. Privacy preserving in cloud environment for obstructed shortest path query. *Wirel. Person. Commun.* 96, 2 (2017), 2305–2322.

[39] Jun Zhou, Shiying Chen, Kim-Kwang Raymond Choo, Zhenfu Cao, and Xiaolei Dong. 2021. EPNS: Efficient privacy preserving intelligent traffic navigation from multiparty delegated computation in cloud-assisted VANETs. *IEEE Trans. Mob. Comput.* 22, 3 (2021).

Received 9 January 2023; revised 25 June 2023; accepted 2 August 2023

Periodic Event-Triggered CACC and Communication Co-design for Vehicle Platooning

ANQI FU, SIJIA CHEN, and JUNFEI QIAO, Faculty of Information Technology, Beijing University of Technology; Key Laboratory of Computational Intelligence and Intelligent Systems, Beijing University of Technology; Beijing Laboratory of Smart Environmental Protection; Beijing Artificial Intelligence Institute, China
CHENGPU YU, School of Automation, Beijing Institute of Technology, China and Beijing Institute of Technology Chongqing Innovation Center, China

Cooperative Adaptive Cruise Control (CACC) based vehicle platooning can increase the safety and efficiency of traffic. This work looks into the communication and control problems of vehicle platooning, and proposes a control and communication co-design for CACC. First, an integrated radar system is presented. This system integrates sensing of relative position, speed, and communication between a predecessor and its follower. Second, a working scheme for the integrated radar system is presented. This scheme allows the radar systems to switch periodically between different working modes without interferences from other modes. Therefore, the relative position, speed, and communication can be asynchronously periodically updated to the controller. Third, a periodic event-triggered control approach is presented. This approach allows asynchronous periodic sampling of the output, and is deeply co-designed with the radar system and its working scheme. Delays are also considered in the control approach. The co-design CACC approach can guarantee the vehicle platoons to be string stable. A numerical example has shown the feasibility of the approach.

CCS Concepts: • **Computer systems organization → Embedded and cyber-physical systems**; • **Networks** → *Network protocols;*

Additional Key Words and Phrases: Periodic event-triggered control, control and communication co-design, vehicle-to-vehicle communication, cooperative adaptive cruise control, vehicle platooning

ACM Reference format:
Anqi Fu, Sijia Chen, Junfei Qiao, and Chengpu Yu. 2023. Periodic Event-Triggered CACC and Communication Co-design for Vehicle Platooning. *ACM Trans. Cyber-Phys. Syst.* 7, 4, Article 28 (October 2023), 19 pages.
https://doi.org/10.1145/3617125

This work was supported by the National Natural Science Foundation of China (NSFC) under Grants 62003009, 62021003, 61890930-5, National Key Research and Development Program of China (2021ZD0112301, 2021ZD0112302).
Authors' addresses: A. Fu (Corresponding author), S. Chen, and J. Qiao, Faculty of Information Technology, Beijing University of Technology; Key Laboratory of Computational Intelligence and Intelligent Systems, Beijing University of Technology; Beijing Laboratory of Smart Environmental Protection; Beijing Artificial Intelligence Institute, Beijing, China, 100124; e-mails: a.fu@bjut.edu.cn, scarlett.chensijia@foxmail.com, adqiao@bjut.edu.cn; C. Yu, School of Automation, Beijing Institute of Technology, Beijing, China, 100081 and Beijing Institute of Technology Chongqing Innovation Center, Chongqing, China, 401120; e-mail: yuchengpu@bit.edu.cn.

2378-962X/2023/10-ART28 $15.00
https://doi.org/10.1145/3617125

1 INTRODUCTION

Safety and efficiency are important in modern transportation. Normally, vehicles require humans to observe the current environment, estimate the routes, make decisions, and control the vehicle via steering wheels and pedals. We want to reduce stress while driving, and therefore, it is necessary to develop and apply automatic driving technologies, meanwhile considering traffic safety and efficiency.

Cooperative Adaptive Cruise Control (CACC) is a type of automatic driving technology that aims at cruising with desired small distances between vehicles, while ensuring safety. Meanwhile, disturbances should be attenuated throughout the string of vehicles. This property is also referred to as *string stability*, see e.g., [10] and the references therein. When several vehicles cruise together with desired velocity and distance, a vehicle platoon is formed. A vehicle platooning is when a group of vehicles drive in a string formation on the road with small distances in between the vehicles [17]. It is meant to increase the road capacity. Meanwhile, it can also save energy consumed by minimizing air drag [15]. The front vehicles in platoons are normally called as leaders, while the rest of the vehicles are named as followers.

Vehicle platoons can bring several advantages during cruising [15, 16, 19]. First, the drivers in the followers do not need to control the vehicles, thus the driving stress can be reduced. Second, the distances between vehicles can be minimized, therefore the utilization of the road resources can be maximized. Meanwhile, with smaller distance, vehicles in the platoons can make full use of the wind generated by their predecessors, and thus the resistance from the wind can be decreased. As a result, drive power can be saved. Third, only the leaders need to execute higher levels of automatic driving algorithms or are to be driven by experienced drivers, while the followers only need to keep up with the leaders. Therefore, system resources consumed by, e.g., sensing and computing, can be saved. Lower levels of automatic driving can also reduce the cost during manufacturing and can deliver safety and efficiency benefits to more and existed vehicles. Therefore, vehicle platooning is a promising way to increase the safety and efficiency of modern transportation.

Vehicle-to-vehicle (V2V) communications and CACC algorithms are two of the major issues in vehicle platooning. CACC requires a solid and fast wireless communication between vehicles for cooperating. There are several candidates. **Global Systems for Mobile** (**GSM**, e.g., 5G, see [22, 29, 30]) is fast and robust. However, the signals are transmitted via Cell Sites, and Cell Sites are difficult to install in the wild and inside tunnels. WiFi (see e.g., [8]) and Zigbee (see e.g., [11, 33]) allow the routers to be installed inside the platoons. However, both technologies work on public radio channels and share wireless channels with other devices. If there are many platoons within a small area, transmission delays will increase dramatically due to limited radio resources, and thus the transmissions are not solid any more. **Dedicated Short Range Communications** (**DSRC**, see e.g., [28, 32, 35]) is proposed based on IEEE 802.11p. It presents a 10MHz radio channel for traffic application, such as Electronic Fee Collection system. This communication technology may be good at broadcasting traffic information, such as jams or accidents. However, it also has the shared bandwidth problem when being applied to vehicle platooning. Therefore, it is necessary to seek for new communication technologies. In this work, we consider V2V communications via radar systems, the feasibility can be found in [2, 21] and the references therein. Applying radar systems for V2V transmissions has the following advantages:

- Compared with GSM, radar based communication is face-to-face, and the communication does not require Cell Sites, making the platoons still stable when there are no Sites around.
- Compared with Zigbee, WiFi, or DSRC, radar based communication does not require any public transmission channels, making the transmissions solid and with small delays, even when there are plenty of vehicles nearby.

With the digitalization of sensors and controllers, the control loops can only be closed at discrete times. To determine the closed time sequences, normally periodic **time-triggered control (TTC)** is applied. TTC closes the control loops based on the timers, while regardless of the system states. Therefore, it wastes system resources, e.g., networked bandwidth, computing resources, and system energy. **Event-triggered control (ETC)**, on the other hand, only closes the control loops when necessary, e.g., when the systems are going to be unstable. Thus, ETC can save system resources. Examples of ETC can be found in, e.g., [1, 3, 20, 25, 27, 34, 37]. **Periodic event-triggered control (PETC)** combines TTC and ETC. The sensors work periodically, instead of continuously as many ETC approaches require. Only when some predesigned event-triggered conditions are satisfied at each discrete time, the control loops will be closed. Therefore, the system resources, such as the sensors' working time and the communication resources, can be further saved. Examples of PETC can be found in e.g., [4–6, 14, 26, 31, 36]. [4, 14, 26] present PETC for linear systems, while [5, 6, 31, 36] study PETC for nonlinear systems. Note that, for the extreme case when the system triggers events during every sampling period, then the PETC strategy would behave like periodic TTC strategy with the same sampling period. Therefore, if a desired stability cannot be guaranteed by the PETC, the same stability can neither be guaranteed by TTC with the same sampling period. However, PETC is still meaningful since it has two major advantages. First, PETC can minimize the communications with pre-designed stability and performance compared with TTC. Second, PETC provides a trade-off between communications and stability, bringing much flexibility during controller design.

In this paper, to solve the two major issues in vehicle platooning, we propose a control and communication co-design for vehicle platoons. First, we present an integrated radar system. In this system, the radars are integrated with both sensing and transmitting working modes. More specifically, the front radars of vehicles can measure the relative distances and speeds with respect to their predecessors. Besides, the same radars can also receive and decode messages transmitted from their predecessor's back radars. Since these transmissions are wireless and face-to-face, the transmission channels are solid and with small delays. Second, a working scheme for the integrated radar system is proposed. This scheme drives the radar to switch between all three modes, i.e., measure the relative distance, measure the relative speed, and transmitting, periodically and asynchronously, thus different working modes will not influence each other. Third, a new type of PETC is presented. This PETC approach allows asynchronous periodic sampling, and considers delays by, e.g., computing and transmitting, at the same time. This type of asynchronous sampling is necessary due to the fact that, at one time, the integrated radar system can only work at one mode. And the switches between different working modes require time. Therefore, compared with other existed PETC approaches which consider synchronous samplings, our proposed PETC method is more suitable for the proposed CACC with integrated radar systems. This asynchronous periodic working manner together the joint design of the parameters make this PETC control approach deeply co-designed with the radar system and its working scheme.

One of the key problems solved in this paper is the asynchronous sampling problem in PETC. The asynchronous sampling means that the elements of the system output vector are sampled asynchronously, due to the asynchronous periodic working manner. Asynchronous sampling problem has been considered in [23], [13], and [12]. [23] proposes aperiodic sampling intervals for TTC only, and thus wastes system resources. Both [13] and [12] consider ETC with asynchronous sampling. However, due to the application of delay-based method, both work use the state feedback control and thus limit their applications to output feedback control systems. Besides, the transmissions from controller to actuators are excluded in the event-triggered mechanism. Our new PETC approach applies hybrid system method, and thus can be applied to output feedback control systems, and the event-triggered mechanism can be applied to both the transmissions from the sensors to

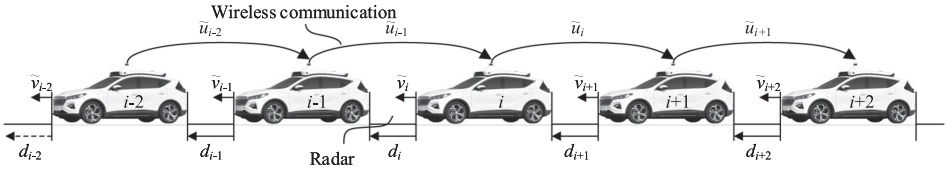


Fig. 1. Vehicle platoon.

the controllers and the controllers to the actuators. Numerical results show that our approach can save system resources while guarantees the systems to be $\mathcal{L}_2$ stable.

This paper is organized as follows. In Section 2, some notations and preliminaries are presented. The construction of a model for the vehicle platoons and problem formulation are presented in Section 3. Followed which are the integrated radar system and its working scheme in Section 4. The PETC approach is presented in Section 5. Section 6 gives a numerical validation of the approach. And the work is concluded in Section 7.

2 DEFINITION AND PRELIMINARIES

We denote the n-dimensional Euclidean space by $\mathbb{R}^n$, the positive real numbers by $\mathbb{R}^+$, by $\mathbb{R}_0^+ = \mathbb{R}^+ \cup \{0\}$. The natural numbers including zero is denoted by $\mathbb{N}$. A symmetric matrix $M \in \mathbb{R}^{n\times n}$ is said to be positive (negative) definite, denoted by $M > 0$ ($M < 0$), whenever $x^{\mathrm{T}}Mx > 0$ ($x^{\mathrm{T}}Mx < 0$) for all $x \neq 0$, $x \in \mathbb{R}^n$. $M \geq 0$ ($M \leq 0$) means M is a positive (negative) semi-definite matrix. For a symmetric matrix $M \in \mathbb{R}^{n\times n}$, $\lambda_{\min}(M)$ and $\lambda_{\max}(M)$ denote the minimum and maximum eigenvalue of M, respectively. For a matrix $N \in \mathbb{R}^{m\times n}$, we denote $N^{\mathrm{T}} \in \mathbb{R}^{n\times m}$ as its transposed matrix. For a locally integrable signal w: $\mathbb{R}^+ \to \mathbb{R}^n$, we denote by $\|w\|_{\mathcal{L}_2} = \sqrt{\int_0^\infty |w(t)|^2 dt}$ its $\mathcal{L}_2$-norm. Furthermore, we define the space of all locally integrable signals with a finite $\mathcal{L}_2$-norm as $\mathcal{L}_2$. A function $\alpha : \mathbb{R}^+ \to \mathbb{R}^+$ belongs to class $\mathcal{K}$ ($\alpha \in \mathcal{K}$) if α is a continuous function, $\alpha(0) = 0$, and $s_1 > s_2 \Rightarrow \alpha(s_1) > \alpha(s_2)$. A function $\alpha : \mathbb{R}^+ \to \mathbb{R}^+$ belongs to class $\mathcal{K}_\infty$ ($\alpha \in \mathcal{K}_\infty$) if $\alpha \in \mathcal{K}$ and $\lim_{s\to\infty} \alpha(s) = \infty$. For brevity, we sometimes replace $\begin{bmatrix} A & B \\ B^{\mathrm{T}} & C \end{bmatrix}$ as $\begin{bmatrix} A & B \\ * & C \end{bmatrix}$.

3 MODEL DESCRIPTION, CONTROL OBJECTIVES, AND PROBLEM FORMULATION

Consider a homogeneous platoon with N vehicles as illustrated in Figure 1. The first vehicle, which is also called as the leader, is denoted as $\mathcal{V}_0$, while the set $\tilde{\mathcal{N}} := \{1, \ldots, N-1\}$ contains all the followers' labels. The main objective of each vehicle $\mathcal{V}_i$, $i \in \tilde{\mathcal{N}}$, is to follow its predecessor, i.e., vehicle $\mathcal{V}_{i-1}$, while guaranteeing a predesigned distance $\tilde{d}_i$. A general form of $\tilde{d}_i$ can be given by

$$\tilde{d}_i(t) := r_i + \tilde{h} v_i(t), \tag{1}$$

where $r_i :\geq 0$ is a scalar, $\tilde{h} \in \mathbb{R}^+$ is a given time interval, named as *time gap*, $v_i(t) \in \mathbb{R}$ denotes the velocity of $\mathcal{V}_i$ at t. Define $q_i(t)$ the position of $\mathcal{V}_i$ at time t. The actual distance between $\mathcal{V}_i$ and its predecessor $\mathcal{V}_{i-1}$, denoted as $d_i(t)$, is given by

$$d_i(t) := q_{i-1}(t) - q_i(t) - L_i, \tag{2}$$

in which L_i is the length of $\mathcal{V}_i$. Therefore, the spacing error $e_i(t)$ can be computed by

$$e_i(t) := d_i(t) - \tilde{d}_i(t) = (q_{i-1}(t) - q_i(t) - L_i) - (r_i + \tilde{h} v_i(t)). \tag{3}$$

Define $\tilde{v}_i(t)$ and $a_i(t)$ the velocity and acceleration of $\mathcal{V}_i$, $i \in \{0, \ldots, N-1\}$ at t respectively. The velocity error $\dot{e}_i(t)$ can be derived as

$$\dot{e}_i(t) = \tilde{v}_{i-1}(t) - \tilde{v}_i(t) - \tilde{h} a_i(t). \tag{4}$$

Define $\tilde{u}_0(t)$ an exogenous input to the vehicle platoon at time t. This input $\tilde{u}_0(t)$ is given to the leader $\mathcal{V}_0$. Physically, $\tilde{u}_0(t)$ is the desired acceleration of the vehicle platoon. The dynamics of each vehicle is given as $\forall i \in \{0, \ldots, N-1\}$:

$$\begin{cases} \dot{v}_i(t) = a_i(t) \\ \dot{a}_i(t) = -\frac{1}{\tilde{\tau}_d} a_i(t) + \frac{1}{\tilde{\tau}_d} \tilde{u}_i(t), \end{cases} \tag{5}$$

where $\tilde{\tau}_d$ denotes the characteristic time constant of the vehicle drive-line. Since we consider homogeneous vehicle platoons, all vehicles in the same platoon have the same $\tilde{\tau}_d$. $\tilde{u}_i(t)$ is the input of $\mathcal{V}_i$, $i \in \tilde{\mathcal{N}}$, and it is the desired acceleration of $\mathcal{V}_i$. The dynamics of $\tilde{u}_i(t)$, $i \in \tilde{\mathcal{N}}$ are given by

$$\dot{\tilde{u}}_i(t) = -\frac{1}{\tilde{h}} \tilde{u}_i(t) + \frac{1}{\tilde{h}} \chi_i(t). \tag{6}$$

where $\chi_i(t)$ denotes the control input of $\mathcal{V}_i$. $\chi_i(t)$ is computed by the following control law:

$$\chi_i(t) = k_p e_i(t) + k_d \dot{e}_i(t) + \tilde{u}_{i-1}(t), \tag{7}$$

in which, k_p and k_d are scalar parameters called as controller gains.

The controller given in (7) has two parts: a PD feedback control part ($k_p e_i(t)$ and $k_d \dot{e}_i(t)$) and a feedforward control part ($\tilde{u}_{i-1}(t)$). Therefore, the control input is computed considering the current spacing error and velocity error between $\mathcal{V}_i$ and its predecessor $\mathcal{V}_{i-1}$, and also the input of $\mathcal{V}_{i-1}$. By the form of (7), it is easy to see that, the controller is a type of *one-vehicle look-ahead control strategy* such that the control of $\mathcal{V}_i$, $i \in \tilde{\mathcal{N}}$ only depends on the local available information, i.e., $e_i(t)$, $\dot{e}_i(t)$, and the information from its predecessor $\mathcal{V}_{i-1}$, i.e., $\tilde{u}_{i-1}(t)$.

The control objective of this vehicle platoon is to guarantee a so-called *string stability*, given as below.

Definition 3.1 (String Stability [9, 10]). For any exogenous input $\tilde{u}_0 \in \mathcal{L}_2$ and any $x(0) \in \mathbb{R}^{n_x}$ with $x = (x_0, x_1, \ldots, x_{N-1})$ the lumped state vector, it should hold that for all $i \in \{0, \ldots, N-1\}$, there exist $\mathcal{K}_\infty$-functions $\alpha_i, \beta_i : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ such that the corresponding solution to (5), and (6) with the corresponding controller (7), satisfies

$$\|\chi_i\|_{\mathcal{L}_2} \leq \alpha_i(\|\tilde{u}_0\|_{\mathcal{L}_2}) + \beta_i(\|x(0)\|). \tag{8}$$

In Definition 3.1, the state vector of each vehicle can be chosen based on requirements. An example is given as $x_i(t) = [e_i(t) \;\; \tilde{v}_i(t) \;\; a_i(t) \;\; \tilde{u}_i(t)]^{\mathrm{T}}$ as shown in [9].

Remark that the string stability objective is closely related to the notion of $\mathcal{L}_2$-stability (see e.g., [18]). From (7), to control each of the vehicle $\mathcal{V}_i$ in the vehicle platoon, local information $e_i(t)$, $\dot{e}_i(t)$ are required, meanwhile the information $\tilde{u}_{i-1}(t)$ from its predecessor $\mathcal{V}_i$ is also required. To obtain these values, digitalized sensing and communicating between two vehicles are necessary. This digitalization will also make the execution time of the controller discretized. Therefore, it is necessary to look into the problem that, under this discretized manner, how to obtain $e_i(t)$, $\dot{e}_i(t)$, and $\tilde{u}_{i-1}(t)$, while some string stability can still be guaranteed. At the same time, system resources can be saved.

Remark 3.2. The exogenous input $\tilde{u}_0$ can contain braking signals. In the proposed vehicle platoon system, there is a trade-off when designing the distance $\tilde{d}_i$ and the maximum acceleration during brakes to avoid collisions. A shorter distance requires the acceleration to be smaller; a bigger distance allows the acceleration to be larger.

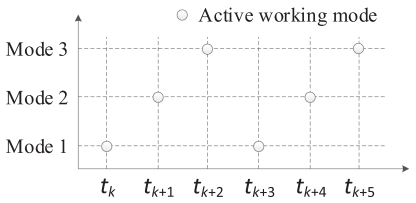


Fig. 2. Integrated radar system working scheme.

4 SENSING AND COMMUNICATING INTEGRATED RADAR SYSTEM AND ITS SCHEDULING

In this section, we propose a new type of integrated radar system that can measure $e_i(t)$, $\dot{e}_i(t)$, and transmit $\tilde{u}_{i-1}(t)$ wirelessly between two neighbouring vehicles in the same platoon.

To support obtaining all three parameters, the integrated radar system has three working modes:

- Mode 1, measure $e_i(t)$ mode. In this mode, only the front radars of the vehicles $\mathcal{V}_i$, $i \in \tilde{\mathcal{N}}$ work. They work similar to a range-only radar: the radars will transmit radio waves, when the radio waves reach its predecessor, they will be reflected back. By multiplying the velocity and the transmitting time intervals of the radio waves, the distances between $\mathcal{V}_i$ and $\mathcal{V}_{i-1}$, i.e., $e_i(t)$ can be measured.
- Mode 2, measure $\dot{e}_i(t)$ mode. In this mode, only the front radars of $\mathcal{V}_i$ work. They work similar to a Doppler radar: the radars will transmit radio waves, by analyzing the frequency changes of the radio waves reflected by the predecessors, $\dot{e}_i(t)$ can be computed.
- Mode 3, transmit $\tilde{u}_{i-1}(t)$ mode. In this mode, both the front radars of the followers $\mathcal{V}_i$ and back radars of the processors $\mathcal{V}_{i-1}$ work. The back radars work as radio transmitters: they encode and send $\tilde{u}_{i-1}(t)$. While the front radars work as receivers: they receive and decode $\tilde{u}_{i-1}(t)$.

The integrated front radar, the integrated back radar, the digital unites, together with the corresponding computing algorithms form the sensing and communicating integrated radar system. In this system, all three modes can work together, or separately. To reduce the complexity of the integrated radar systems, and increase the robustness of the communication channels, we let the modes work separately. More specifically, we activate the different working modes of the radar systems one after another. Thus, a working scheme is presented that drives the radar systems to switch between different modes. The working scheme is shown in Figure 2.

The radar systems work at periodic discrete time, called as working time. At each working time, the radars work only on one of the three modes, and switches following the order of *Mode 1, Mode 2, Mode 3*. We call going through all three modes one time as a *round*. After one *round*, $e_i(t)$, $\dot{e}_i(t)$, and $\tilde{u}_{i-1}(t)$ can be obtained once. With this working scheme, each mode will work periodically and asynchronously from each other.

Meanwhile, at each working time, the controllers compute the control signals $\chi_i(t)$. Whether the newly obtained parameters are used in the computation of the control signals or not, dependents on an event-triggered mechanism, which will be discussed in the next section. When computing the control input, for those with events, the newly obtained samplings are applied; for those without events, history information will be applied instead.

By now, we can rewrite the model of the vehicle platoon. For vehicle $\mathcal{V}_i$, since we are interested in string stability, it is of interest to evaluate the input-output behavior in terms of $\mathcal{L}_2$-gain with respect to input χ_{i-1} and output χ_i. To describe the input-output behavior of $\mathcal{V}_i$, construct the

system state as

$$\tilde{\xi}_i(t) = \begin{bmatrix} e_i(t) & \dot{e}_i(t) & a_{i-1}(t) & \tilde{u}_{i-1}(t) & a_i(t) & \tilde{u}_i(t) \end{bmatrix}^{\mathrm{T}}. \tag{9}$$

select system output as

$$\tilde{y}_i(t) = \begin{bmatrix} e_i(t) & \dot{e}_i(t) & \tilde{u}_{i-1}(t) \end{bmatrix}^{\mathrm{T}}. \tag{10}$$

Thus, the dynamics is given as

$$\begin{aligned} \dot{\tilde{\xi}}_i(t) &= \tilde{A}\tilde{\xi}_i(t) + \tilde{B}\chi_i(t) + \tilde{E}\chi_{i-1}(t) \\ \tilde{y}_i &= \tilde{C}\tilde{\xi}_i(t), \end{aligned} \tag{11}$$

where

$$\tilde{A} := \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & -\frac{1}{\tilde{\tau}_d} & \frac{1}{\tilde{\tau}_d} & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{\tilde{h}} & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{\tilde{\tau}_d} & \frac{1}{\tilde{\tau}_d} \\ 0 & 0 & 0 & 0 & 0 & -\frac{1}{\tilde{h}} \end{bmatrix}$$

$$\tilde{B} := \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \frac{1}{\tilde{h}} \end{bmatrix}^{\mathrm{T}}$$

$$\tilde{E} := \begin{bmatrix} 0 & 0 & 0 & \frac{1}{\tilde{h}} & 0 & 0 \end{bmatrix}^{\mathrm{T}}$$

$$\tilde{C} := \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}.$$

The controller is given as

$$\chi_i(t) = K\hat{\tilde{y}}_i(t), \tag{12}$$

where $\hat{\tilde{y}}_i$ is updated from $\tilde{y}_i(t)$ via an event-triggered sample-and-hold mechanism shown in the next section, and

$$K := \begin{bmatrix} k_p & k_d & 1 \end{bmatrix}.$$

A performance output $\tilde{z}_i(t)$ is given by

$$\tilde{z}_i(t) = K\tilde{y}(t) = K\tilde{C}\tilde{\xi}_i(t). \tag{13}$$

It is easy to see that, $\tilde{z}_i(t)$ is a version of $\chi_i(t)$ without the sample-and-hold mechanism applied to system output $\tilde{y}(t)$.

Definition 4.1. The system (11), (12), and (13) is said to be $\mathcal{L}_2$-stable from input χ_{i-1} to output $\tilde{z}_i$ with an $\mathcal{L}_2$-gain less than or equal to θ, if there exists a $\mathcal{K}_\infty$-function $\beta : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ such that for any exogenous input $\chi_{i-1} \in \mathcal{L}_2$, and any initial condition $\tilde{\xi}_i(0)$, the corresponding solution to the system satisfies

$$\|\tilde{z}_i\|_{\mathcal{L}_2} \leq \beta(|\tilde{\xi}_i(0)|) + \theta\|\chi_{i-1}\|_{\mathcal{L}_2}. \tag{14}$$

To deeply co-design with the integrated radar systems and the proposed working scheme, in the next section, we will present a type of periodic event-triggered mechanism with asynchronous sampling scheme, while guaranteeing $\mathcal{L}_2$ stability. Note that, the mechanism is given with respect to a general form of linear system, and can easily be applied to the vehicle platoon presented in Section 3 and this section.

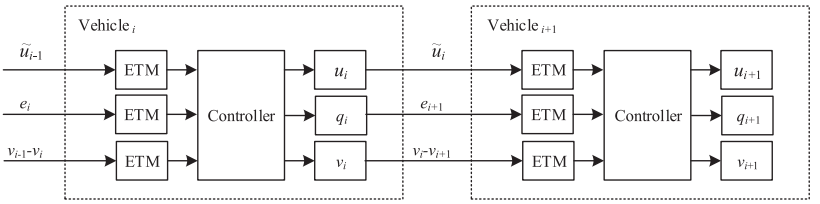


Fig. 3. System structure of periodic event-triggered controlled vehicle platoon, in which ETM is short for event-triggered mechanism.

5 PERIODIC EVENT-TRIGGERED CONTROL DESIGN

In this section, we present a novel PETC for linear output feedback systems. This mechanism considers asynchronous sampling and transmission delays. When applied to the vehicle platoon, the structure of the event-triggered mechanism is given in Figure 3.

Consider the following **linear time-invariant (LTI)** system:

$$\begin{cases} \dot{\xi}_p(t) = A_p\xi_p(t) + B_p\hat{v}(t) + Ew(t) \\ y(t) = C_p\xi_p(t), \end{cases} \tag{15}$$

where $\xi_p(t) \in \mathbb{R}^{n_p}$ denotes the state vector of the plant, $\hat{v}(t) \in \mathbb{R}^{n_v}$ denotes the input vector of the plant, $y(t) \in \mathbb{R}^{n_y}$ denotes the output vector of the plant. $w(t) \in \mathbb{R}^{n_w}$ denotes an unknown disturbance. A dynamic controller is given as:

$$\begin{cases} \xi_c(t_{k+1}) = A_c\xi_c(t_k) + B_c\hat{y}(t_k) \\ v(t_k) = C_c\xi_c(t_k) + D_c\hat{y}(t_k), \end{cases} \tag{16}$$

where $\xi_c(t_k) \in \mathbb{R}^{n_c}$ is the state vector of the controller, $\hat{y}(t_k) \in \mathbb{R}^{n_y}$ is the input vector of the controller, $v(t_k) \in \mathbb{R}^{n_v}$ is the output vector of the controller. The dynamic here means the controller has it own state. This controller executes periodically with an interval $h > 0$. Thus, an execution sequence of the controller can be achieved as:

$$\mathcal{T}_k := \{t_k | t_k := kh,\ k \in \mathbb{N}\}. \tag{17}$$

Define $\tau : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ be the elapsed time since the last execution time, that is:

$$\tau(t) := t - t_k,\ t \in [t_k, t_{k+1}].$$

Define the input and output of the plant (15) and controller (16) as:

$$\hat{u}(t_k) := \begin{bmatrix} \hat{y}(t_k) \\ \hat{v}(t_k) \end{bmatrix} \in \mathbb{R}^{n_u},\ u(t) := \begin{bmatrix} y(t) \\ v(t) \end{bmatrix} \in \mathbb{R}^{n_u},$$

where $n_u := n_y + n_v$. The input $\hat{u}(t_k)$ is updated from the output $u(t)$ via an event-triggered mechanism which will be discussed later.

For the update of $\hat{u}(t_k)$, there may be delays, these delays may caused by, e.g., transmissions and computations. In this paper, we assume that, for a specified system and event-triggered mechanism, for each element of the system output u_i, the delay τ_d^i comes from network transmissions and is invariant along the system timeline. This is possible since a static reserved MAC protocols, such as **time-division multiple access (TDMA)** protocols can provide an upper bound for the transmission delays. As for the vehicle platoons, the delays may come from measuring, computing, and face-to-face transmitting, all of which can be upper bounded. Besides, we introduce an assumption for the delays.

Assumption 5.1. *The delay satisfies*

$$\tau_d^i < h,\ \forall i \in \{1, \ldots, n_u\}.$$

Assumption 5.1 can be guaranteed by, e.g., a proper design of the working schemes and a careful choice of the parameters.

Inspired by [4], to analyze the dynamics of the system with respect to delays, we introduce a memory variable $s(t) \in \mathbb{R}^{n_u}$, which can be further detailed as $s(t) := [s_y^{\mathrm{T}}(t) \quad s_v^{\mathrm{T}}(t)]^{\mathrm{T}}$, and $s_y(t) \in \mathbb{R}^{n_y}$ and $s_v(t) \in \mathbb{R}^{n_v}$. Define:

$$\xi(t) := \begin{bmatrix} \xi_p^{\mathrm{T}}(t) & \xi_c^{\mathrm{T}}(t) & \hat{y}^{\mathrm{T}}(t) & \hat{v}^{\mathrm{T}}(tvvv) & s_y^{\mathrm{T}}(t) & s_v^{\mathrm{T}}(t) \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^{n_\xi},$$

where $n_\xi := n_p + n_c + 2n_u$. A performance variable of the plant (15) and controller (16) is given by:

$$z(t) := \bar{C}\xi(t) + \bar{D}w(t), \tag{18}$$

with $\bar{C}$ and $\bar{D}$ being properly designed matrices.

It is easy to see that, system (11), (12), and (13) is a special case of system (15), (16), (17), and (18). For system (15), (16), (17), and (18), the stability and performance we considered are given as follows.

Definition 5.2 (Exponential Stability [24]). For system (15), (16), (17), and (18), $t \in \mathbb{R}_0^+$, $\xi(t) \in \mathbb{R}^n$ is said to be **global exponential stable (GES)**, if there exist scalars $a, c \in \mathbb{R}^+$ such that for any $t_0 \geq 0$ the following holds:

$$|\xi(t, \xi(0))| \leq c|\xi(0)|e^{-at}, \forall t \geq t_0.$$

Definition 5.3 ($\mathcal{L}_2$-gain [14]). The system (15), (16), (17), and (18) is said to have an $\mathcal{L}_2$-gain from w to z smaller than or equal to γ, if there is a $\mathcal{K}_\infty$ function $\delta : \mathbb{R}^{n_\xi} \to \mathbb{R}^+$ such that for any $w \in \mathcal{L}_2$, any initial state $\xi(0) = \xi_0 \in \mathbb{R}^{n_\xi}$, the corresponding solution to the system satisfies

$$\|z\|_{\mathcal{L}_2} \leq \delta(\xi_0) + \gamma\|w\|_{\mathcal{L}_2}.$$

Next, the PETC with asynchronous sampling scheme is presented. In this scheme, each element of the system output is sampled periodically. However, when looking at the whole output vector, all the elements won't sample at the same time, but asynchronously at different times. Therefore, the PETC works in an asynchronous periodic manner. Whether to update the sampling or not is dependent on a localized event-triggered condition, which is given as follows.

Denote set $\mathcal{J}$ and $\mathcal{J}_c$ as:

$$\begin{aligned} &\forall i \in \{1, \ldots, n_u\},\ \kappa(t) = a\,constant: \\ &i \in \mathcal{J},\ \text{iff } |u_i(t_k) - \hat{u}_i(t_k)| > \sigma_i|u_i(t_k)| \wedge \mathcal{S}_u(l(t_k)) = i \\ &i \in \mathcal{J}_c,\ \text{iff } |u_i(t_k) - \hat{u}_i(t_k)| \leq \sigma_i|u_i(t_k)| \wedge \mathcal{S}_u(l(t_k)) = i, \end{aligned} \tag{19}$$

where

$$\begin{aligned} u(t) &= \begin{bmatrix} u_1(t) & \cdots & u_{n_u}(t) \end{bmatrix}^{\mathrm{T}} \\ \hat{u}(t) &= \begin{bmatrix} \hat{u}_1(t) & \cdots & \hat{u}_{n_u}(t) \end{bmatrix}^{\mathrm{T}}, \end{aligned}$$

and $\sigma_i, i \in \{1, \ldots, n_u\}$ are predesigned event-triggered condition parameters. Therefore, the event can be generated only based on the local information. The system will update the output $u_i(t)$, $i \in \mathcal{J}$ following a fixed sequence $\mathcal{S}_u$. $\mathcal{S}_u$ contains the sampling and update order of all the elements of $u(t)$, but those $i \in \mathcal{J}_c := \{1, \ldots, n_u\} \setminus \mathcal{J}$ will be skipped when updating. A typical $\mathcal{S}_u$ is $\mathcal{S}_u = \{1, \ldots, n_u\}$, which however might not always be applicable, due to, e.g., system design. Pointer $l : \mathbb{R}_0^+ \to \mathbb{N}$ indicates the next to be sampled element of $u(t)$. More specifically, at t_k, the $\mathcal{S}_u(l(t_k))$-th element of $u(t)$ should sample and validate the event-triggered condition. We call the process of going through $\mathcal{S}_u$ one time a *round*. That is, *round* represents the process of sampling

the whole vector of $u(t)$ following the sequence $\mathcal{S}_u$. Note that, the time required of going through a *round* is $n_u h$. Counter $\kappa : \mathbb{R}_0^+ \to \mathbb{N}$ indicates the total number of *rounds*. In (19), $\kappa(t) = a\,constant$ means the system is executing within a *round*.

Now we present the update law of both $s(t)$ and $\hat{u}(t)$:

$$\begin{aligned} s_i(t) &= \begin{cases} u_i(t_k), & \text{if } i \in \mathcal{J} \wedge \mathcal{S}_u(l(t_k)) = i \\ s_i(t_{k-1}), & \text{if } i \in \mathcal{J}_c \wedge \mathcal{S}_u(l(t_k)) \neq i \end{cases}, \text{ for } t \in [t_k, t_k + n_u h) \\ \hat{u}(t) &= s_i(t_k), \text{ for } t \in [t_k + \tau_d^i, t_k + n_u h + \tau_d^i), \end{aligned} \tag{20}$$

where $s(t) = [s_1(t) \quad \cdots \quad s_{n_u(t)}]^{\mathrm{T}}$. A **zero-order hold (ZOH)** mechanism is applied between two updates of both $s_i(t)$ and $\hat{u}(t)$.

Remark 5.4. In (20), $s(t)$ is a storage of $\hat{u}(t)$ by the networks. At each sampling time t_k, if $\mathcal{S}_u(l(t_k)) = i$, i.e., according to sequence $\mathcal{S}_u$, the coming sample and update happen at sensor i, this sensor will sample the corresponding system output and check if there is a local event according to the event condition in (19). If there is one, then $i \in \mathcal{J}$, and this $s_i(t_k)$ will be updated by $u_i(t_k)$. After the update, $s_i(t_k)$ will be injected to the network immediately. After holding for τ_d^i period, at $t_k + \tau_d^i$, the time when the $s_i(t_k)$ is received, $\hat{u}_i(t_k + \tau_d^i)$ will be updated by $s_i(t_k)$. For all t_k that $\mathcal{S}_u(l(t_k)) \neq i$, $s_i(t_k)$ will keep the same; for all $t_k + \tau_d^i$ that $\mathcal{S}_u(l(t_k + \tau_d^i)) \neq i$, $\hat{u}_i(t_k + \tau_d^i)$ will also keep the same.

From (20), it is easy to see that, at the update time of $s(t)$ and $\hat{u}(t)$, the following hold: if $\mathcal{S}_u(l(t_k)) = i$ and

$$\begin{aligned} i \in \mathcal{J} \quad & \text{then } \xi^{\mathrm{T}}(t_k)\acute{Q}_i\xi(t_k) > 0 \\ i \in \mathcal{J}_c \quad & \text{then } \xi^{\mathrm{T}}(t_k)\acute{Q}_i\xi(t_k) \leq 0 \\ i \in \mathcal{J} \quad & \text{then } \xi^{\mathrm{T}}(t_k)\tilde{Q}_i\xi(t_k) > 0 \\ i \in \mathcal{J}_c \quad & \text{then } \xi^{\mathrm{T}}(t_k)\tilde{Q}_i\xi(t_k) \leq 0 \\ i \in \mathcal{J} \quad & \text{then } \xi^{\mathrm{T}}(t_k + \tau_d^i)\hat{Q}_i\xi(t_k + \tau_d^i) > 0 \\ i \in \mathcal{J}_c \quad & \text{then } \xi^{\mathrm{T}}(t_k + \tau_d^i)\hat{Q}_i\xi(t_k + \tau_d^i) \leq 0, \end{aligned}$$

where

$$\acute{Q}_i := \begin{bmatrix} (1-\sigma_i)C^{\mathrm{T}}\Gamma_i C & \mathbf{0} & (1-\sigma_i)C^{\mathrm{T}}\Gamma_i D - C^{\mathrm{T}}\Gamma_i \\ * & \mathbf{0} & \mathbf{0} \\ * & * & (D-I)^{\mathrm{T}}\Gamma_i(D-I) - \sigma_i D^{\mathrm{T}}\Gamma_i D \end{bmatrix}$$

$$\tilde{Q}_i := \begin{bmatrix} (1-\sigma_i)C^{\mathrm{T}}\Gamma_i C & (1-\sigma_i)C^{\mathrm{T}}\Gamma_i D - C^{\mathrm{T}}\Gamma_i & \mathbf{0} \\ * & (D-I)^{\mathrm{T}}\Gamma_i(D-I) - \sigma_i D^{\mathrm{T}}\Gamma_i D & \mathbf{0} \\ * & * & \mathbf{0} \end{bmatrix}$$

$$\hat{Q}_i := \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ * & \Gamma_i & -\Gamma_i \\ * & * & (1-\sigma_i)\Gamma_i \end{bmatrix}.$$

with I being an identity matrix of proper dimension, $\mathbf{0}$ is a zero matrix of proper dimension, and

$$\Gamma_i := \mathrm{diag}\{\Gamma_i^y, \Gamma_i^v\} := \mathrm{diag}\{\gamma_i^1, \ldots, \gamma_i^{n_y}, \gamma_i^{n_y+1}, \ldots, \gamma_i^{n_u}\},$$

in which $\gamma_i^j = 1$ when $i = j$, otherwise $\gamma_i^j = 0$. Therefore, γ_i^j indicates which element of the system output $u_i(t)$ the event-triggered condition in (19) is validating. Besides,

$$C := \begin{bmatrix} C_p & \mathbf{0} \\ \mathbf{0} & C_c \end{bmatrix}, \; D := \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ D_c & \mathbf{0} \end{bmatrix},$$

Since the sampling and update are close to each other, we introduce $\eta : \mathbb{R}_0^+ \to \{0, 1\}$ to specify the jump dynamics, i.e., when $\eta(t) = 1$, the next jump is an update; when $\eta(t) = 0$, the next jump is a sampling.

Now we present an impulsive system based on (15), (16), (17), (18), (19), and (20), given as:

$$\begin{bmatrix} \dot{\xi}(t) \\ \dot{\tau}(t) \\ \dot{l}(t) \\ \dot{\kappa}(t) \\ \dot{\eta}(t) \end{bmatrix} = \begin{bmatrix} \bar{A}\xi(t) + \bar{B}w(t) \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \text{ when } \begin{cases} \tau(t) \in \mathbb{R}_0^+ \\ \tau(t) \neq \tau_d^l \\ \tau(t) \neq h \end{cases} \tag{21a}$$

$$\begin{bmatrix} \xi(t^+) \\ \tau(t^+) \\ l(t^+) \\ \kappa(t^+) \\ \eta(t^+) \end{bmatrix} = \begin{bmatrix} \acute{J}_{\mathcal{S}_u(l)}\xi(t) \\ 0 \\ l(t) \\ \kappa(t) \\ 1 \end{bmatrix}, \text{ when } \begin{cases} \tau(t) = h \\ \mathcal{S}_u(l) \in \mathcal{J} \\ \eta(t) = 0 \end{cases} \tag{21b}$$

$$\begin{bmatrix} \xi(t^+) \\ \tau(t^+) \\ l(t^+) \\ \kappa(t^+) \\ \eta(t^+) \end{bmatrix} = \begin{bmatrix} \tilde{J}_0\xi(t) \\ 0 \\ l(t) + 1 \\ \kappa(t) \\ 0 \end{bmatrix}, \text{ when } \begin{cases} \tau(t) = h \\ l(t) < n_u \\ \mathcal{S}_u(l) \in \mathcal{J}_c \\ \eta(t) = 0 \end{cases} \tag{21c}$$

$$\begin{bmatrix} \xi(t^+) \\ \tau(t^+) \\ l(t^+) \\ \kappa(t^+) \\ \eta(t^+) \end{bmatrix} = \begin{bmatrix} \tilde{J}_0\xi(t) \\ 0 \\ 1 \\ \kappa(t) + 1 \\ 0 \end{bmatrix}, \text{ when } \begin{cases} \tau(t) = h \\ l(t) = n_u \\ \mathcal{S}_u(l) \in \mathcal{J}_c \\ \eta(t) = 0 \end{cases} \tag{21d}$$

$$\begin{bmatrix} \xi(t^+) \\ \tau(t^+) \\ l(t^+) \\ \kappa(t^+) \\ \eta(t^+) \end{bmatrix} = \begin{bmatrix} \tilde{J}_{\mathcal{S}_u(l)}\xi(t) \\ \tau(t) \\ l(t) + 1 \\ \kappa(t) \\ 0 \end{bmatrix}, \text{ when } \begin{cases} \tau(t) = \tau_d^l \\ l < n_u \\ \eta(t) = 1 \end{cases} \tag{21e}$$

$$\begin{bmatrix} \xi(t^+) \\ \tau(t^+) \\ l(t^+) \\ \kappa(t^+) \\ \eta(t^+) \end{bmatrix} = \begin{bmatrix} \tilde{J}_{\mathcal{S}_u(n_u)}\xi(t) \\ \tau(t) \\ 1 \\ \kappa(t) + 1 \\ 0 \end{bmatrix}, \text{ when } \begin{cases} \tau(t) = \tau_d^l \\ l = n_u \\ \eta(t) = 1 \end{cases} \tag{21f}$$

$$z(t) = \bar{C}\xi(t) + \bar{D}w(t). \tag{21g}$$

where

$$\bar{A} := \begin{bmatrix} A_p & \mathbf{0} & \mathbf{0} & B_p & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \bar{B} := \begin{bmatrix} E \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}$$

and

$$\acute{J}_i = \begin{bmatrix} I & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & A_c & B_c & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & I & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & I & \mathbf{0} & \mathbf{0} \\ \Gamma_i^y C_p & \mathbf{0} & \mathbf{0} & \mathbf{0} & (I-\Gamma_i^y) & \mathbf{0} \\ \mathbf{0} & \Gamma_i^v C_c & \Gamma_i^v D_c & \mathbf{0} & \mathbf{0} & (I-\Gamma_i^v) \end{bmatrix}$$

$$\tilde{J}_i := \begin{bmatrix} I & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & I & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & I-\Gamma_i^y & \mathbf{0} & \Gamma_i^y & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & I-\Gamma_i^v & \mathbf{0} & \Gamma_i^v \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & I & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & I \end{bmatrix}$$

$$\tilde{J}_0 = \begin{bmatrix} I & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & A_c & B_c & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & I & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & I & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & I & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & I \end{bmatrix}.$$

It is easy to see that (21b) models the jumps at sampling times t_k, $k \in \mathbb{N}$ when there is an event; (21c) and (21d) model the jumps at sampling times t_k, $k \in \mathbb{N}$ when there is no event; (21e) and (21f) model the jumps at update times $t_k + \tau_d^i$.

For the impulsive system (21), to analyze its stability and $\mathcal{L}_2$-gain performance, we consider the following Lyapunov function:

$$V(\xi, \tau, l, \eta) = \begin{cases} \xi^{\mathrm{T}} P_{1l}(\tau)\xi, \text{ when } \tau \in [0, \tau_d^l],\ \eta = 1 \\ \xi^{\mathrm{T}} P_{0l}(\tau)\xi, \text{ when } \tau \in [0, h],\ \eta = 0, \end{cases} \tag{22}$$

where $P_{1i}(\tau)$ and $P_{0i}(\tau)$ satisfy:

$$\frac{\mathrm{d}}{\mathrm{d}\tau}P_{1i} = -\bar{A}^{\mathrm{T}}P_{1i} - P_{1i}\bar{A} - 2\rho P_{1i} - \gamma^{-2}\bar{C}^{\mathrm{T}}\bar{C} - G_{1i}^{\mathrm{T}}MG_{1i}$$
$$\frac{\mathrm{d}}{\mathrm{d}\tau}P_{0i} = -\bar{A}^{\mathrm{T}}P_{0i} - P_{0i}\bar{A} - 2\rho P_{0i} - \gamma^{-2}\bar{C}^{\mathrm{T}}\bar{C} - G_{0i}^{\mathrm{T}}MG_{0i}$$

where M is defined as;

$$M := (I - \gamma^{-2}\bar{D}^{\mathrm{T}}\bar{D})^{-1}$$

and

$$G_{1i} := \bar{B}^{\mathrm{T}}P_{1i} + \gamma^{-2}\bar{D}^{\mathrm{T}}\bar{C}$$
$$G_{0i} := \bar{B}^{\mathrm{T}}P_{0i} + \gamma^{-2}\bar{D}^{\mathrm{T}}\bar{C}$$

Construct the Hamiltonian matrix:

$$H := \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix},$$

where

$$H_{11} := \bar{A} + \rho I + \gamma^{-2}\bar{B}M\bar{D}^{\mathrm{T}}\bar{C}$$
$$H_{12} := \bar{B}M\bar{B}^{\mathrm{T}}$$
$$H_{21} := -\bar{C}^{\mathrm{T}}(\gamma^2 I - \bar{D}\bar{D}^{\mathrm{T}})^{-1}\bar{C}$$
$$H_{22} := -(\bar{A} + \rho I + \gamma^{-2}\bar{B}M\bar{D}^{\mathrm{T}}\bar{C})^{\mathrm{T}}.$$

and its matrix exponential:

$$F(r) := e^{-Hr} = \begin{bmatrix} F_{11}(r) & F_{12}(r) \\ F_{21}(r) & F_{22}(r) \end{bmatrix}. \tag{23}$$

Assumption 5.5. *$F_{11}(r)$ is invertible $\forall r \in [0, h]$.*

Since $F_{11}(0) = I$ and $F_{22}(\tau)$ is continuous, Assumption 5.5 can always be satisfied for sufficiently small h. If Assumption 5.5 holds, then $-F_{11}^{-1}(h)F_{12}(h)$ is positive semi-definite. Define the matrix $S(r)$ satisfying $S(r)S^{\mathrm{T}}(r) := -F_{11}^{-1}(r)F_{12}(r)$. For the sake of brevity, we define $P_{1id} := P_{1\mathcal{S}_u(i)}(\tau_d^{\mathcal{S}_u(i)})$ and $P_{0ih} := P_{0\mathcal{S}_u(i)}(h)$.

Theorem 5.6. *Consider the system (21), event-triggered mechanism (19), input update law (20), Lyapunov function (22), and Assumption 5.5 and 5.1 hold. Let $\gamma^2 > \lambda_{\max}(\bar{D}^{\mathrm{T}}\bar{D})$, $\rho > 0$. If there exist matrices $P_{1id} \succ 0$, $P_{0ih} \succ 0$, scalars $\acute{\mu}^i_{\mathcal{J}} \geq 0$, $\acute{\mu}^i_{\mathcal{J}_c} \geq 0$, $\tilde{\mu}^i_{\mathcal{J}} \geq 0$, $\tilde{\mu}^i_{\mathcal{J}_c} \geq 0$, $\mu_i \geq 0$ $i \in \{1, \ldots, n_u\}$, such that the following LMIs hold:*

$$\begin{bmatrix} P_{0ih} + G_{1i} & \acute{J}^{\mathrm{T}}_{\mathcal{S}_u(i)}F_{11}^{-\mathrm{T}}(\tau_d^{\mathcal{S}_u(i)})P_{1id}S(\tau_d^{\mathcal{S}_u(i)}) & \acute{J}^{\mathrm{T}}_{\mathcal{S}_u(i)}H_{1i} \\ * & I - S^{\mathrm{T}}(\tau_d^{\mathcal{S}_u(i)})P_{1id}S(\tau_d^{\mathcal{S}_u(i)}) & \boldsymbol{0} \\ * & * & H_{1i} \end{bmatrix} \succ 0 \tag{24a}$$

$$\begin{bmatrix} P_{1id} + G_{2i} & \tilde{J}^{\mathrm{T}}_{\mathcal{S}_u(i)}F_{11}^{-\mathrm{T}}(h - \tau_d^{\mathcal{S}_u(i)})H_{4i} & \tilde{J}^{\mathrm{T}}_{\mathcal{S}_u(i)}H_{2i} \\ * & I - S^{\mathrm{T}}(h - \tau_d^{\mathcal{S}_u(i)})H_{4i} & \boldsymbol{0} \\ * & * & H_{2i} \end{bmatrix} \succ 0 \tag{24b}$$

$$\begin{bmatrix} P_{0ih} - \mu_i\hat{Q}_i & \tilde{J}_0^{\mathrm{T}}F_{11}^{-\mathrm{T}}(h)P_{0i^+h}S(h) & \tilde{J}_0^{\mathrm{T}}H_{3i} \\ * & I - S^{\mathrm{T}}(h)P_{0i^+h}S(h) & \boldsymbol{0} \\ * & * & H_{3i} \end{bmatrix} \succ 0 \tag{25}$$

where

$$G_{1i} := -\acute{\mu}^i_{\mathcal{J}}\acute{Q}_i - \tilde{\mu}^i_{\mathcal{J}}\tilde{Q}_i$$
$$G_{2i} := \acute{\mu}^i_{\mathcal{J}_c}\acute{Q}_i + \tilde{\mu}^i_{\mathcal{J}_c}\tilde{Q}_i$$
$$H_{1i} := F_{11}^{-\mathrm{T}}(\tau_d^{\mathcal{S}_u(i)})P_{1id}F_{11}^{-1}(\tau_d^{\mathcal{S}_u(i)}) + F_{21}(\tau_d^{\mathcal{S}_u(i)})F_{11}^{-1}(\tau_d^{\mathcal{S}_u(i)})$$
$$H_{2i} := F_{11}^{-\mathrm{T}}(h - \tau_d^{\mathcal{S}_u(i)})P_{0i^+h}F_{11}^{-1}(h - \tau_d^{\mathcal{S}_u(i)}) + F_{21}(h - \tau_d^{\mathcal{S}_u(i)})F_{11}^{-1}(h - \tau_d^{\mathcal{S}_u(i)})$$
$$H_{3i} := F_{11}^{-\mathrm{T}}(h)P_{0i^+h}F_{11}^{-1}(h) + F_{21}(h)F_{11}^{-1}(h)$$
$$H_{4i} := P_{0i^+h}S(h - \tau_d^{\mathcal{S}_u(i)})$$
$$i^+ = \begin{cases} i + 1, & \text{if } i < n_u \\ 1, & \text{if } i = n_u, \end{cases}$$

then the impulsive system (21) is GES with convergence rate ρ when $w = 0$; and the $\mathcal{L}_2$-gain from w to z is smaller than or equal to γ.

Proof. The basic idea of the proof follows the proof of Theorem III.4 of [4] and the proof of Theorem III.2 of [14]. To proof the result of Theorem 5.6, there are three steps. The first step is to show that, the function V satisfies $\forall t \in \mathbb{R}_0^+$:

$$\tilde{c}_1|\xi|^2 \leq V(\xi, \tau, l, \eta) \leq \tilde{c}_2|\xi|^2,$$

where c_1 and c_2 are scalars satisfy $0 < c_1 \leq c_2$. According to [4] and [14], it holds that $\forall \tau \in [0, \tau_d^{\mathcal{S}_u(i)}]$, $P_{1i}(\tau) \succ 0$; $\forall \tau \in [0, h]$, $P_{0i}(\tau) \succ 0$. Besides:

$$\tilde{c}_1 = \min_{i \in \{1,\ldots,n_u\}} \{ \min_{\tau \in [0, \tau_d^i]} \lambda_{\min}(P_{1i}(\tau)), \min_{\tau \in [0,h]} \lambda_{\min}(P_{0i}(\tau))\}$$
$$\tilde{c}_2 = \max_{i \in \{1,\ldots,n_u\}} \{ \max_{\tau \in [0, \tau_d^i]} \lambda_{\max}(P_{1i}(\tau)), \max_{\tau \in [0,h]} \lambda_{\max}(P_{0i}(\tau))\}.$$

The second step is to show that V has a supply rate $\theta^{-2}z^{\mathrm{T}}z - w^{\mathrm{T}}w$ during flow. Since $\gamma^2 > \lambda_{\max}(\bar{D}^{\mathrm{T}}\bar{D})$ from the hypothesis of the theorem, by derivating V along t, the following inequation is obtained directly: $\forall t \in [0, h]$, $t \neq \tau_d^i$, $i \in \{1, \ldots, n_u\}$:

$$\frac{\mathrm{d}}{\mathrm{d}t}U(t) \leq -2\rho V(t) - \gamma^{-2}z^{\mathrm{T}}(t)z(t) + w^{\mathrm{T}}(t)w(t).$$

The third step is to show that V does not increase during jumps. According to the proof of Theorem III.2 of [14], it is easy to obtain:

$$\begin{aligned}
P_{1i}(0) = {} & F_{21}(\tau_d^i)F_{11}^{-1}(\tau_d^i) + F_{11}^{-\mathrm{T}}(\tau_d^i)(P_{1i}(\tau_d^i) + P_{1i}(\tau_d^i)S(\tau_d^i)(I - S^{\mathrm{T}}(\tau_d^i)P_{1i}(\tau_d^i)S(\tau_d^i))^{-1}S^{\mathrm{T}}(\tau_d^i) \\
& P_{1i}(\tau_d^i))F_{11}^{-1}(\tau_d^i) \\
P_{0i}(0) = {} & F_{21}(h)F_{11}^{-1}(h) + F_{11}^{-\mathrm{T}}(h)(P_{0i}(h) + P_{0i}(h)S(h)(I - S^{\mathrm{T}}(h)P_{0i}(h)S(h))^{-1}S^{\mathrm{T}}(h)P_{0i}(h))F_{11}^{-1}(h) \\
P_{0i}(\tau_d^i) = {} & F_{21}(h - \tau_d^i)F_{11}^{-1}(h - \tau_d^i) + F_{11}^{-\mathrm{T}}(h - \tau_d^i)(P_{0i}(h) + P_{0i}(h)S(h - \tau_d^i)(I - S^{\mathrm{T}}(h - \tau_d^i) \\
& P_{0i}(h)S(h - \tau_d^i))^{-1}S^{\mathrm{T}}(h - \tau_d^i)P_{0i}(h))F_{11}^{-1}(h - \tau_d^i)
\end{aligned}$$

When there is a sample and update on $u_i(t)$, By applying Schur complement and S-procedure (see e.g., [7]) to (24), it holds that:

$$\xi^{\mathrm{T}}(t_k)P_{0i}(h)\xi(t_k) \geq \xi^{\mathrm{T}}(t_k)\acute{J}_i^{\mathrm{T}}P_{1i}(0)\acute{J}_i\xi(t_k)$$
$$\xi^{\mathrm{T}}(t_k + \tau_d^i)P_{1i}(\tau_d^i)\xi(t_k + \tau_d^i) \geq \xi^{\mathrm{T}}(t_k + \tau_d^i)\tilde{J}_i^{\mathrm{T}}P_{0i^+}(\tau_d^i)\tilde{J}_i\xi(t_k + \tau_d^i)$$

which means the Lyapunov function V does not increase during jumps. When there is no update on $u_i(t)$, it follows from (25) that:

$$\xi^{\mathrm{T}}(t_k)P_{0i}(h)\xi(t_k) \geq \xi^{\mathrm{T}}(t_k)\tilde{J}_0^{\mathrm{T}}P_{0i^+}(0)\tilde{J}_0\xi(t_k).$$

which means the Lyapunov function V does not increase during jumps. By now, we conclude that V is a proper storage function and the system is GES with convergence rate ρ when $w = 0$ and has an $\mathcal{L}_2$-gain from w to z smaller than or equal to γ. This ends the proof. □

Remark 5.7. For LMIs (24) and (25), the given parameters are ρ and γ; and the design parameters are matrix P_{1id}, P_{0ih}, scalars $\acute{\mu}_{\mathcal{J}}^i$, $\acute{\mu}_{\mathcal{J}_c}^i$, $\tilde{\mu}_{\mathcal{J}}^i$, $\tilde{\mu}_{\mathcal{J}_c}^i$, μ_i, $i \in \{1, \ldots, n_u\}$. The given parameters ρ and γ determine the convergence rate and $\mathcal{L}_2$-gain of the system. Once the stability and performance are given, both parameters are decided. The next step is to find proper design parameters satisfy $P_{1id} \succ 0$, $P_{0ih} \succ 0$, scalars $\acute{\mu}_{\mathcal{J}}^i \geq 0$, $\acute{\mu}_{\mathcal{J}_c}^i \geq 0$, $\tilde{\mu}_{\mathcal{J}}^i \geq 0$, $\tilde{\mu}_{\mathcal{J}_c}^i \geq 0$, $\mu_i \geq 0$, $i \in \{1, \ldots, n_u\}$; while the LMIs are feasible, then the designed stability and performance can be guaranteed, regardless of the scenarios. This is the theoretical guarantee for the control performance. If one cannot find such design parameters, then the designed stability and performance cannot be guaranteed. We

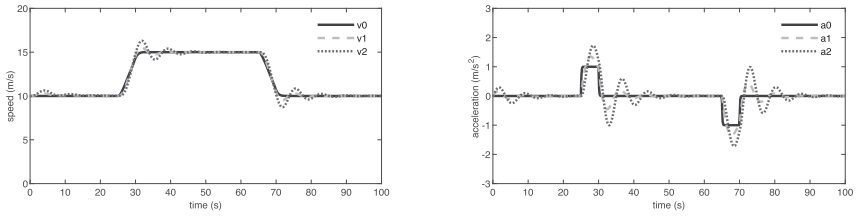


Fig. 4. Velocities (left) and accelerations (right) of each vehicle in the platoon.

need to either relax the designed stability and performance, i.e., reduce the convergence rate ρ and increase the $\mathcal{L}_2$-gain γ, and check the LMIs again; or change to other ETC approaches.

Remark 5.8. Since the convergence rate ρ and $\mathcal{L}_2$-gain γ enter the LMIs in a nonlinear way, it is difficult to compute the sets of feasible ρ and γ directly when the event-triggering parameters σ_i, $i \in \{1, \ldots, n_u\}$ and other parameters are given. Instead, we can apply a searching algorithm to approximate the sets. More specifically, given some parameters, e.g., σ_i, $i \in \{1, \ldots, n_u\}$, by increasing ρ from a sufficient small value with fixed steps, the sets of the feasible ρ can be computed via solving the LMIs.

Remark 5.9. When applying the control and communication co-design approach, the switching time of each mode of the radar system should be equal to the sampling interval h. According to the periodic event-triggered mechanism and asynchronous sampling mechanism, whenever there is an update from any element of the system output, i.e., $e_i(t)$, $\dot{e}_i(t)$, and $\tilde{u}_{i-1}(t)$, the controller will execute once.

Remark 5.10. According to (11), (12), and (13), since the model only consider the parameters of the current vehicle and its predecessor, thus within a sensing and communication integrated radar system, the clock should be synchronized to guarantee that $e_i(t)$, $\dot{e}_i(t)$, and $\tilde{u}_{i-1}(t)$ are sampled following the working scheme. However, for different integrated radar systems in the same vehicle platoon, the clocks are not necessarily to be synchronized. Also, the radars in each vehicle are not required to switch to the same mode at the same time, except those belong to the same integrated radar systems.

6 NUMERICAL VALIDATION

In this section, we validate the proposed periodic event-triggered mechanism in the vehicle platoon. We consider a platoon consists of three vehicles from [9]. The codes simulating the dynamics of each vehicle are borrowed from Matlab Vehicle Platooning Example. The vehicle parameters are given by $\tilde{h} = 0.6$, $\tilde{\tau}_d = 0.1$, $L_i = 5$, $r_i = 0$. The control gain is given by $k_p = 0.1$, $k_d = 0.9$. The sampling interval is $h = 0.01s$, the delay is $\tau_d = 0.001s$. The switching time of each mode of the radar system is also equal to $h = 0.01s$. The parameters for the event-triggered mechanism is given as $\sigma_i = 0.001$, $\forall i \in \{1, 2, 3\}$, $\rho = 0.001$. The initial state is given as $d_1(0) = 2m$, $d_2(0) = 2m$, $v_0(0) = v_1(0) = v_2(0) = 10m/s$, $a_0(0) = a_1(0) = a_2(0) = 0m/s^2$.

A feasible solution can be found when $\gamma = 9000$ by solving the LMIs (24) and (25). Figure 4 shows the velocities and accelerations of each vehicle. Figure 5 shows the position errors and speed errors of $\mathcal{V}_1$ and $\mathcal{V}_2$. We can see that, given the initial position errors to be $2m$, the followers $\mathcal{V}_1$ and $\mathcal{V}_2$ can keep up with the leader, and position errors converge to 0. When given acceleration signal to the leader $\mathcal{V}_0$, $\mathcal{V}_1$ and $\mathcal{V}_2$ can still follow the leader. The maximum position errors and speed errors are all bounded.

Figure 6 shows the inter event intervals of $\mathcal{V}_1$ and $\mathcal{V}_2$. The inter event intervals from $\tilde{u}_0$ are $25s$, $5s$, $35s$, and $5s$, respectively, showing that, events are triggered when the acceleration signal

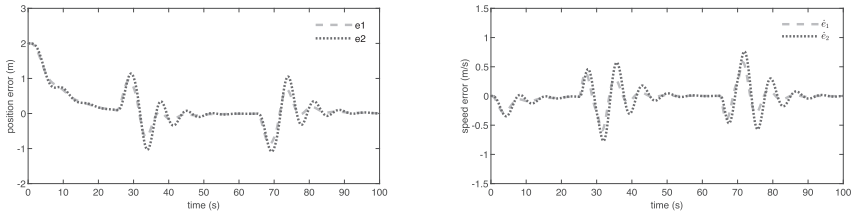


Fig. 5. Position errors (left) and speed errors (right) of $\mathcal{V}_1$ and $\mathcal{V}_2$: the followers in the platoon.

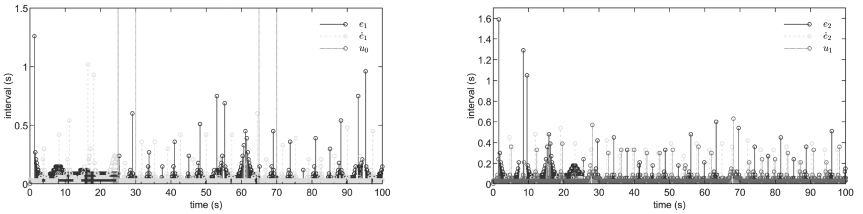


Fig. 6. Inter event intervals of $\mathcal{V}_1$ (left) and $\mathcal{V}_2$ (right). The event intervals from u_0 are $25s$, $5s$, $35s$, and $5s$, respectively.

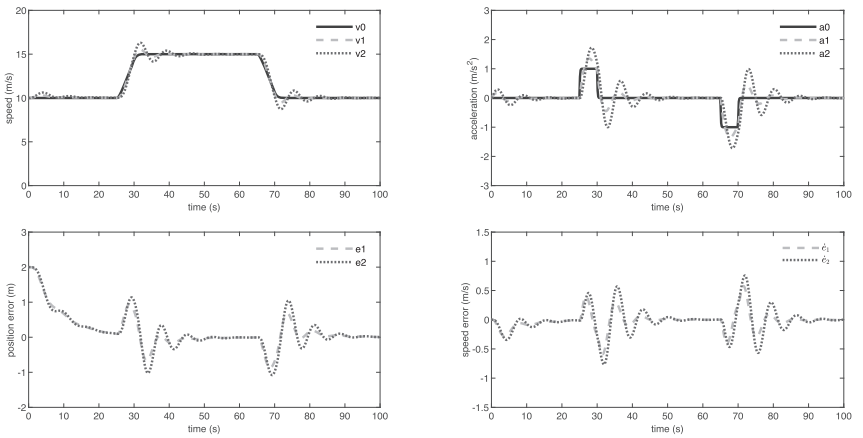


Fig. 7. Applying periodic time-triggered control with the same working scheme: the velocities, accelerations, position errors, and speed errors, showing that the performance is almost the same compared with that of applying the proposed periodic event-triggered control.

changes. Further simulation results show that, e_1 triggered 1,825 events, $\dot{e}_1$ triggered 2,504 events, $\tilde{u}_0$ triggered 4 events; e_2 triggered 2,283 events, $\dot{e}_2$ triggered 2,770 events, $\tilde{u}_1$ triggered 2,753 events. Therefore, a total of 12,139 events are triggered. If applying periodic TTC with the same working scheme, the triggering number goes up to 20,000. Figure 7 shows the simulation result applying TTC with the same working scheme. It is easy to see that, the proposed PETC can save up to 39.31% triggers compared to TTC, and with almost the same performance, i.e., the same convergence rate ρ and $\mathcal{L}_2$-gain γ. We can conclude that, the PETC can indeed saves system resources.

Figure 8 shows the simulation result of feasible convergence rate ρ and $\mathcal{L}_2$-gain γ with different $\sigma_1 = \sigma_2 = \sigma_3$ and $\tau_d^1 = \tau_d^2 = \tau_d^3$. It is easy to see that, σ_i determines the convergence rate ρ, a smaller σ_i can increase the rate, while a smaller ρ can tolerant a bigger σ_i, and thus results in fewer events. Also, the simulation result shows that, γ is mostly influenced by the delay τ_d^i. Note

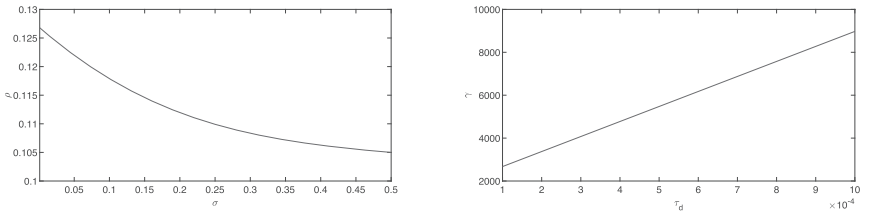


Fig. 8. Simulation result of feasible convergence rate ρ and $\mathcal{L}_2$-gain γ: the left figure shows when given different $\sigma_1 = \sigma_2 = \sigma_3$, the maximum feasible ρ; the right figure shows when given different delay $\tau_d^1 = \tau_d^2 = \tau_d^3$, the minimum feasible γ.

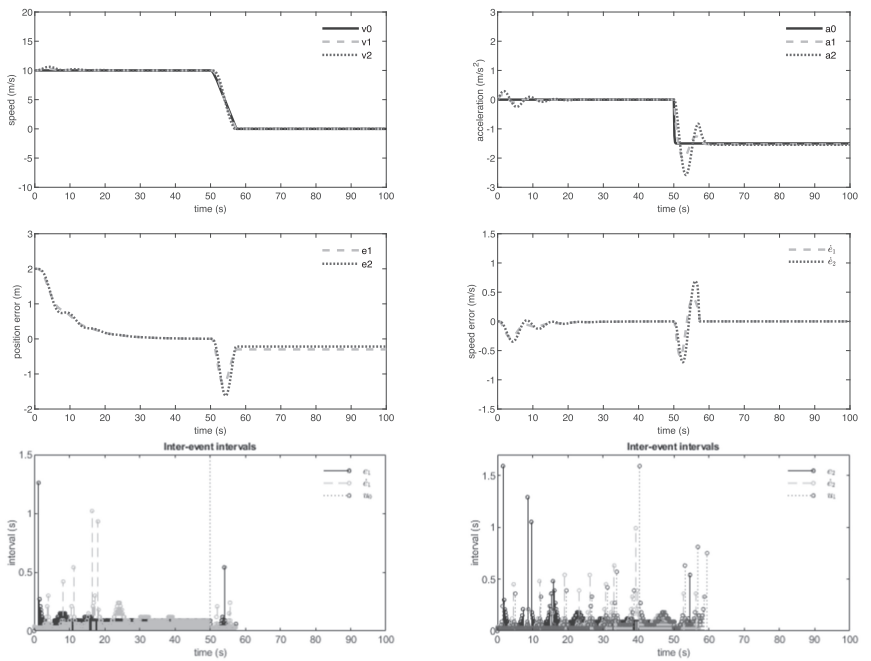


Fig. 9. Simulation result when sudden brake happens at 50s of the simulation: the velocities, accelerations, position errors, speed errors, and controller execution intervals.

that, the γ computed is indeed very conservative. Let $\tau_d^i = 0$, $\forall i \in \{1, 2, 3\}$, a feasible solution can be found when $\gamma = 1.4$. Therefore, it is important to increase the performance of the embedded systems to reduce τ_d^i.

We also present the simulation result when sudden brake happens, which is shown in Figure 9. From the simulation result, one can see that, there is a trade-off between the maximum acceleration during braking and the space interval between neighbored vehicles. A bigger acceleration requires a longer interval, a shorter interval requires the maximum acceleration to be smaller.

7 CONCLUSION

Short range robust wireless communication and control algorithms are two of the key problems in **Cooperative Adaptive Cruise Control (CACC)** for vehicle platooning. In this work, we proposed a control and communication co-design for CACC. We first presented an integrated radar

system, which combines sensing and communicating. The communications via radar systems are face-to-face and thus robust when there are plenty vehicles nearby. A working scheme was then presented for the radar systems. With this working scheme, the radar systems can switch between different working modes periodically and asynchronously, and thus the measurements and transmissions could work independently without interference. After which, a new type of periodic event-triggered control (PETC) approach was proposed for the vehicle platoons. This PETC can exploit the presented radar systems and working schemes, and guarantees string stability with respect to delays. Meanwhile, thanks to the event-triggered mechanism, system resources can be saved. Future works include applying the co-design approach to real vehicle platoons.

REFERENCES

[1] Mahmoud Abdelrahim, Romain Postoyan, Jamal Daafouz, and Dragan Nešić. 2016. Stabilization of nonlinear systems using event-triggered output feedback controllers. *IEEE Trans. Automat. Control* 61, 9 (2016), 2682–2687.

[2] Özgür B. Akan and Muharrem Arik. 2020. Internet of radars: Sensing versus sending with joint radar-communications. *IEEE Communications Magazine* 58 (2020), 13–19.

[3] Mohammadhadi Balaghiinaloo, Duarte J. Antunes, and W. P. M. H. Heemels. 2021. An L2-consistent event-triggered control policy for linear systems. *Automatica* 125 (2021), 109412.

[4] D. P. Borgers, V. S. Dolk, and W. P. M. H. Heemels. 2018. Riccati-based design of event-triggered controllers for linear systems with delays. *IEEE Trans. Automat. Control* 63, 1 (2018), 174–188.

[5] D. P. Borgers, Romain Postoyan, Adolfo Anta, Paulo Tabuada, Dragan Nesic, and W. P. M. H. Heemels. 2018. Periodic event-triggered control of nonlinear systems using overapproximation techniques. *Automatica* 94 (2018), 81–87.

[6] Alessandro Borri, Pierdomenico Pepe, Ilaria Di Loreto, and Mario Di Ferdinando. 2021. Finite-dimensional periodic event-triggered control of nonlinear time-delay systems with an application to the artificial pancreas. *IEEE Control Systems Letters* 5 (2021), 31–36.

[7] Stephen P. Boyd, Laurent El Ghaoui, Eric Feron, and Venkataramanan Balakrishnan. 1994. *Linear Matrix Inequalities in System and Control Theory*, Vol. 15. SIAM.

[8] Victor S. Dolk. 2017. *Resource-aware and Resilient Control : With Applications to Cooperative Driving*. Ph.D. Dissertation. Technische Universiteit Eindhoven.

[9] Victor S. Dolk, Jeroen Ploeg, and W. P. M. H. Heemels. 2017. Event-triggered control for string-stable vehicle platooning. *IEEE Transactions on Intelligent Transportation Systems* 18 (2017), 3486–3500.

[10] Shuo Feng, Yi Zhang, Shengbo Eben Li, Zhong Cao, Henry X. Liu, and Lei Li. 2019. String stability for vehicular platoon control: Definitions and analysis methods. *Annual Reviews in Control* 47 (2019), 81–97.

[11] Razvan A. Gheorghiu, Valentin Iordache, and Marius Minea. 2019. Assessment of ZigBee communications efficiency for truck platooning applications. *2019 11th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, 1–4.

[12] Yanpeng Guan, Qing-Long Han, and Xiaohua Ge. 2018. On asynchronous event-triggered control of decentralized networked systems. *Information Sciences* 425 (2018), 127–139.

[13] Yanpeng Guan, Qing-Long Han, and Chen Peng. 2013. Decentralized event-triggered control for sampled-data systems with asynchronous sampling. In *American Control Conference (ACC), 2013*. IEEE, 6565–6570.

[14] W. P. M. H. Heemels, M. C. F. Donkers, and Andrew R. Teel. 2013. Periodic event-triggered control for linear systems. *IEEE Trans. Automat. Control* 58, 4 (2013), 847–861.

[15] Dongyao Jia, Kejie Lu, Jianping Wang, Xiang Zhang, and Xuemin Shen. 2016. A survey on platoon-based vehicular cyber-physical systems. *IEEE Communications Surveys & Tutorials* 18 (2016), 263–284.

[16] Dongyao Jia and Dong Ngoduy. 2016. Platoon based cooperative driving model with consideration of realistic inter-vehicle communication. *Transportation Research Part C-emerging Technologies* 68 (2016), 245–264.

[17] Alexander Johansson, Ting Bai, Karl Henrik Johansson, and Jonas Martensson. 2023. Platoon cooperation across carriers: From system architecture to coordination. *IEEE Intelligent Transportation Systems Magazine* 15, 3 (2023), 132–144.

[18] Hassan K. Khalil. 1996. *Noninear Systems*. Prentice-Hall, New Jersey.

[19] Kuo-Yun Liang, Jonas Mårtensson, and Karl Henrik Johansson. 2016. Heavy-duty vehicle platoon formation for fuel efficiency. *IEEE Transactions on Intelligent Transportation Systems* 17 (2016), 1051–1061.

[20] Steffen Linsenmayer, Dimos V. Dimarogonas, and Frank Allgöwer. 2019. Periodic event-triggered control for networked control systems based on non-monotonic Lyapunov functions. *Automatica* 106 (2019), 35–46.

[21] Fan Liu, Christos Masouros, Athina P. Petropulu, Hugh Griffiths, and Lajos Hanzo. 2020. Joint radar and communication design: Applications, state-of-the-art, and the road ahead. *IEEE Transactions on Communications* 68, 6 (2020), 3834–3862.

[22] Haixia Peng, Dazhou Li, Qiang Ye, Khadige Abboud, Hai Zhao, Weihua Zhuang, and Xuemin Shen. 2017. Resource allocation for cellular-based inter-vehicle communications in autonomous multiplatoons. *IEEE Transactions on Vehicular Technology* 66, 12 (2017), 11249–11263.

[23] Alexandre Seuret. 2012. A novel stability analysis of linear systems under asynchronous samplings. *Automatica* 48, 1 (2012), 177–182.

[24] Eduardo D. Sontag. 2008. Input to state stability: Basic concepts and results. In *Nonlinear and Optimal Control Theory*. Springer, 163–220.

[25] Nard Strijbosch, Geir E. Dullerud, Andrew R. Teel, and W. P. M. H. Heemels. 2021. L2-gain analysis of periodic event-triggered control and self-triggered control using lifting. *IEEE Trans. Automat. Control* 66, 8 (2021), 3749–3756.

[26] Jiankun Sun, Jun Yang, Shihua Li, and Zhigang Zeng. 2021. Predictor-based periodic event-triggered control for dual-rate networked control systems with disturbances. *IEEE Transactions on Cybernetics* (2021), 1–12.

[27] Dionysio Theodosis and Dimos V. Dimarogonas. 2019. Event-triggered control of nonlinear systems with updating threshold. *IEEE Control Systems Letters* 3, 3 (2019), 655–660.

[28] Johan Thunberg, Nikita Lyamin, Katrin Sjoberg, and Alexey Vinel. 2019. Vehicle-to-vehicle communications for platooning: Safety analysis. *IEEE Networking Letters* 1, 4 (2019), 168–172.

[29] Sebastian van de Hoef, Jonas Mårtensson, Dimos V. Dimarogonas, and Karl H. Johansson. 2020. A predictive framework for dynamic heavy-duty vehicle platoon coordination. *ACM Transactions on Cyber-Physical Systems* 4 (2020), 1–25.

[30] Pengfei Wang, Boya Di, Hongliang Zhang, Kaigui Bian, and Lingyang Song. 2019. Platoon cooperation in cellular V2X networks for 5G and beyond. *IEEE Transactions on Wireless Communications* 18, 8 (2019), 3919–3932.

[31] Wei Wang, Romain Postoyan, Dragan Nešić, and W. P. M. H. Heemels. 2020. Periodic event-triggered control for nonlinear networked control systems. *IEEE Trans. Automat. Control* 65, 2 (2020), 620–635.

[32] Myounggyu Won. 2021. L-platooning: A protocol for managing a long platoon with DSRC. *IEEE Transactions on Intelligent Transportation Systems* (2021), 1–14.

[33] Nan Wu and Harutoshi Ogai. 2013. Coordinative Platoon Running of SEV based on Vehicle-to-vehicle Communication. 2676–2681.

[34] Xinlei Yi, Kun Liu, Dimos V. Dimarogonas, and Karl H. Johansson. 2019. Dynamic event-triggered and self-triggered control for multi-agent systems. *IEEE Trans. Automat. Control* 64, 8 (2019), 3300–3307.

[35] Fan Yu and Subir Biswas. 2007. Self-configuring TDMA protocols for enhancing vehicle safety with DSRC based vehicle-to-vehicle communications. *IEEE Journal on Selected Areas in Communications* 25, 8 (2007), 1526–1537.

[36] Hao Yu, Tongwen Chen, and Fei Hao. 2020. Output-based periodic event-triggered control for nonlinear plants: An approximate-model method. *IEEE Transactions on Control of Network Systems* 7 (2020), 1342–1354.

[37] Di Zhao, Zidong Wang, Guoliang Wei, and Qing-Long Han. 2020. A dynamic event-triggered approach to observer-based PID security control subject to deception attacks. *Automatica* 120 (2020), 109–128.

Received 8 March 2022; revised 15 November 2022; accepted 13 August 2023

Event-Triggered Control with Intermittent Communications over Erasure Channels for Leader–Follower Problems with the Combined-Slip Effect

MOHAMMAD H. MAMDUHI, Automatic Control Laboratory, ETH Zürich, Switzerland
EHSAN HASHEMI, Mechanical Engineering Department, University of Alberta, Canada

In this article, we investigate the vehicle path-following problem for a vehicle-to-vehicle (V2V)–enabled leader–follower scenario and propose an integrated control policy for the following vehicle to accurately follow the leader's path. We propose a control strategy for the follower vehicle to maintain a velocity-dependent distance relative to the leader vehicle while stabilizing its longitudinal and lateral dynamics considering the combined-slip effect and tire force saturation. In light of reducing wireless communication errors and efficient usage of battery power and resources, we propose an intermittent V2V communication in which transmissions are scheduled based on an event-triggered law. An event is triggered and a transmission is scheduled in subsequent sample time if some of the well-defined path-following error functions (relative distance error and lateral error) exceed given tolerance bounds. Considering that the V2V communication channel might be erroneous or a transmission fails due to, e.g., vehicles' distance or low battery power, we consider data loss in the V2V channel. Our proposed control law consists of two components: a receding horizon feedback controller with state constraints based on a safe operation envelop and a feedforward controller that generates complementary control inputs when the leader's states are successfully communicated to the follower. To mitigate the effects of data loss on the follower's path-following performance, we design a remote estimator for the follower to predict the leader's state using its on-board sensor equipment when an event is triggered but the corresponding state information is not received by the follower due to a packet loss. Incorporating this estimator allows the follower to apply cautionary control inputs knowing that the path-following error had exceeded a tolerance bound. We show that while the feedback controller stabilizes the follower's dynamics, the feedforward component improves the safety margins and reduces the path-following errors even in the presence of data loss. High-fidelity simulations are performed using *CarSim* to validate the effectiveness of our proposed control architecture specifically in harsh maneuvers and high-slip scenarios on various road surface conditions.

CCS Concepts: • **Computer systems organization → Embedded and cyber-physical systems**; • **Computing methodologies → Model verification and validation;**

Additional Key Words and Phrases: Path-following control, V2V communication, model predictive control (MPC), feedforward control

This work is supported by the Natural Science and Engineering Research Council of Canada, Discovery Grant No. RGPIN-05097-2020.

Authors' addresses: M. H. Mamduhi (corresponding author), Automatic Control Laboratory, ETH Zürich, Physikstrasse 3, Zürich, Switzerland, 8092; e-mail: mmamduhi@ethz.ch; E. Hashemi, Mechanical Engineering Department, University of Alberta, Edmonton, Canada; e-mail: ehashemi@ualberta.ca.

2378-962X/2023/10-ART29 $15.00
https://doi.org/10.1145/3625562

ACM Reference format:
Mohammad H. Mamduhi and Ehsan Hashemi. 2023. Event-Triggered Control with Intermittent Communications over Erasure Channels for Leader–Follower Problems with the Combined-Slip Effect. *ACM Trans. Cyber-Phys. Syst.* 7, 4, Article 29 (October 2023), 25 pages.
https://doi.org/10.1145/3625562

1 INTRODUCTION

Connected Automated Driving Systems (ADSs) are revolutionizing mobility by exchanging information that could be used for safe decision-making, motion planning, perception enhancement, and even vehicle longitudinal/lateral stabilization in safety-critical conditions in intelligent transportation settings [4, 6]. Cooperative autonomous driving, as an example of connected advanced driver assistance systems (ADASs), is also benefiting from the resilience and robustness that connectivity brings in between different ADSs in platooning and path following [1, 41, 42] and, more specifically, cooperative control for urban driving [34]. Inter-vehicle communications through vehicle-to-vehicle (V2V) and the fast growing road-side infrastructures (vehicle-to-infrastructure [V2I]) are shown to play key roles in improving safe and energy-efficient ADSs [37, 43]. The concept of vehicular networks has created a great potential for individual vehicles to consider maneuvers of the surrounding vehicles cooperatively when planning their own maneuvers [1]. A classical yet challenging problem in vehicular networks is the path-following problem that aims for efficient/optimal motion planning, specifically for networks of designated vehicles, e.g., platoons [32, 47]. In this regard, by utilizing followers' on-board local sensory data (i.e., inertial measurement unit (IMU), visual, radar, LiDAR) or constant communication between agents, leader–follower robust control strategies have been developed in the literature in various contexts [21, 32]. In [45], a look-ahead control policy for heavy-duty vehicle platoons is proposed; a game-theoretic ADS path-planning algorithm is proposed in [15]; and a combined path-planning and motion control scheme is proposed in [26] using the hybrid automata model.

For communication of the inertial, heading, or steering data in cooperative ADSs, dedicated short-range communication (DSRC) [36] and other wireless technologies such as VANET [19] have been utilized. Nonetheless, such technologies suffer from network-induced issues, including delay, data loss, and channel reliability, primarily due to the inherent digital form of wireless communication with packet-based transmissions [3].

State-of-the-art on the leader–follower formation utilizes a variety of advanced control methods to ensure that the overall system maintains the desired formation and the follower can track the leader accurately. Traditional controllers such as Proportional Integral (PI) control has been used in [44] for smoothing traffic flow, whereas a neural network–based longitudinal control algorithm, which ensures uniform convergence of tracking errors, is presented in [33]. However, both of these works consider only longitudinal motion control, which significantly limits their practical implementation. A cooperative driving strategy for the connected vehicles by integrating vehicle velocity prediction, motion planning, and robust fuzzy path-following control is presented in [11] and the largely popular sliding-mode approach has been extensively used for control of leader–follower systems as found in [2, 21]. These works, however, consider continuous communication over error-free wireless channels, which is not always realizable in real-world scenarios. Control of dynamical systems over communication networks under network-induced errors such as delay, data loss, and bandwidth limitations are also studied in [9, 17, 20, 28, 35], mainly for the general networked control systems or mobile sensor networks.

In [12], analytical stability results are provided in the presence of system noise and ways of reducing the time and energy for attenuating perturbations are investigated, whereas a nonlinear

model predictive control (MPC) for combined lateral and longitudinal motions is presented in [31]. However, in all of these works, the combined-slip effect, which significantly affects the follower's stability and maneuverability, is not considered. To the best of our knowledge, the combined lateral and longitudinal path tracking for connected leader–follower ADSs in which the inter-vehicle communication is not continuous but event-triggered based on real-time tracking errors, and with an erasure channel as the transmission medium, is not yet addressed in the literature.

Our first motivation to study this problem is the need for an integrative and applicable safety-critical control policy for a granular model of connected ADSs such that, relying on inter-vehicle data exchange, safety envelopes and critical performance margins are improved and stability of the connected ADSs is guaranteed with less conservative constraints. Specifically for the leader–follower ADSs, this means that the designed controller should maintain the two-dimensional path-tracking error functions below some tolerable bounds, while guaranteeing stability of the interconnected ADSs where the follower's dynamics is externally affected by the leader's states, the road conditions, and the state of communication with the leader.

Our second motivation to tackle this problem is to argue that, from the control perspective on the one hand, a continuous inter-vehicle communication is not necessary while it can occur intermittently at times when data exchange leads to noticeable stability augmentation and tracking error reduction. On the other hand, from the communication perspective, reliability of wireless communication between mobile agents, especially with low-power short-range sending and receiving components, can be severely affected by, e.g., vehicles' distance, low battery power, temperature, line of sight, and cyber attacks [49]. Therefore, continuous inter-vehicle wireless communication is more exposed to deficiencies and interruptions. Hence, designing a control policy relying on continuous and perfect inter-vehicle communication is rather optimistic and less realistic.

In real-world application, communication between the leader and the followers is key towards smooth functioning of the overall multi-vehicle system and for formulating an effective control strategy. However, network-induced constraints are faced mainly due to the wireless nature of the communications and limited communication and battery-powered sending and receiving components. A well-studied and proven way in which the drawbacks of limited communication power can be countered is employing the event-triggered scheme [13, 25]. Intermittent signals are an inherent feature of the digital implementation of an event-triggered controller designed for any system. Only when a predefined triggering condition is reached, the control signal will be updated, in turn establishing a better trade-off between the communication load and the desired system performance [40]. The aperiodic event-triggered scheme achieves the aforementioned trade-off in such a manner that the requirement of continuous monitoring for neighbors' state can also be avoided. To facilitate the triggering of V2V transmissions, an event has been designed in this work that considers the follower's suitably designed path following error functions. Error dependency in designing the event-triggered functions is shown to be capable of significantly reducing the communication load while maintaining the required performance level [38, 39]. Furthermore, with the ability of state constraints and system response prediction over the horizon, MPC has been recently used for controlling and stabilizing vehicular leader–follower dynamics [29, 30, 48]. The design of the proposed feedback component of the controller using an event-triggered strategy for the MPC would also increase its usefulness in addition to decreasing the use of communication (and power) resources, since it reduces the frequency of solving predictive optimization problems [31]. In the state-of-the-art, the reported developments of solving MPC under event-triggered control has been discussed, e.g., in [8]. However, event-triggered communication-based MPC has not been fully studied for the leader–follower problem, particularly for the situation in which there is an energy-efficient transmission-scheduling scheme that is performance sensitive.

The primary contributions of this article are summarized as follows:

(1) An event-triggered V2V communication scheme is proposed that is resource efficient, less exposed to induced errors, and safety sensitive (function of the desired safety and vehicle stability margins). V2V communications are established through non-ideal wireless channels, i.e., inter-vehicle transmissions are subject to data loss where the loss probability may depend on parameters such as vehicles' distance, battery power, and outside temperature.
(2) An integrative control policy is proposed that consists of two components: a dominant MPC feedback controller by considering combined-slip tire forces and tire capacities where the MPC controller is modified for the intermittent state update, and an event-triggered V2V-enabled feedforward controller.
(3) A remote state estimation scheme is proposed to compensate for data loss that can be implemented at the follower to reduce the tracking error by estimating the states of the leader when an event is triggered but the corresponding data is lost.

In this work, we use a sufficiently accurate nonlinear vehicle dynamics [16] and a high-fidelity combined-slip tire model [24], which are described in the next section.

The rest of this article is organized as follows. Section 2 describes the model and introduces the problem statement. The MPC-based feedback control, V2V-enabled feedforward control, and remote state estimator are proposed in Section 3, along with a comprehensive performance analysis. Simulation results are presented in Section 4, and concluding remarks are summarized in Section 5.

2 PROBLEM STATEMENT AND PRELIMINARIES

We consider two identical vehicles in which the preceding one (leader) is controlled by a human or an exogenous input that generates the desired steering and acceleration/deceleration, i.e., the leader's path. With this, our approach is applicable to both the fully automated and human-driven lead vehicle. Dynamics of the leader is not affected by the follower, which aims to accurately track the leader's path, maintaining a desired velocity-dependent relative distance while ensuring its lateral stability. Inter-vehicle communications are in the form of unilateral V2V, i.e., leader's steering angle and longitudinal/lateral accelerations (in the body frame) can be exchanged with the follower. Depending on availability of the leader's state information, the follower computes its control inputs (steering angle and acceleration/deceleration) such that: (i) the desired relative distance is maintained, (ii) the path tracking error is bounded, and (iii) the lateral vehicle stability is guaranteed.

2.1 Combined-Slip Prediction Model

We employ the nonlinear single-track model for vehicle longitudinal and lateral dynamics and the average lumped LuGre model [46] for tire forces. In the average lumped LuGre model, the tire internal state $\bar{z}_p$, for each longitudinal/lateral direction $p \in \{x, y\}$, relates to the relative speeds as

$$v_{r,x} = R\omega - v_x, \tag{1}$$

$$v_{r,y} = v_x\alpha, \tag{2}$$

$$\dot{\bar{z}}_p = v_{r,p} - \bar{C}_p\bar{z}_p - k_pR|\omega|\bar{z}_p. \tag{3}$$

Table 1 summarizes the variables and parameters used to describe the model and the proposed control policy in this article. The tire slip angle, vehicle speed, tire rotational speed, and effective rolling radius are denoted by α, v_x, ω, and R, respectively. The transient tire force parameter $\bar{C}_p$ is related to the longitudinal (and lateral) rubber stiffness $\sigma_{0,x}$ (and $\sigma_{0,y}$), relative speeds $\mathbf{v}_r = [v_{r,x} \quad v_{r,y}]^\top$, and the road friction $0.1 \leq \theta \leq 1$, which ranges for icy to dry conditions. Then, the

Table 1. Parameter Descriptions

Parameter	Description	Parameter	Description
$\bar{z}_p$	Tire internal state	e_p, e_y	Relative distance error and lateral error
$v_{r,x},\ v_{r,y}$	Relative speeds in the tire frame $\{t\}$	$e_\phi(t)$	Yaw angle error
α	Tire slip angle	d_r	Velocity-dependent relative distances
ω	Tire rotational speed	h	Headway time
R	Effective rolling radius	d_0	Minimum safe relative distance
$\bar{C}_p$	Transient tire force parameter	κ_f	Path curvature
$\sigma_{0,1,2}$	Stiffness/damping tire parameters $\in \mathbb{R}^{2\times2}$	$u = [a_x^d\ \delta_d]^\top$	Input
θ	Road friction	$\psi(k) \in \{0,1\}$	Event trigger indicator
$g^\pm$	Combined-slip transition function for acceleration/brake	T_s	Sampling time
F_x^j, F_y^j	Longitudinal/lateral tire forces at corner j	$\gamma_c(k) \in \{0,1\}$	Dropout indicator
α^j, λ^j	Lateral and longitudinal slip	$P_{d,^\circ C}(k)$	Dropout probability
v_x, v_y	Vehicle longitudinal/lateral speed in the body frame $\{b\}$	$\zeta(k)$	V2V communication indicator
ϕ, r_z	Yaw angle and yaw rate about z axis in $\{b\}$	μ_r	Rear tires' friction capacity
δ_l, δ	Leader's and follower's front steering angle	g	Gravitational acceleration
l_f, l_r	Front/rear axle distance to vehicle CG	N_h	Prediction horizon
m, I_z	Vehicle mass and moment of inertia	$a^d_{x,FF}$	Desired feedforward acceleration input
a_x^d	Desired acceleration	$\hat{a}_{x,l}$	Estimated acceleration of leader
τ_a	Time constant for longitudinal dynamics (actuation)	$n_a \sim \exp(\bar{\lambda})$	Exponentially distributed random noise
$\alpha_f, \alpha_r, \alpha_b$	Front/rear tire's slip angle and saturation slip angle	ϕ_s	Heading angle in navigation frame $\{w\}$
β	Follower's side-slip angle in navigation frame $\{w\}$	$\Gamma_{x,y}$	Combined-slip variable

normalized longitudinal/lateral tire forces yield

$$\mathbf{f}_n = \sigma_0 \bar{\mathbf{z}} + \sigma_1 \dot{\bar{\mathbf{z}}} + \sigma_2 \mathbf{v}_r, \tag{4}$$

in which, $\mathbf{f}_n = [f_{n,x},\ f_{n,y}]^\top$, $\mathbf{v}_r, \bar{\mathbf{z}} \in \mathbb{R}^2$ can be described in the combined or pure-slip force descriptions, and $\sigma_{0,1,2} \in \mathbb{R}^{2\times2}$ denotes the rubber stiffness and damping matrices. Finally, the combined-slip forces of the steady-state LuGre model, for each (longitudinal or lateral) direction $p \in \{x, y\}$, can be written as

$$F_p = \left(\frac{\theta g(\mathbf{v}_r)}{\Gamma_p + \tau_p \theta g(\mathbf{v}_r)} + \sigma_{2,p} \right) v_{r,p} F_z, \tag{5}$$

where $\tau_p = \frac{k_p R\omega}{\sigma_{0,p}}$. We define the combined-slip variables as $\Gamma_x := \sqrt{(\eta^{-1} v_x \alpha)^2 + (\lambda R\omega)^2}$, $\Gamma_y := \sqrt{(v_x\alpha)^2 + (\eta\lambda R\omega)^2}$, where η is a (combined-slip) tire parameter and λ is the longitudinal slip ratio at each tire. The transition function $g = g(\mathbf{v}_r)$ is approximated [24] by g^+, g^- for acceleration and brake scenarios, respectively, as

$$g^+ = c - c_\lambda \lambda - c_\alpha [1-\lambda]\alpha, \qquad g^- = c + c_\lambda \lambda + c_\alpha [1-\lambda]\alpha. \tag{6}$$

As a result, longitudinal and lateral forces F_x^j, F_y^j at each tire $j \in \{fl, fr, rl, rr\}$ (for the front-left, front-right, rear-left, and rear-right corners) can be expressed as the lateral slip α^j and longitudinal slip λ^j. It is to be noted that

$$\lambda^j = \begin{cases} \frac{v_{r,x}^j}{R\omega^j}, & \text{for acceleration scenario,} \\ \frac{v_{r,x}^j}{v_x}, & \text{for brake case.} \end{cases}$$

The tire slip angles on the front and rear axles (shown in Figure 1) are defined according to the slip angle and combined-slip ratio for the front and rear axles as follows:

$$\alpha^j = \delta^j - \frac{v_y + l_j \bar{\gamma}^j r_z}{v_x},$$

where $\delta^{rl} = \delta^{rr} = 0, \bar{\gamma}^{fl,fr} = -\bar{\gamma}^{rl,rr} = 1, l^{fl} = l^{fr} = l^f$, and $l^{rl} = l^{rr} = l^r$. The decreasing effect of longitudinal slip on the normalized lateral tire forces (shown in Figure 1) is considered in

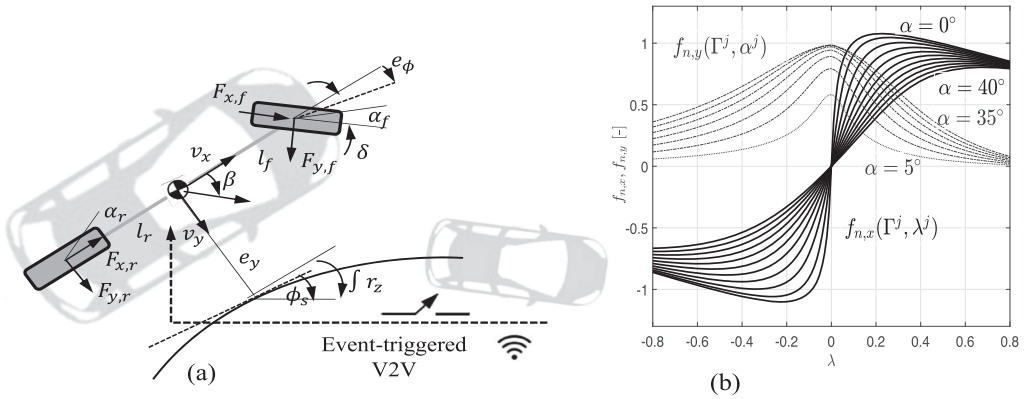


Fig. 1. (a) Vehicle frames, lateral states and error states; (b) combined-slip normalized tire forces $f_{n,p} := F_p/F_z$ for the receding horizon control.

the developed combined-slip model. The proposed combined-slip model (with the approximated transition function $g^{\pm}$) is close to real behavior of the tire, especially in high-slip cases caused by stabilization during path following, which is one of our main contributions.

The leader is controlled by exogenous inputs, therefore, we propose the error-sensitive path tracking and stabilizing control architecture for the follower. The following vehicle longitudinal/lateral kinetics with combined-slip tire forces thus refers to the follower, unless clearly stated otherwise. We then write the dynamics of the vehicles as

$$m(\dot{v}_x - r_z v_y) = F_{x,f}\cos\delta + F_{x,r} - F_{y,f}\sin\delta, \tag{7}$$

$$m(\dot{v}_y - r_z v_x) = F_{x,f}\sin\delta + F_{y,r} + F_{y,f}\cos\delta, \tag{8}$$

$$I_{zz}\dot{r_z} = (F_{x,f}\sin\delta + F_{y,f}\cos\delta)l_f - F_{y,r}l_r, \tag{9}$$

where, the longitudinal tire forces on the front and rear axles are

$$F_{x,f} = F_x^{fl} + F_x^{fr}, \qquad F_{x,r} = F_x^{rl} + F_x^{rr},$$

and the lateral axle forces are

$$F_{y,f} = F_y^{fl} + F_y^{fr}, \qquad F_{y,r} = F_y^{rl} + F_y^{rr},$$

where the longitudinal and lateral speeds in the body frame $\{b\}$ attached to the follower's center of mass are v_x, v_y; the yaw rate is r_z; the front tires' steering angle is denoted by δ; the front and rear axle distance to vehicle center of mass are l_f, l_r; and the follower's mass and moment of inertia are m and I_{zz}. The complex pitch/roll dynamics are not explicitly included in the model description, but their effect is inherently considered in the vertical forces F_z^j estimated by a linear observer proposed in [27]. The bandwidth of the lower-level longitudinal controls (due to brake, electric motor, or engine actuation systems) is also considered in the predictive model using a first-order dynamics as

$$\dot{a}_x = -\frac{1}{\tau_a}a_x + \frac{1}{\tau_a}a_x^d, \tag{10}$$

where a_x^d is the desired acceleration generated by the longitudinal control unit and τ_a is the actuator's time constant. The constraints on control inputs are $a_x^{\min} \le a_x^d(t) \le a_x^{\max}$ for the desired acceleration, and $\delta_l \le \delta_d(t) \le \delta_u$ for the desired front steering. Moreover, we impose state constraints as $0 \le v_x(t) \le v_x^{\max}, \forall t \ge 0$ for the longitudinal speed, and $a_x^{\min} \le a_x(t) \le a_x^{\max}$ for longitudinal acceleration, which is bounded based on the road friction condition.

2.2 Error Functions and Dynamics

To measure the path tracking accuracy, we define four time-varying error functions: the relative distance error $e_p(t)$; the relative velocity error $e_v(t)$; the lateral error $e_y(t)$; and the yaw error $e_\phi(t)$. The relative distance error is defined by $e_p(t) = d(t) - d_r(t)$, where $d(t)$ is the actual relative distance between the vehicles' center of masses at time t, and $d_r(t) = v_x(t)h + d_0$ is the desired velocity-dependent relative distance at time t with the constant parameters h and d_0 as the headway time and the minimum safe relative distance (i.e., at standstill), respectively. The longitudinal relative distance error is subject to the constraint $-d_r(t) \leq e_p(t) \leq e_p^{\max}$ to avoid a rear-end collision, where e_p^{max} is a desired maximum threshold for the relative distance error.

The time derivative of the relative distance error, for which we drop the time indicator t for simplicity, yields $\dot{e}_p = \dot{d} - \dot{v}_x h$. Then, substituting the inter-vehicle relative velocity error e_v defined by $e_v := \dot{d} - r_z v_y h$ (recall that the distance between the two vehicles in our scenario is a two-dimensional curve and involves rotational motion) into $\dot{e}_p$, we obtain that

$$\dot{e}_p = e_v - (\dot{v}_x - r_z v_y)h. \tag{11}$$

The inter-vehicle relative velocity error has the following dynamics by calculating the relative accelerations between two vehicles as

$$\dot{e}_v = a_{x,l} - \dot{v}_x + r_z v_y. \tag{12}$$

The lateral error e_y is defined as the distance between the follower's center of mass and the path of the leader at the point orthogonal to the tangent of the leader's path (shown in Figure 1). The dynamics of e_y can be derived as

$$\dot{e}_y = v_y + v_x e_\phi, \tag{13}$$

where the yaw error $e_\phi = \phi - \phi_s$ is defined as the angular difference between the tangent line of the leader's path and the follower's orientation (see Figure 1). Then, the dynamics of the yaw error can be derived as

$$\dot{e}_\phi = r_z - \kappa_f v_x. \tag{14}$$

Note that the path curvature (the inverse of road radius) is defined by $\kappa_f := \frac{\partial \phi}{\partial s} = \frac{d\phi_s}{dt} / \frac{ds}{dt} \approx \frac{\dot{\phi}_s}{v_x}$ along the path; thus, $\dot{\phi}_s \approx \kappa_f v_x$. Clearly, for a straight road (i.e., only longitudinal motion is considered) $\kappa_f \to 0$, and yaw error derivative would reduce to the angular velocity of the follower, as expected.

The goal is to design a control policy for the follower to guarantee stable dynamics and to keep the mentioned four error functions within their tolerable bounds, where the control unit has intermittent access to the leader's state information through V2V communications.

By substituting the combined-slip tire forces in Equations (7) to (9) and using the error function definitions, the resulting nonlinear prediction model can be written as

$$\dot{x}(t) = f(x(t), u(t), \mathbf{w}(t)), \tag{15}$$

with the augmented state variable $x = [a_x \; v_x \; v_y \; r_z \; e_p \; e_v \; e_\phi \; e_y]^\top$ (assumed to be measurable), the input $u = [a_x^d \; \delta_d]^\top$, and the uncontrolled input $\mathbf{w} = [a_{x,l} \; \delta_l]^\top$.

Remark 1. The follower is not equipped with a GNSS, hence, does not have any information about the leader's and its own positions in the global coordination frame $\{W\}$. However, it is able to determine the closest reference point based on the follower's IMU and local visual data (via monocular or stereo camera). The follower is equipped with an on-board camera (or radar) that can measure (with some uncertainty) the inter-vehicle distance, relative velocity, and the path curvature. The relative distance error, e_p, can thus be measured with ease, which only requires

information about the actual relative distance between the vehicles' center of masses and the desired velocity-dependent relative distance. Also, the lateral error, e_y, can be measured using the distance between the follower's center of mass and the path of the leader at the point orthogonal to the longitudinal direction of the follower. This also highlights the fact that the proposed structure utilizes only commonly available and cost-effective sensors for the required measurements.

2.3 Intermittent V2V Communication Model

In this article, the communication between the leader and follower is intermittent and unilateral, i.e., state information of the leader is partially shared with the follower, while the leader's dynamics is not affected by states of the follower. Communication instances are determined by the follower based on a real-time event-triggered law that generates triggering events whenever the path-tracking error functions e_p and e_y exceed their tolerable bounds. Since communication networks operate digitally, we design the event-triggered law in the discrete time domain, and, therefore, use the discrete versions of the error functions e_p and e_y. Discrete models can be obtained by applying the forward Euler method on the continuous model with a sufficiently small sampling time $T_s \in \mathbb{R}_{>0}$ such that a discrete time-step $k \in \mathbb{N}_0$ represents the time instance $T_s k \in \mathbb{R}_{\geq 0}$.

We define $\psi(k) \in \{0, 1\}$ as the event trigger indicator, i.e., $\psi(k) = 1$ indicates that the follower needs a state update from the leader (steering angle and acceleration). Reception of the leader's state information is expected at the subsequent time-step $k + 1$ through the dedicated V2V channel. The triggering instances are decided by the follower according to the event-triggered law introduced below:

$$\psi(k) = \begin{cases} 1, & \text{if } |e_p(k)| > e_p^{\max} \text{ or } |e_y(k)| > e_y^{\max}, \\ 0, & \text{otherwise}, \end{cases} \tag{16}$$

where $e_p^{\max}$ and $e_y^{\max}$ are the maximum tolerable bounds for the relative distance error and lateral error, respectively. We assume constant tolerable bounds; however, the results of this article extend to time-varying bounds $e_p^{\max}(k)$ and $e_y^{\max}(k)$ given that they are known at all times.

Remark 2. The yaw error e_ϕ is a local error measure of the follower independent of the leader's dynamics and, therefore, awareness of the leader's states does not help reduce it (represented in (14)). In fact, the local MPC controller of the follower ensures that the yaw error remains bounded utilizing the state constraints (vehicle safe operating envelope) in the optimization program (25). Hence, without loss of generality, we exclude e_ϕ and e_v from the event-triggered law in (16).

According to (16), if one of the error functions $e_p(k)$ or $e_y(k)$ exceeds its tolerable bound at a time-step k, a V2V communication is requested by the follower and transmission will be arranged by the leader for the immediate next sample time $k + 1$. Since communication is unilateral, and moreover, the sequence of transmission instances cannot be computed *a priori*, the leader is informed immediately when the follower is in need of a state update. This can be implemented by a Request-To-Send (RTS) signal from the follower to the leader. For vehicular networks, a common protocol for the physical and medium access control (MAC) layers that is customized for dedicated short-range wireless communication, and is particularly used for vehicular communication systems, is the IEEE 802.11p, from the entrenched family of IEEE 802.11 standards [10]. Dedicated Short Range Communication (DSRC) is a common and practical low-latency and reliable wireless communication technology for V2V communication without requiring access to the cellular network. Within this protocol, sending RTS frames is possible. For the ease of technicality, we assume that a transmitted RTS signal will always reach leader within the same sample time that it is transmitted. This assumption is not restrictive, as RTS signals are light (<50 bytes) and the delivery time is in the range of microseconds [22], which is negligible compared with the typical sampling times

for vehicular applications (∼ 20–50 milliseconds). In terms of latency, DSRC technology for V2V communication induces very low latency of magnitude, ∼ 1–5 milliseconds. This is still well within the mentioned vehicle control sample time of 20–50 milliseconds. therefore, we can conveniently assume that a V2V communication will be completed within one control sample time.

Transmissions from the leader to the follower are prone to dropout due to the external effects on the quality of wireless channels. We define the dropout indicator $\gamma_c(k) \in \{0, 1\}$, as

$$\gamma_c(k) = \begin{cases} 1, & \text{data packet is successfully delivered at time-step } k, \\ 0, & \text{data packet is dropped.} \end{cases} \tag{17}$$

Packet dropouts are commonly modeled by a probabilistic i.i.d. process as it often depends on statistically uncorrelated parameters such as channel traffic, erasure, distance of the sending and receiving nodes, signal direction, and even temperature. In this article, the V2V channel is dedicated between the vehicles; hence, there is no external data traffic that causes collisions. We take the scenario that the probability of packet loss at a time k (given that a transmission is triggered by the follower at time $k-1$) depends on the actual relative distance between the vehicles $d(k)$ and the outside temperature $^\circ C(k)$. These two parameters in reality can be considered statistically uncorrelated. Intuitively, greater relative distances or lower temperatures lead to higher packet loss probability. Analytical derivation of a multi-variable distribution of this sort for the packet loss probability is in practice challenging. However, such a distribution can be empirically modeled by fitting a distribution to a large enough dataset [7]. Detailed discussions of distribution derivation is out of the scope of this article; here, we assume that a probability distribution exists from which i.i.d. dropout probabilities $P_{d,^\circ C}(k) \in [0, 1]$ will be realized at any time k given every pair of $\{d(k),^\circ C(k)\}$, i.e.,

$$\mathsf{P}[\gamma_c(k) = 1 | \psi(k-1) = 1] = 1 - P_{d,^\circ C}(k), \tag{18}$$

$$\mathsf{P}[\gamma_c(k) = 0 | \psi(k-1) = 1] = P_{d,^\circ C}(k). \tag{19}$$

Note that if a transmission is not requested, i.e., $\psi(k-1) = 0$, possibility of a dropout is irrelevant. Finally, if a transmission is scheduled for time-step k but the corresponding data packet is not received by the follower at time k, the follower can assume that a packet dropout has occurred.

In summary, the considered problem can be seen as a tracking problem in which the combined lateral and longitudinal tracking is to be addressed for a connected leader–follower ADS, wherein the inter-vehicle communications are not continuous and an erasure channel as the transmission medium is considered. The steering and acceleration/deceleration of the leader is controlled by a human or autonomously, whereas the follower has no bearing on the leader's dynamics. The objective of the follower is to maintain the desired relative distance, which relies on the velocity while precisely tracking the leader's course. Additionally, the follower should guarantee lateral stability. Unilateral V2V communication enables sharing of the leader's steering angle and longitudinal/lateral accelerations with the follower during inter-vehicle communications.

3 PREDICTIVE FEEDBACK AND V2V-ENABLED FEEDFORWARD CONTROL DESIGN

In this section, we propose our control architecture for the described leader–follower scenario with intermittent V2V communication governed by the event-triggered law (16), over an erasure channel. The proposed control policy for the follower includes: *(i)* an MPC feedback controller; and *(ii)* a V2V-enabled feedforward controller, described in discrete time as follows:

$$\bar{u}(k) = g_k(u(k), w(k-1)) = u_{MPC}(k) + \zeta(k)\, u_{FF}(k), \tag{20}$$

where u_{MPC} is the dominant predictive-based feedback controller and u_{FF} denotes the feedforward control component. The binary variable $\zeta(k) = \gamma_c(k)\psi(k-1)$ ensures that u_{FF} is in the control

loop only when the leader's states are successfully communicated with the follower. Clearly, the control law $\bar{u}(k)$ in (20) will be reduced to only the MPC component ($\zeta(k) = 0$) when either a communication with the leader is not triggered, i.e., $\psi(k-1) = 0$, or it is triggered but dropped, i.e., $\gamma_c(k) = 0$.

As mentioned earlier, the control inputs for the follower are the desired acceleration and steering angle, i.e., $u = [a_x^d \ \delta_d]^\top$. With the mentioned variables as the *outputs* of the control unit and the vector $\mathbf{w} = [a_{x,l} \ \delta_l]^\top$ as the *inputs* to the control unit, the controller model for the follower can be expressed in linear time-varying form in discrete time as follows:

$$u_{MPC}(k) = \bar{A}(k)u(k) + \bar{B}(k)\mathbf{w}(k-1) + \bar{\omega}(k-1), \tag{21}$$

$$u_{FF}(k) = \tilde{A}(k)u(k) + \tilde{B}(k)\mathbf{w}(k-1) + \tilde{\omega}(k-1), \tag{22}$$

where $\bar{\omega}(k-1)$ represents the uncertainty associated with the controller's input $\mathbf{w}(k-1)$ estimated by the follower and $\tilde{\omega}(k-1)$ is the uncertainty associated with $\mathbf{w}(k-1)$ when it is communicated with the follower. The time-varying 2×2 matrices $\bar{A}(k), \bar{B}(k), \tilde{A}(k)$ and $\tilde{B}(k)$ are computed based on the nonlinear model of the vehicle and the tire forces as well as the designed controller gains for the predictive controller and the feedforward controller, as we will discuss later.

Remark 3. It should be noted that the predictive controller only uses the leader's acceleration as the external input to generate the output $[a_x^d \ \delta_d]^\top$ and does not use the leader's steering state δ_l. This means that the matrix $\bar{B}(k)$ is of the form $\begin{bmatrix} * & 0 \\ * & 0 \end{bmatrix}, \forall k$. The predictive controller is always in the loop, as mentioned in (20), while the feedforward controller is engaged only when a V2V communication is established. Therefore, the feedforward controller has, at time k, access to the communicated $\mathbf{w}(k-1)$, while the predictive controller needs to measure/estimate the leader's acceleration (using radar or on-board camera) when this data is not communicated. It is a valid assumption that communicated information is more precise/less uncertain compared with its remotely measured or estimated versions. Therefore, we can realistically assume that the uncertainty term $\bar{\omega}(k)$ is greater than or equal to $\tilde{\omega}(k), \ \forall k$, where being "greater than or equal to" applies to the corresponding parameter(s) of the uncertainly type, e.g., if it is stochastic, then it applies to the mean, variance, or higher moments; or if it is bounded uncertainty, it will be applied on the upper/lower bounds of the uncertainty set.

From (20) to (22), we can express the control law for the follower in aggregate form as

$$\bar{u}(k) = (\bar{A}(k) + \zeta(k)\tilde{A}(k))u(k) + (\bar{B}(k) + \zeta(k)\tilde{B}(k))\mathbf{w}(k-1) + \bar{\omega}(k-1) + \zeta(k)\tilde{\omega}(k-1). \tag{23}$$

In the case of a triggering followed by a dropout, we propose a remote estimation of the leader's acceleration state at the follower to reduce the longitudinal error. The remote estimate, which is less accurate than if it was transmitted through the V2V channel, to some extent compensates the effects of packet loss on the path-following performance. We will discuss the design of this estimator in Section 3.4.

3.1 Predictive-Based Feedback Control

The prediction model is linearized around the operating point $(x(k), u(k-1), \mathbf{w}(k-1))$, at each time-step k in discrete-time. The linear affine prediction model can be written as

$$x(k+1) = A(k)x(k) + B_\zeta(k)u(k) + \tilde{B}_\zeta(k)\mathbf{w}(k-1) + \bar{\Omega}(k), \tag{24}$$

where $B_\zeta(k) = \bar{A}(k) + \zeta(k)\tilde{A}(k)$, $\tilde{B}_\zeta(k) = \bar{B}(k) + \zeta(k)\tilde{B}(k)$, and $\bar{\Omega}(k) = \Omega(k) + \bar{\omega}(k-1) + \zeta(k)\tilde{\omega}(k-1)$, in which the discrete-time realization is approximated by the step-invariance method (expressed by the Peano-Baker series); the linearized realization is assumed to not vary a lot in each interval

$[t_k, t_{k+1}]$, which is valid for the proposed integrated following-stabilization control model with the sample time $T_s = 20ms$.

The safety and performance constraints on the predictive model and the consequent optimization program are as follows: stable sideslip $\beta(t) := \frac{v_y(t)}{v_x(t)}$ and yaw rate state trajectories are restricted to an envelope (parallelogram) with side boundaries $|\beta(t) - \frac{r_z(t)l_r}{v_x(t)}| \leq \tan\alpha_b$, in which α_b is the saturation slip angle for the (lateral) tire force experimental data in *CarSim*. Assuming that v_x is constant over the prediction horizon N_h, the constraint on the sideslip angle $\beta(t)$ can be rewritten as a linear combination of v_y, r_z. The symmetric horizontal safe boundaries in $|r_z(t)| \leq r_{z,b} := \frac{\mu_r g}{v_x(t)}, \forall t \geq 0$, are denoted by $r_{z,b}$, where μ_r is the rear tires' friction capacity affected by longitudinal slip ratios [5] and g is the gravitational acceleration. We repeat here the control input constraints $a_x^{\min} \leq a_x^d(t) \leq a_x^{\max}$, and $\delta_l \leq \delta_d(t) \leq \delta_u$, as well as the state constraints $-d_r(t) \leq e_p(t) \leq e_p^{\max}$ to avoid a rear-end collision, $0 \leq v_x(t) \leq v_x^{\max}, \forall t \geq 0$ for the longitudinal speed, and $a_x^{\min} \leq a_x(t) \leq a_x^{\max}$ for longitudinal acceleration, which is bounded based on the road friction condition.

3.2 Optimization Program

The control objective is to stabilize the follower by keeping the sideslip angle β and yaw rate within the safety envelope and maintain a safe inter-vehicular distance while minimizing the attitude and lateral position errors (i.e., e_v and e_ϕ will be controlled by the predictive controller). By employing state and constraints, the following optimization problem is solved in the receding horizon fashion:

$$\dot{u} = \underset{u(k), s_1, s_2}{\arg\min} \quad \mathcal{J} \tag{25}$$

subject to

$$x_{i+1|k} = A(k)x_{i|k} + B_\zeta(k)u_{i|k} + \tilde{B}_\zeta(k)\mathbf{w}_{i|k-1} + \bar{\Omega}(k) \tag{26a}$$

$$x_{0|k} = x(k),\ \mathbf{w}_{0|k} = \mathbf{w}(k) \tag{26b}$$

$$u^{\min} \leq u_{k+i|k} \leq u^{\max},\quad i \in \{0, 1, \ldots, N_h - 1\} \tag{26c}$$

$$0 \leq v_{x,k+i|k} \leq v_x^{\max},\quad a_x^{\min} \leq a_{x,k+i|k} \leq a_x^{\max} \tag{26d}$$

$$|r_{z,k+i|k}| \leq \frac{\mu_r g}{v_x(k)} + s_1 \tag{26e}$$

$$|\beta_{k+i|k} - \frac{r_{z,k+i|k}l_r}{v_x(k)}| \leq \tan\alpha_b + s_2 \tag{26f}$$

$$x_{i|k} = x_{k+i|k},\quad u_{i|k} = u_{k+i|k} \tag{26g}$$

where (26a) describes the constraint imposed due to the system dynamics, (26b) describes the initial condition constraints, the input saturation is described through the constraints in (26c), the constraints on velocity and acceleration are represented using (26d), the upper and lower bounds of the yaw rate are constrained through (26e), and the yaw rate state trajectories are restricted by the constraints given in (26f), whereas the cost function $\mathcal{J}$ for the prediction horizon is given as

$$\begin{aligned}\mathcal{J} = &\sum_{i=0}^{N_h-1}\left[x_{i|k}^\top Q x_{i|k} + u_{i|k}^\top R_1 u_{i|k} + \tilde{u}_{i|k}^\top R_2 \tilde{u}_{i|k}\right] \\ &+ x_{N_h|k}^\top Q x_{N_h|k} + l_1^s s_1 + l_2^s s_2,\end{aligned} \tag{27}$$

where $Q \succeq 0$, $R_1, R_2 \succ 0$ are state, input, and proximity weighting matrices with compatible sizes, and the proximity term is

$$\tilde{u}_{i|k} = u_{i|k} - u_{i|k-1}.$$

The side-slip for the state constraint is also defined as

$$\beta_{k+i|k} := \frac{v_{y,k+i|k}}{v_x}.$$

The slack variables, $s_1, s_2 \geq 0$, with weights $l_1^s, l_2^s \in \mathbb{R}_{>0}$, are introduced to resolve feasibility issues in state inequality constraints. The vector of optimized control inputs is

$$\dot{u}(k) = [u_{0|k}^\top \; u_{1|k}^\top \; \dots \; u_{N_h-1|k}^\top]^\top.$$

The stable-operation envelope constraints of the follower (on $r_z(k)$ and $v_y(k)$) change over the prediction horizon, while we consider $v_{x,k+i|k} = v_x(k), i \in \{0, 1, \dots, N_h\}$ to have the constraints linear with regard to the side-slip and yaw rate states. It should be noted that the discrete-time system (24) satisfies the finite time controllability condition in the sense that given a continuous non-decreasing function $G(\epsilon, k)$ in the class $\mathcal{KL}_0$ (i.e., for each $\epsilon > 0$, $\lim_{k\to\infty} G(\epsilon, k) = 0$, and for $k \geq 0$, $G(\epsilon, k) \in \mathcal{K}_\infty$) for each $x(0) \in \mathcal{X}$, there exists a control function $u_{x_0} \in \mathcal{U}$ satisfying [18]

$$\mathcal{J}(x(k, u_{x_0}), u_{x_0}(k)) \leq G(\mathcal{J}^*(x(0)), k). \tag{28}$$

The cost function and associated state constraints are transformed into input constraints for computational efficiency and real-time applications, where the conventional qpOASES [14] Quadratic Programming solver is used. This has been done by expanding the state vector over the prediction horizon as

$$x^h(k) = \mathcal{F}_x x(k) + \mathcal{F}_u \dot{u}(k) + \mathcal{F}_{\mathbf{w}} \mathbf{w}^h(k) + \mathcal{F}_\Omega,$$

where $x^h(k) = [x_{1|k}^\top \; x_{2|k}^\top \; \dots \; x_{N_h|k}^\top]^\top$, $\dot{u}(k)$ is the optimized input vector, $\mathcal{F}_x, \mathcal{F}_u, \mathcal{F}_{\mathbf{w}}$ are conventional evolution matrices for the predictive model (24), and $\mathbf{w}^h(k) = [\mathbf{w}_{0|k}^\top \; \mathbf{w}_{1|k}^\top \; \dots \; \mathbf{w}_{N_h-1||k}^\top]^\top$. The cost function in (27) can be written as $\mathcal{J} = \frac{1}{2}\dot{u}(k)^\top \mathbf{D}\dot{u}(k) + \dot{u}(k)^\top \mathbf{E} + l_1^s s_1 + l_2^s s_2 + \text{Const.}$, whereas

$$\begin{aligned} \mathbf{D} &= 2(\mathbf{R}_1 + \mathcal{F}_u^\top \mathbf{Q} \mathcal{F}_u + \mathbf{R}_2), \\ \mathbf{E} &= 2\mathcal{F}_u^\top \mathbf{Q}(\mathcal{F}_x x(k) + \mathcal{F}_{\mathbf{w}} \mathbf{w}^h(k) + \mathcal{F}_\Omega) - 2\mathbf{R}_2 \dot{u}(k-1), \end{aligned} \tag{29}$$

in which $\mathbf{R}_1$, $\mathbf{R}_2$, and $\mathbf{Q}$ are block-diagonal matrices formed by the matrices R_1, R_2, and Q, respectively. The cost function is then rearranged as a standard quadratic programming as

$$\mathcal{J} = \frac{1}{2}\bar{\dot{u}}(k)^\top \begin{bmatrix} \mathbf{D} & \mathbf{0}_{2N_h \times 2} \\ \mathbf{0}_{2\times 2N_h} & l \end{bmatrix} \bar{\dot{u}}(k) + \bar{\dot{u}}(k)^\top \begin{bmatrix} \mathbf{E} \\ \mathbf{0}_{2\times 1} \end{bmatrix}, \tag{30}$$

whereas $l = \text{diag}(l_1^s, \, l_2^s)$ and $\bar{\dot{u}}(k) = [\dot{u}(k)^\top, \; s_1, \; s_2]^\top$ is the optimal sequence of controlled inputs and the slack vector.

3.3 V2V-Enabled Feedforward Control

The follower receives the steering angle and acceleration states of the leader at the subsequent time-steps of the triggering times, given that the transmitted state information is not dropped, and computes/estimates its desired steering angle and also generates a complementary acceleration input. The road curvature needed in the MPC controller is locally estimated by the follower using its on-board camera.

As expressed in (16), the discrete versions of the real-time relative position and lateral errors, i.e., $e_p(k)$ and $e_y(k)$, will decide when an event should be triggered (if they exceed their corresponding

tolerable bounds), resulting in a request for a V2V communication. The feedforward controller, together with the predictive-based feedback controller, aim to, first, stabilize the follower's dynamics, and, second, keep the four path tracking error functions (i.e., e_p, e_v, e_y, and e_ϕ) below their desired thresholds. Boundedness of e_v, e_ϕ is directly guaranteed by the MPC controller. In the following, we introduce our feedforward controller aiming to bound $e_p(k)$ and $e_y(k)$.

Proposition 3.1. *In a leader–follower scenario, let the leader's state vector $[a_{x,l}\ \delta_l]^\top$ be communicated with the follower over a V2V channel according to the event-triggered law (16). Given that the communication is successful, the follower's feedforward control input $u_{FF} = u_{FF}^{e_p} + u_{FF}^{e_y}$ generates $[a_{x,FF}^d\ \delta_{FF}^d]^\top$, minimizing the relative distance error e_p and ensuring boundedness of the lateral error e_y such that $|e_y| \le e_y^{max}$, with u_{FF}, $u_{FF}^{e_p}$ and $u_{FF}^{e_y}$(in the s-domain) expressed as*

$$u_{FF}(s) = u_{FF}^{e_p} + u_{FF}^{e_y} = \frac{a_{x,FF}^d(s)}{a_{x,l}(s)} + \frac{\delta_{FF}^d(s)}{\delta_l(s)},$$

$$u_{FF}^{e_p}(s) = \frac{a_{x,FF}^d(s)}{a_{x,l}(s)} = \frac{1}{sh+1},$$

$$u_{FF}^{e_y}(s) = \frac{\delta_{FF}^d(s)}{\delta_l(s)} \le se^{-sh}\frac{\bar{K}\left(se_y^{max} - v_y(s)\right) + v_x\eta(s)}{v_x\bar{K}\left(r_z(s) - \kappa_f v_x\right) + s\ v_x\eta(s)},$$

where $\bar{K}$ is a known constant and $\eta(s)$ is a known nonlinear function of the follower's tire forces and combined tire slips in the s-domain. □

Proof. To minimize e_p, the feedforward controller generates a desired acceleration input $a_{x,FF}^d$. Define $e_p(s)$ as the frequency domain counterpart of $e_p(t)$. It is desired to have $\dot{e}_p = 0$; hence, from (11), we obtain

$$\dot{e}_p = 0 \quad \rightarrow \quad e_v = (\dot{v}_x - r_z v_y)h. \tag{31}$$

According to (31), the desired acceleration generated by the feedforward controller should then be $a_{x,FF}^d = \dot{v}_x - r_z v_y$ to minimize e_p. Taking the s-transform from $\dot{e}_v$ in (12) leads to $se_v(s) = a_{x,l}(s) - (\dot{v}_x - r_z v_y)(s) = a_{x,l}(s) - a_{x,FF}^d(s)$. From (31) we have that $e_v(s) = a_{x,FF}^d(s)h$; hence, $sa_{x,FF}^d(s)h = a_{x,l}(s) - a_{x,FF}^d(s)$, and we readily obtain the following low-pass filter as the feedforward component that minimizes e_p:

$$u_{FF}^{e_p} := \frac{a_{x,FF}^d(s)}{a_{x,l}(s)} = \frac{1}{sh+1}. \tag{32}$$

The lateral error e_y has nonlinear dynamics and is non-convex with regard to the steering input and tire slips. This is due to severe nonlinearities in the combined-slip tire forces and the vehicle lateral/longitudinal dynamics. Therefore, its global minimum cannot be simply obtained by setting $\dot{e}_y = 0$ as there might be local minima, and solving this optimization problem is not straightforward. Here, we design a practical feedforward component $u_{FF}^{e_y}$ to guarantee that $|e_y| \le e_y^{max}$ when a V2V communication is established successfully, for which we use the steering angle of the leader, i.e., δ_l. Let δ_{FF}^d be the desired steering angle that the feedforward controller generates given δ_l. The leader is assumed to drive aligned with the road curvature (either controlled by exogenous control inputs or a rational driver); hence, the desired steering angle of the follower at any point of the path is the steering angle of the leader when it was at that point of the path, i.e., we can write that

$$\delta_{FF}^d(t) = \delta_l(t-h), \quad (h : \text{headway time}). \tag{33}$$

Assuming perfect motion alignment of the leader with regard to the road curvature, the follower's steering angle is in fact compensating its headway attitude error e_ϕ, or simply put, the ideal steering of the follower would result in $e_\phi = 0$. This is, however, not possible in reality due to the vehicle side-slip and the vehicle kinetics due to the combined-slip tire forces. This mismatch between the vehicle heading (due to steering and tire slip ratios) and the path can be precisely modeled using the nonlinear vehicle kinetics and combined-slip tire forces presented in (5) to (9), which is quite exhaustive. Therefore, without loss of generality, we can approximately express the desired steering angle of the follower as

$$\delta^d_{FF}(t) = \bar{K} e_\phi(t) + \eta(t), \tag{34}$$

where $\bar{K}$ is constant during a sample time (includes measured variables and fixed parameters), and $\eta(t)$ is a nonlinear function that can be derived from the prediction model. Taking s-transforms from $\dot{e}_y$ and $\dot{e}_\phi$ in (13) and (14) yields:

$$se_y(s) = v_y(s) + v_x e_\phi(s), \qquad se_\phi(s) = r_z(s) - \kappa_f v_x. \tag{35}$$

The longitudinal velocity v_x can be assumed constant within one sample time, as the sampling periods for ADSs and ADASs are typically very short (i.e., between 10–50 ms). Moreover, we take s-transforms from (33) and (34), which yield

$$\delta^d_{FF}(s) = e^{-sh}\delta_l(s), \qquad \delta^d_{FF}(s) = \bar{K} e_\phi(s) + \eta(s), \tag{36}$$

where $\eta(s)$ can be computed numerically due to its complex nonlinear form. Substituting $e_\phi(s)$ in (35) with its equivalent from the expression (36) results in

$$se_y(s) = v_y(s) + \frac{v_x}{\bar{K}}\left(\delta^d_{FF}(s) - \eta(s)\right), \tag{37}$$

$$\frac{s}{\bar{K}}\left(e^{-sh}\delta_l(s) - \eta(s)\right) = r_z(s) - \kappa_f v_x. \tag{38}$$

From (37) and (38), we can readily obtain the following mixed-phase nonlinear filter as the feedforward component that bounds the lateral error e_y:

$$u^{e_y}_{FF} := \frac{\delta^d_{FF}(s)}{\delta_l(s)} = se^{-sh}\frac{\bar{K}\left(se_y(s) - v_y(s)\right) + v_x\eta(s)}{v_x\bar{K}\left(r_z(s) - \kappa_f v_x\right) + sv_x\eta(s)}. \tag{39}$$

It is desired to have $|e_y(s)| \le e_y^{max}$; therefore, $u^{e_y}_{FF}$ should generate the desired steering angle such that the following inequality holds, for which we obtain from (39) that

$$\frac{\delta^d_{FF}(s)}{\delta_l(s)} \le se^{-sh}\frac{\bar{K}\left(se_y^{max} - v_y(s)\right) + v_x\eta(s)}{v_x\bar{K}\left(r_z(s) - \kappa_f v_x\right) + sv_x\eta(s)}, \tag{40}$$

where the inequality in (40) refers to the modulus of the complex variable s. Defining $[a^d_{x,FF}\ \delta^d_{FF}]^\top$ as the feedforward control output and $[a_{x,l}\ \delta_l]^\top$ as its input, the proof is then complete from (32) and (40). □

Remark 4. Both components of the feedforward controller in (32) and (39) can be transformed to their time-domain forms by applying the inverse s-transform. The nonlinear frequency domain transfer function in (39) is proper ($\eta(s)$ is a high-degree function in s). However, it is a mixed-phase filter showing minimum-phase and non-minimum-phase behavior across various frequency ranges. This is, in fact, intuitive and consistent with the bicycle model we employed in this article, which is known to be a non-minimum-phase system at low speed and minimum- phase system at high speed. This entails that the feedforward component derived in (39) performs reliably at high

speeds, which is indeed desired, especially due to the fact that the lateral error is usually quite controllable at low speeds.

3.4 Remote Estimation of Leader's State

The feedforward controller is incorporated in the control loop only when the leader's state information is successfully communicated with the follower. In case the follower is in need of a state update from the leader, an event will be triggered according to the triggering law (16). If an event is triggered while no update from the leader is received by the follower, i.e., the transmitted data packet is lost, then a remote state estimator at the follower is incorporated to estimate the leader's acceleration state using the follower's onboard sensors. In a persistence manner, there would be two communication attempts in a row. Then, the estimation process starts if both are unsuccessful. There would be more attempts for successful communications while states are being estimated by the follower. It should be noted that the follower cannot practically estimate the leader's steering angle with the use of the conventional onboard sensors. The follower is equipped with a camera and can estimate the relative velocity. Using a camera-based digital filter, the relative acceleration and, hence, the leader's acceleration can be estimated with some uncertainty that is a function of onboard camera's (or radar's) accuracy. Aiming to reduce e_p, the follower estimates the leader's acceleration and uses this estimate to adjust its own acceleration accordingly.

Let the estimated acceleration of the leader be denoted by $\hat{a}_{x,l} := a^n_{x,l} + n_a$, where $a^n_{x,l}$ is the nominal value and $n_a \sim \exp(\bar{\lambda})$ denotes an exponentially distributed random noise with the rate $\bar{\lambda} > 0$, which is determined by the known accuracy of the follower's onboard sensors. The follower uses this noisy estimation to generate its desired acceleration by minimizing the longitudinal error e_p. Define $a^d_{x,RS}$ as the desired acceleration of the follower based on the remotely estimated acceleration of the leader (i.e., $\hat{a}_{x,l}$). Similar to Proposition 3.1, and expression (31), we obtain that

$$\dot{e}_p = 0 \quad \rightarrow \quad e_v = (\dot{v}_x - r_z v_y)h := a^d_{x,RS} h.$$

From the definition of $\dot{e}_v$ in (12), now computed based on the estimated acceleration of the leader (since the real value of the leader's acceleration is not available due to a packet loss), we obtain that

$$\dot{e}_v = \hat{a}_{x,l} - \dot{v}_x + r_z v_y \quad \xrightarrow{\mathcal{L}} \quad s e_v(s) = a^n_{x,l}(s) + \frac{\bar{\lambda}}{s+\bar{\lambda}} - a^d_{x,RS}(s),$$

where the term $\frac{\bar{\lambda}}{s+\bar{\lambda}}$ is the frequency-domain equivalent of the exponentially distributed random variable n_a with the rate $\bar{\lambda}$. Using $e_v(s) = a^d_{x,RS}(s)h$ and substituting it in the above expression, we obtain the follower's desired acceleration using the remotely estimated acceleration of the leader as

$$a^d_{x,RS}(s) = \frac{a^n_{x,l}(s)}{sh+1} + \frac{\bar{\lambda}}{(sh+1)(s+\bar{\lambda})}. \tag{41}$$

Comparing the expression (41) with the feedforward component $u^{e_p}_{FF}$ in (32), it is clear that if the follower can estimate the leader's acceleration perfectly, i.e., the measurement sensors are noise-free, then V2V communication becomes irrelevant as the two expressions coincide when $\bar{\lambda} = 0$.

Note that, one can use other distributions for the noise process n_a instead of exponential distribution, e.g., Gaussian distribution. For the selected distribution to be relevant, it should be properly characterized with regard to the sensors' accuracy, i.e., if the accuracy of onboard sensors is low, the distribution should return greater noise realizations with higher probability. The state estimate in (41) will also be modified according to the Laplace transform of the selected distribution.

3.5 V2V-Enabled Feedforward Control and Path Tracking Performance

In this section, we discuss the fact that the incorporation of the feedforward control component improves the performance margins asymptotically. Before stating the main result, we introduce the V2V communication rate as a function of the event-triggered law (16), as follows:

$$r_c = \lim_{T\to\infty} \frac{1}{T} \sum_{k=0}^{T} \mathsf{E}\left[\psi(k)\right], \tag{42}$$

where, clearly, $r_c \in [0, 1]$. In fact, the feedforward control component is asymptotically engaged in the control loop with this rate, and the follower is controlled with the predictive controller only with the rate $1 - r_c$. Equivalently, r_c represents the rate that at least one of the two states $e_p(k)$ and $e_y(k)$ exceeds their thresholds, i.e., $|e_p(k)| > e_p^{\max}$ and/or $|e_y(k)| > e_y^{max}$. In that sense, r_c can be seen as the probability that an event is triggered in the asymptotic regime.

Theorem 3.2. *Consider the described leader–follower problem in which the follower's dynamics is stabilized by the predictive controller in* (25)*. Let the feedforward controller in* (22) *be incorporated in the control-loop when successful V2V communications occur according to the event-triggered law* (16)*. The asymptotic tracking performance, measured by the mean-square value of the follower's state vector, then outperforms the asymptotic tracking performance in the absence of the feedforward controller.*

Proof. To compare the performance of the follower's control mechanism with and without the feedforward component, we first consider that the feedforward controller is absent. This entails that there is no V2V communication between the leader and follower to be triggered when $|e_p(k)| > e_p^{\max}$ and/or $|e_y(k)| > e_y^{max}$. From (24), the follower's state dynamics reduces to

$$x(k+1) = A(k)x(k) + \bar{A}(k)u(k) + \bar{B}(k)\mathbf{w}(k-1) + \Omega(k) + \bar{\omega}(k-1). \tag{43}$$

Knowing that the follower is a controllable system, and from (29), the optimal predictive controller described in (25) with the constraints (26a)–(26g) and the predictive cost function (27) ensures stability of the follower's closed-loop dynamics in the mean-square sense, i.e., there exists $M < \infty$, such that for $x(k) = [a_x(k)\ v_x(k)\ v_y(k)\ r_z(k)\ e_p(k)\ e_v(k)\ e_\phi(k)\ e_y(k)]^\top$, we have that

$$\lim_{k\to\infty} \mathsf{E}\left[\|x(k+1)\|_2^2\right] < M. \tag{44}$$

Let $M_e \in \mathbb{R}^+$ denote the part of M that corresponds to the follower's error states $x_e(k) := [e_p(k)\ e_v(k)\ e_\phi(k)\ e_y(k)]^\top$, such that, having (44), we can write

$$\lim_{k\to\infty} \mathsf{E}\left[\|x_e(k+1)\|_2^2\right] < M_e, \tag{45}$$

where $M_e < M$. With a closer study of the optimization problem (25) and the constraints set (26a)–(26g), one can readily conclude that the first four states $[a_x(k)\ v_x(k)\ v_y(k)\ r_z(k)]$ are guaranteed to be bounded through the hard/holonomic constraints (26d)–(26f). Boundedness of the last four non-holonomic error states $x_e(k)$ cannot be obtained through imposing hard constraints; instead, their boundedness is guaranteed by optimizing the horizon-specific quadratic cost function $\mathcal{J}$ in (27). This means that the weight matrix Q acts only on the error states $x_e(k)$ and not on the holonomic states $[a_x(k)\ v_x(k)\ v_y(k)\ r_z(k)]$. Moreover, problem (25) is a multivariate optimization problem that minimizes $x_e(k)$, i.e., jointly with regard to all error variables. This means that the obtained jointly optimal controller may not minimize all error states globally. Moreover, it is clear that the feedforward control input does not result in the holonomic states to violate their bounds as they are controlled by the hard constraints (26d)–(26f); additionally, the feedforward input acts only on the states $e_p(k)$ and $e_y(k)$. Therefore, to analyze control performance of the follower vehicle, we can solely study the behavior of the non-holonomic state $x_e(k)$.

Recall that with the asymptotic rate of r_c, we have that $|e_p(k)| > e_p^{\max}$ and/or $|e_y(k)| > e_y^{max}$, and with the rate of $1-r_c$, we have that $|e_p(k)| \leq e_p^{\max}$ and $|e_y(k)| \leq e_y^{max}$. Therefore, we can write

$$\lim_{k\to\infty} \mathsf{E}\left[\|x_e(k+1)\|_2^2\right] = r_c \lim_{k\to\infty} \mathsf{E}\left[\|x_e(k+1)\|_2^2 \;\middle|\; |e_p(k)| > e_p^{\max} \vee |e_y(k)| > e_y^{max}\right] \tag{46}$$
$$+ (1-r_c) \lim_{k\to\infty} \mathsf{E}\left[\|x_e(k+1)\|_2^2 \;\middle|\; |e_p(k)| \leq e_p^{\max} \wedge |e_y(k)| \leq e_y^{max}\right] < M_e,$$

where $\vee$ and $\wedge$ represent the logical OR and AND operators, respectively. Stochastic terms $\Omega_e(k)$ and $\bar{\omega}_e(k-1)$ are defined as partials of the stochastic terms $\Omega(k)$ and $\bar{\omega}(k-1)$ in (43) that correspond to $x_e(k)$, where they are zero mean, element-wise independent from each other, and have the respective variances $\sigma^2_{\Omega_e}$ and $\sigma^2_{\bar{\omega}_e}$. Error states (excluding the additive disturbances) are deterministic and, hence, statistically independent from each other. Therefore, we can write

$$\begin{aligned}\lim_{k\to\infty} \mathsf{E}\left[\|x_e(k+1)\|_2^2\right] &= \sigma^2_{\Omega_e} + \sigma^2_{\bar{\omega}_e} + e_\phi(k+1)^2 + e_v(k+1)^2 \\ &+ r_c\left[\lim_{k\to\infty} \mathsf{E}\left[e_p(k+1)^2 + e_y(k+1)^2 \;\middle|\; |e_p(k)| > e_p^{\max} \vee |e_y(k)| > e_y^{max}\right]\right] \\ &+ (1-r_c)\left[\lim_{k\to\infty} \mathsf{E}\left[e_p(k+1)^2 + e_y(k+1)^2 \;\middle|\; |e_p(k)| \leq e_p^{\max} \wedge |e_y(k)| \leq e_y^{\max}\right]\right] < M_e.\end{aligned}$$

If an event is triggered at a time k, then we either have $|e_p(k)| > e_p^{\max}$ or $|e_y(k)| > e_y^{\max}$, or both. Assume first that $|e_p(k)| > e_p^{\max}$, but $|e_y(k)| \leq e_y^{max}$. Without feedforward controller, although possible, there is no guarantee that $|e_p(k+1)| \leq e_p^{\max}$ after having $|e_p(k)| > e_p^{\max}$ since such hard constraints cannot be imposed in the optimal MPC controller on a non-holonomic state. Therefore, in order to guarantee bounded MSE (in the absence of feedforward controller) for all possible scenarios, the upper-bound M_e should satisfy the worst-case scenario, which is when $|e_p(k+1)| > e_p^{\max}$ and $|e_y(k+1)| > e_y^{max}$. Hence, M_e should satisfy

$$\bar{M}_e \triangleq \sigma^2_{\Omega_e} + \sigma^2_{\bar{\omega}_e} + e_\phi(k+1)^2 + e_v(k+1)^2 + e_p^{max^2} + e_y^{max^2} \leq \lim_{k\to\infty} \mathsf{E}\left[\|x_e(k+1)\|_2^2\right] < M_e. \tag{47}$$

Incorporating the feedforward controller ensures that if $|e_p(k)| > e_p^{\max}$, then $|e_p(k+1)| = e_p^{\min} < e_p^{\max}$ since the feedforward controller globally minimizes the relative position error (see (31)) by its component in (32). Moreover, when feedforward is engaged, regardless of how the event is triggered, we will have that $|e_y(k+1)| \leq e_y^{max}$. Therefore, considering the worst-case scenario (in the presence of feedforward), we can write

$$\begin{aligned}\lim_{k\to\infty} \mathsf{E}\left[\|x_e(k+1)\|_2^2\right] &= r_c \lim_{k\to\infty} \mathsf{E}\left[\|x_e(k+1)\|_2^2 \;\middle|\; |e_p(k+1)| = e_p^{min}, |e_y(k+1)| \leq e_y^{max}\right] \\ &+ (1-r_c) \lim_{k\to\infty} \mathsf{E}\left[\|x_e(k+1)\|_2^2 \;\middle|\; |e_p(k+1)| > e_p^{\max}, |e_y(k+1)| > e_y^{\max}\right].\end{aligned}$$

Recall that the last term above is the worst-case scenario when the feedforward is absent (see (47)). Therefore, the last equality can be rewritten as

$$\lim_{k\to\infty} \mathsf{E}\left[\|x_e(k+1)\|_2^2\right] < r_c\left[\sigma^2_{\Omega_e} + \sigma^2_{\tilde{\omega}_e} + e_\phi(k+1)^2 + e_v(k+1)^2 + e_p^{min^2} + e_y^{max^2}\right] + (1-r_c)M_e,$$

where $\tilde{\omega}_e$ is the disturbance corresponding to the V2V-transmitted states of the leader and is associated with the error states $x_e(t)$, for which we have that $\sigma^2_{\tilde{\omega}_e} < \sigma^2_{\bar{\omega}_e}$. Define $\tilde{M}_e$ as

$$\tilde{M}_e \triangleq \left[\sigma^2_{\Omega_e} + \sigma^2_{\tilde{\omega}_e} + e_\phi(k+1)^2 + e_v(k+1)^2 + e_p^{min^2} + e_y^{max^2}\right].$$

It is then clear that $\tilde{M}_e < \bar{M}_e < M_e$. Hence, with feedforward controller incorporated, we have that

$$\lim_{k\to\infty} \mathsf{E}\left[\|x_e(k+1)\|_2^2\right] < r_c\tilde{M}_e + (1-r_c)M_e < r_c\bar{M}_e + (1-r_c)M_e < M_e,$$

which proves that the MSE upper bound is lower when feedforward controller is engaged. With similar reasoning, one can reach similar conclusions for the cases that $|e_p(k)| \leq e_p^{\max}$ but $|e_y(k)| > e_y^{\max}$ or $|e^p(k)| > e_p^{\max}$ and $|e^y(k)| > e_y^{\max}$, and the proof is then complete. □

Remark 5. Theorem 3.2 is proved with the V2V communication rate defined in (42), which indicates that every V2V communication triggered according to the triggering law (16) will be successfully received by the follower. However, we assumed to have a packet loss rate of $l_c := 1 - \lim_{T\to\infty} \frac{1}{T}\sum_{k=0}^{T} \mathsf{E}[\gamma_c(k)]$. The results of Theorem 3.2 remain valid for all $1 \geq r_c(1-l_c) > 0$, which is violated only if all the transmissions end up being lost in the channel or no event is ever triggered, which means we basically never have feedforward in the loop. Note that the path tracking performance considering the packet loss rate degrades. However, with any successful V2V communication rate $r_c(1-l_c) > 0$, the mean squared error is upper bounded with a lower bound that is higher than M_e.

The proposed control architecture in this article and its implementation is summarized in Algorithm 1.

4 NUMERICAL RESULTS AND DISCUSSIONS

Performance of the developed integrated leader–follower stabilization and guidance control strategy with intermittent communications is evaluated in this section by using a high-fidelity *CarSim* model, with specifications listed in Table 2. The tests are conducted in various pure- and combined-slip scenarios on slippery surfaces, and the control system has the sampling time of $T_s = 50$ms. The proposed receding horizon controller with a larger state variable vector, which also includes position and heading, was tested in real time using *dSpace MicroAutobox* (for a single vehicle's autonomous navigation in [23, 24]) that confirms the maximum computational time of 25ms. Identification of the tire model parameters (including the friction transition function coefficients) is done by using the combined-slip tire force experimental data from *CarSim* and the nonlinear least squares, with the RMS error of 4.1%. Using the derived regression model, the sufficiently precise coefficients are identified for actual normal loads that can be readily estimated at every sample time [27] (or measured using available suspension height sensors in existing ADAS control systems). Performance of the integrated predictive lateral stabilization and feedforward control framework with intermittent communication (denoted as PC+FF in the plots) is compared with the predictive controller (PC) case only, which does not utilize the feedforward term, and the packet loss case (PC+RS which uses the remote estimator in Section 3.4). The tests include harsh *accelerated* step-steer due to obstacle avoidance and accelerated double-lane-change (DLC) scenarios on highly slippery surfaces (with the coefficient $\theta = 0.4$) under pure- and combined-slip conditions, which are challenging for the existing automated leader–follower control approaches with available proprioceptive and visual sensor data.

4.1 Harsh Cornering on Slippery Surfaces

A scenario of harsh obstacle avoidance that includes accelerating to maintain the distance with the leader is tested on a surface covered with snow, and the performance of the controller in yaw tracking and stabilization, while maintaining the common speed, is investigated. Figure 2 compares the speed and the steering input (by the follower's active front steering actuator) for predictive control utilizing the feedforward term with intermittent communication, including packet drop

ALGORITHM 1: Event-Triggered Control with Intermittent Communications

Initialize: Leader's inputs (δ_0^l, a_{x0}^l) at $k = 0$, and the event trigger indicator $\psi(0) = 1$. Set values for $e_p^{\max}$ and $e_y^{\max}$.

while $k \geq 1$ **do**

`1.Combined-slip prediction model:`

i. Calculate follower's corner slips λ^j, α^j using $v_{r,x}, v_{r,y}$ from (1)–(2)

ii. Obtain follower's tire forces F_x^j, F_y^j using combined-slip variables Γ_x, Γ_y and $g^{\pm}$

iii. Form the prediction model $\dot{x} = f(x, u, \mathbf{w})$ using axles' forces from (ii)

`2.Intermittent V2V communication:`

if $\psi(k-1) = 1$ **then**

Communicate leader's input $\mathbf{w}(k) = [a_{x,l}(k),\ \delta_l(k)]^\top$;

Identify packet dropout based on $P_{d,\circ C(k)}$ and assign $\gamma_c(k) \in \{0, 1\}$;

if $\gamma_c(k) = 0$ **then**

Remote estimation of leader's input $\hat{a}_x^l$ (using follower's onboard sensors);

Compute $a_{x,RS}^d$ and the corresponding FF control input;

else

Use a_x^l and δ_l information to compute $u_{FF} = u_{FF}^{e_p} + u_{FF}^{e_y}$

end

else

Follower includes no FF component at time k

end

`3.Event-triggered V2V communication:`

Form the error dynamics using (11)–(14)

if $|e_p(k)| > e_p^{\max}$ *or* $|e_y(k)| > e_y^{\max}$ **then**

Initiate communication request by $\psi(k) = 1$

else

Set $\psi(k) = 0$ to indicate no request for communication

end

`4.Update follower control input:`

Assign $\zeta(k) = \gamma_c(k)\psi(k-1)$;

Optimization program (25) for u_{MPC} s.t (26a)–(26g);

Generate follower control input $\bar{u}(k) = u_{MPC}(k) + \zeta(k)u_{FF}(k)$

end

Table 2. Specifications of the Test Vehicle

Parameter	Unit	Value	Description
I_w, I_z	[kg/m^2]	1.6, 4310	Moments of inertia
m_s, m	[kg]	1750, 1980	Sprung and total mass
$\sigma_0, \sigma_2, \iota$	[-]	145, 0.001, 11.8	Tire parameters
c, c_λ, c_α	[-]	1.3, 0.57, 0.4	Transition function parameters
l_f, l_r, R	[m]	1.4, 1.41, 0.34	Axles to CG and radius
τ_b, τ_p	[ms]	60, 220	Brake/powertrain bandwidths
$\delta^{\min}, \delta^{\max}, \alpha_b$	[deg]	±30, 9	AFS and saturation slip bounds

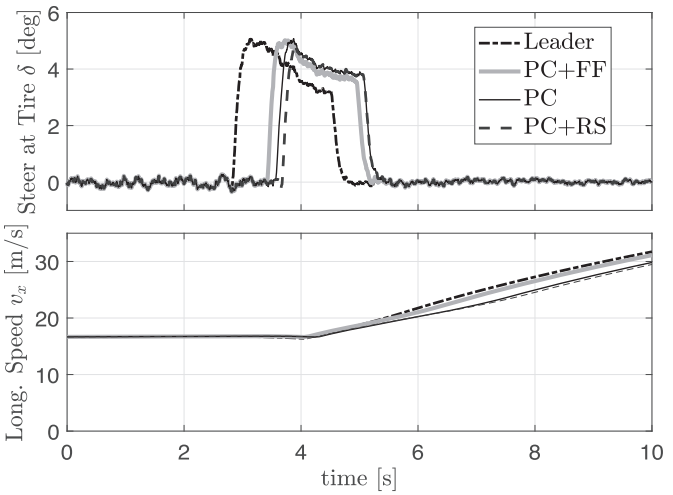


Fig. 2. Steering control input and the longitudinal speed for harsh step-steer on snow. Lateral tire forces drop significantly due to combined-slip effect generated by acceleration during cornering, and large load transfer.

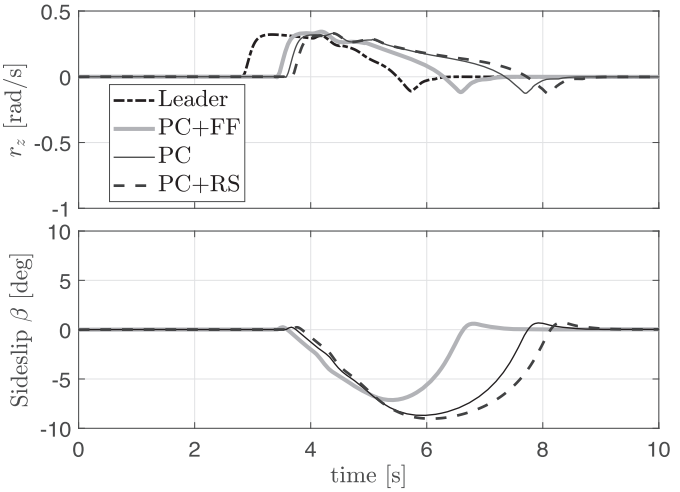


Fig. 3. The measured yaw rate and side-slip angle for stabilization of the follower on the slippery surface with the combined-slip tire force effect.

(i.e, PC+FF and PC+RS) and the pure lateral stability control (PC). This maneuver is challenging due to load transfer in the quick steering and consequent inner tires' capacity reduction that results in instability and large side-slip angles. The vehicle responses are compared in Figure 3, in which both PC and PC+RS apply large steering (and consequently corrective yaw moment) to stabilize the vehicle. However, it deteriorates stability/side-slip due to the combined-slip effect while the follower is accelerating to minimize the relative distance and to reach the common speed. The integrated controller, however, not only keeps side-slip angles low and satisfies the $r(k) - \beta(k)$ safety envelope constraints but maintains the speed.

The measured lateral acceleration and position error are provided in Figure 4, where the PC+FF demonstrates lower lateral error and more stable yaw tracking. This is because the integrated

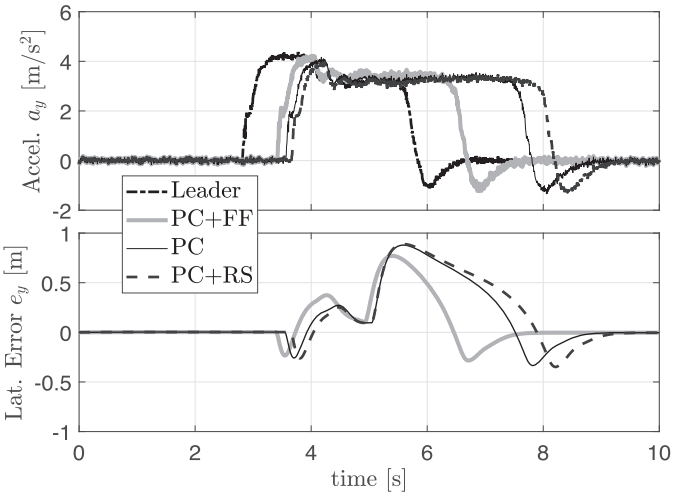


Fig. 4. Lateral acceleration and position error in harsh cornering on snow.

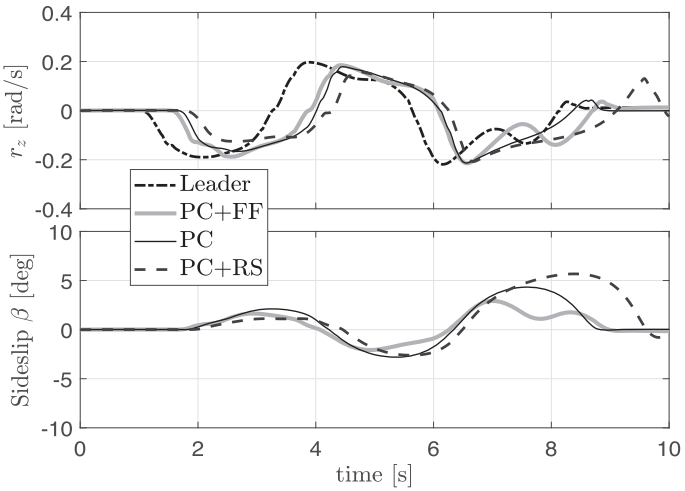


Fig. 5. The measured yaw rate and side-slip angle for stabilization of the follower on wet asphalt with the combined-slip tire force effect.

controller with intermittent communications maintains lateral forces by considering the safe operating envelop utilizing timely actuation by the feedforward term.

4.2 High-Slip Lane-Change and Obstacle Avoidance with Additional Acceleration

To validate the integrated framework in the presence of large longitudinal slips, experiments on lane-change (LC) maneuvers with the initial speed of 70 kph and harsh acceleration (due to the more traction request by the leader) are conducted on a wet surface. Lateral responses as well as yaw/position tracking are compared in Figures 5 and 6.

The tracking and stabilization results confirm that the developed integrated controller stabilizes the follower while minimizing the heading/position error on such slippery surfaces with reduction of tire capacities (due to the longitudinal slip and large load transfer), even with packet drop

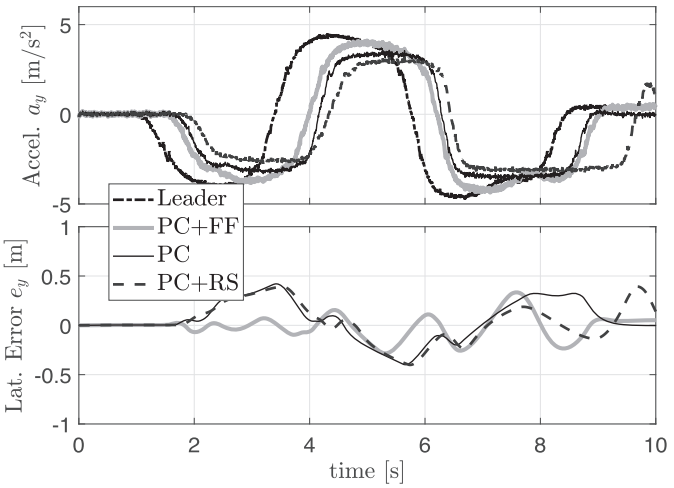


Fig. 6. Lateral acceleration and position error in accelerated obstacle avoidance.

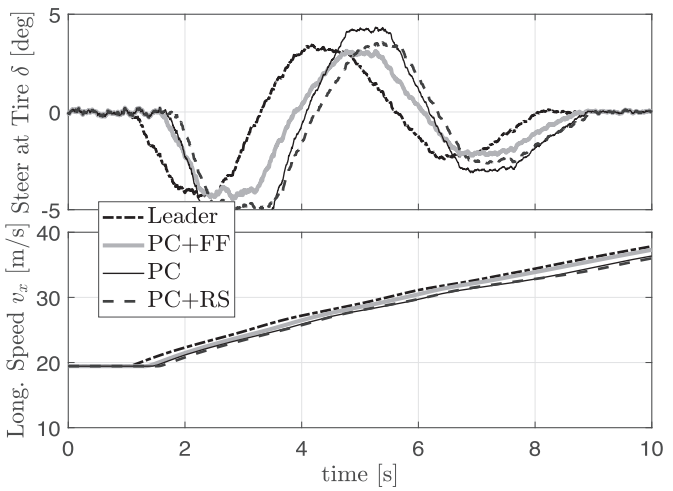


Fig. 7. Steering control input and the longitudinal speed for LC on wet.

incidents. The steering input and the longitudinal speed profile are compared in Figure 7, which confirm less longitudinal speed drop for the PC+FF control strategy, which is decisive in terms of maneuverability for such quick avoidance scenarios. Interestingly, the triggering rates for the two illustrated scenarios with harsh maneuvers have been observed to be $r_c \approx \%10$, which confirms our argument that communication can be triggered intermittently only when necessary.

5 CONCLUSIONS

In this article, an integrated longitudinal and lateral stabilization and path tracking control framework, which considers the combined-slip tire forces and utilizes a combined feedback and feedforward control policy with intermittent V2V communication over erasure channels, is developed. The framework is validated by high-fidelity *CarSim* simulations on various surface conditions in pure- and combined-slip scenarios. V2V communications are triggered when path-tracking errors

exceed given tolerable thresholds. We assume that the dedicated V2V channel is prone to data loss. We derived the optimal MPC controller, with its main goal being stabilizing the vehicle and minimizing the yaw and velocity errors, and proposed a feedforward controller that is activated when a V2V communication is successful to minimize the longitudinal error and ensure a bounded lateral error. In addition, we proposed a sensor-based remote state estimation of the leader's acceleration to compensate the effects of failed transmissions on the path-tracking performance, which allows us to apply complementary control inputs when a V2V communication is scheduled but failed due to data loss. We analytically showed that the incorporation of the V2V-enabled feedforward controller improves the path-tracking performance for the follower in the mean-squared sense. As a future avenue, the team is working towards extending the existing framework to platoons of more than two vehicles and developing a hardware-in-the-loop system using the actual active front steering signal and *CarSim* co-simulations in real time in the presence of model uncertainties.

REFERENCES

[1] Ejaz Ahmed and Hamid Gharavi. 2018. Cooperative vehicular networking: A survey. *IEEE Transactions on Intelligent Transportation Systems* 19, 3 (2018), 996–1014.

[2] Matteo Amodeo, Antonella Ferrara, Riccardo Terzaghi, and Claudio Vecchio. 2007. Slip control for vehicles platooning via second order sliding modes. In *2007 IEEE Intelligent Vehicles Symposium*. IEEE, 761–766.

[3] Giuseppe Araniti, Claudia Campolo, Massimo Condoluci, Antonio Iera, and Antonella Molinaro. 2013. LTE for vehicular networking: A survey. *IEEE Communications Magazine* 51, 5 (2013), 148–157.

[4] Fabio Arena, Giovanni Pau, and Alessandro Severino. 2020. An overview on the current status and future perspectives of smart cars. *Infrastructures* 5, 7 (2020).

[5] Craig Earl Beal and J. Christian Gerdes. 2013. Model predictive control for vehicle stabilization at the limits of handling. *IEEE Transactions on Control Systems Technology* 21, 4 (2013), 1258–1269.

[6] Klaus Bengler, Klaus Dietmayer, Berthold Farber, Markus Maurer, Christoph Stiller, and Hermann Winner. 2014. Three decades of driver assistance systems: Review and future perspectives. *IEEE Intelligent Transportation Systems Magazine* 6, 4 (2014), 6–22.

[7] Irina Bocharova, Boris Kudryashov, Maben Rabi, Nikita Lyamin, Wouter Dankers, Erik Frick, and Alexey Vinel. 2019. Characterizing packet losses in vehicular networks. *IEEE Transactions on Vehicular Technology* 68, 9 (2019), 8347–8358.

[8] Florian David Brunner, W. P. M. H. Heemels, and Frank Allgöwer. 2017. Robust event-triggered MPC with guaranteed asymptotic bound and average sampling rate. *IEEE Trans. Automat. Control* 62, 11 (2017), 5694–5709.

[9] Hong Cai and Yasamin Mostofi. 2022. Co-optimization of motion, communication, and sensing in real wireless channel environments via Monte Carlo tree search. *IEEE Transactions on Control of Network Systems* 9, 3 (2022), 1493–1505.

[10] Claudia Campolo, Alexey Vinel, Antonella Molinaro, and Yevgeni Koucheryavy. 2011. Modeling broadcasting in IEEE 802.11p/WAVE vehicular networks. *IEEE Communications Letters* 15, 2 (2011), 199–201.

[11] Yimin Chen, Chao Lu, and Wenbo Chu. 2020. A cooperative driving strategy based on velocity prediction for connected vehicles with robust path-following control. *IEEE Internet of Things Journal* 7, 5 (2020), 3822–3832.

[12] Shumo Cui, Benjamin Seibold, Raphael Stern, and Daniel B. Work. 2017. Stabilizing traffic flow via a single autonomous vehicle: Possibilities and limitations. In *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 1336–1341.

[13] Dimos V. Dimarogonas and Karl H. Johansson. 2009. Event-triggered control for multi-agent systems. In *Proceedings of the 48th IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*. IEEE, 7131–7136.

[14] Hans Joachim Ferreau, Christian Kirches, Andreas Potschka, Hans Georg Bock, and Moritz Diehl. 2014. qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation* 6, 4 (2014), 327–363.

[15] Jaime F. Fisac, Eli Bronstein, Elis Stefansson, Dorsa Sadigh, S. Shankar Sastry, and Anca D. Dragan. 2019. Hierarchical game-theoretic planning for autonomous vehicles. In *2019 International Conference on Robotics and Automation (ICRA)*. 9590–9596.

[16] Merijn Floren, Amir Khajepour, and Ehsan Hashemi. 2021. An integrated control approach for the combined longitudinal and lateral vehicle following problem. In *2021 American Control Conference (ACC)*. IEEE, 436–441.

[17] Alireza Ghaffarkhah and Yasamin Mostofi. 2011. Communication-aware motion planning in mobile networks. *IEEE Trans. Automat. Control* 56, 10 (2011), 2478–2485.

[18] Lars Grüne. 2009. Analysis and design of unconstrained nonlinear MPC schemes for finite and infinite dimensional systems. *SIAM Journal on Control and Optimization* 48, 2 (2009), 1206–1228.

[19] Ge Guo and Shixi Wen. 2015. Communication scheduling and control of a platoon of vehicles in VANETs. *IEEE Transactions on Intelligent Transportation Systems* 17, 6 (2015), 1551–1563.
[20] Meng Guo and Michael M. Zavlanos. 2018. Multirobot data gathering under buffer constraints and intermittent communication. *IEEE Transactions on Robotics* 34, 4 (2018), 1082–1097.
[21] Xianggui Guo, Jianliang Wang, Fang Liao, and Rodney Swee Huat Teo. 2016. Distributed adaptive sliding mode control strategy for vehicle-following systems with nonlinear acceleration uncertainties. *IEEE Transactions on Vehicular Technology* 66, 2 (2016), 981–991.
[22] Chong Han, Mehrdad Dianati, Rahim Tafazolli, Ralf Kernchen, and Xuemin Shen. 2012. Analytical study of the IEEE 802.11p MAC sublayer in vehicular networks. *IEEE Transactions on Intelligent Transportation Systems* 13, 2 (2012), 873–886.
[23] Ehsan Hashemi, Milad Jalali, Amir Khajepour, Alireza Kasaiezadeh, and Shih-ken Chen. 2020. Vehicle stability control: Model predictive approach and combined-slip effect. *IEEE/ASME Transactions on Mechatronics* 25, 6 (2020), 2789–2800.
[24] Ehsan Hashemi, Yechen Qin, and Amir Khajepour. 2022. Slip-aware driver assistance path tracking and stability control. *Control Engineering Practice* 118 (2022), 104958.
[25] Wilhelmus P. M. H. Heemels, Karl Henrik Johansson, and Paulo Tabuada. 2012. An introduction to event-triggered and self-triggered control. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. IEEE, 3270–3285.
[26] Zichao Huang, Duanfeng Chu, Chaozhong Wu, and Yi He. 2019. Path planning and cooperative control for automated vehicle platoon using hybrid automata. *IEEE Transactions on Intelligent Transportation Systems* 20, 3 (2019), 959–974.
[27] Milad Jalali, Ehsan Hashemi, Amir Khajepour, Shih-ken Chen, and Bakhtiar Litkouhi. 2017. Integrated model predictive control and velocity estimation of electric vehicles. *Mechatronics* 46 (2017), 84–100.
[28] Yiannis Kantaros and Michael M. Zavlanos. 2016. Distributed communication-aware coverage control by mobile sensor networks. *Automatica* 63 (2016), 209–220.
[29] Roozbeh Kianfar, Paolo Falcone, and Jonas Fredriksson. 2015. A control matching model predictive control approach to string stable vehicle platooning. *Control Engineering Practice* 45 (2015), 163–173.
[30] Hong Li, Donghui Pan, and C. L. Philip Chen. 2014. Intelligent prognostics for battery health monitoring using the mean entropy and relevance vector machine. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 44, 7 (2014), 851–862.
[31] Liya Li, Peng Shi, Ramesh K. Agarwal, Choon Ki Ahn, and Wen Xing. 2019. Event-triggered model predictive control for multiagent systems with communication constraints. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 51, 5 (2019), 3304–3316.
[32] Yongfu Li, Chuancong Tang, Kezhi Li, Xiaozheng He, Srinivas Peeta, and Yibing Wang. 2019. Consensus-based cooperative control for multi-platoon under the connected vehicles environment. *IEEE Transactions on Intelligent Transportation Systems* 20, 6 (2019), 2220–2229.
[33] Yuan-Xin Li and Guang-Hong Yang. 2017. Model-based adaptive event-triggered control of strict-feedback nonlinear systems. *IEEE Transactions on Neural Networks and Learning Systems* 29, 4 (2017), 1033–1045.
[34] Steffen Linsenmayer, Dimos V. Dimarogonas, and Frank Allgöwer. 2017. Event-based vehicle coordination using nonlinear unidirectional controllers. *IEEE Transactions on Control of Network Systems* 5, 4 (2017), 1575–1584.
[35] Dipankar Maity, Mohammad H. Mamduhi, Sandra Hirche, and Karl H. Johansson. 2022. Optimal LQG control of networked systems under traffic-correlated delay and dropout. *IEEE Control Systems Letters* 6 (2022), 1280–1285.
[36] Tony K. Mak, Kenneth P. Laberteaux, Raja Sengupta, and Mustafa Ergen. 2008. Multichannel medium access control for dedicated short-range communications. *IEEE Transactions on Vehicular Technology* 58, 1 (2008), 349–366.
[37] Mohammad H. Mamduhi, Ehsan Hashemi, John S. Baras, and Karl H. Johansson. 2020. Event-triggered add-on safety for connected and automated vehicles using road-side network infrastructure. *IFAC-PapersOnLine* 53, 2 (2020), 15154–15160. 21st IFAC World Congress.
[38] Mohammad H. Mamduhi, Adam Molin, Domagoj Tolić, and Sandra Hirche. 2017. Error-dependent data scheduling in resource-aware multi-loop networked control systems. *Automatica* 81 (2017), 209–216.
[39] Mohammad H. Mamduhi, Domagoj Tolić, and Sandra Hirche. 2015. Robust event-based data scheduling for resource constrained networked control systems. In *2015 American Control Conference (ACC)*. 4695–4701.
[40] Mohammad H. Mamduhi, Domagoj Tolić, Adam Molin, and Sandra Hirche. 2014. Event-triggered scheduling for stochastic multi-loop networked control systems with packet dropouts. In *53rd IEEE Conference on Decision and Control*. 2776–2782.
[41] Philip E. Paré, Ehsan Hashemi, Raphael Stern, Henrik Sandberg, and Karl Henrik Johansson. 2019. Networked model for cooperative adaptive cruise control. *IFAC-PapersOnLine* 52, 20 (2019), 151–156.
[42] Aneesh Paul, Rohan Chauhan, Rituraj Srivastava, and Mriganka Baruah. 2016. *Advanced Driver Assistance Systems*. Technical Report. SAE Technical Paper.
[43] Ankur Sarker, Haiying Shen, Mizanur Rahman, Mashrur Chowdhury, Kakan Dey, Fangjian Li, Yue Wang, and Husnu S. Narman. 2020. A review of sensing and communication, human factors, and controller aspects for information-aware connected and automated vehicles. *IEEE Transactions on Intelligent Transportation Systems* 21, 1 (2020), 7–29.

[44] Raphael E. Stern, Shumo Cui, Maria Laura Delle Monache, Rahul Bhadani, Matt Bunting, Miles Churchill, Nathaniel Hamilton, Hannah Pohlmann, Fangyu Wu, Benedetto Piccoli, et al. 2018. Dissipation of stop-and-go waves via control of autonomous vehicles: Field experiments. *Transportation Research Part C: Emerging Technologies* 89 (2018), 205–221.

[45] Valerio Turri, Bart Besselink, and Karl H. Johansson. 2017. Cooperative look-ahead control for fuel-efficient and safe heavy-duty vehicle platooning. *IEEE Transactions on Control Systems Technology* 25, 1 (2017), 12–28.

[46] E. Velenis, P. Tsiotras, C. Canudas-de Wit, and M. Sorine. 2005. Dynamic tyre friction models for combined longitudinal and lateral vehicle motion. *Vehicle System Dynamics* 43, 1 (2005), 3–29.

[47] Hong Wang, Bing Lu, Jun Li, Teng Liu, Yang Xing, Chen Lv, Dongpu Cao, Jingxuan Li, Jinwei Zhang, and Ehsan Hashemi. 2021. Risk assessment and mitigation in local path planning for autonomous vehicles with LSTM based predictive model. *IEEE Transactions on Automation Science and Engineering* (2021).

[48] Shouyang Wei, Yuan Zou, Xudong Zhang, Tao Zhang, and Xiaoliang Li. 2019. An integrated longitudinal and lateral vehicle following control system with radar and vehicle-to-vehicle communication. *IEEE Transactions on Vehicular Technology* 68, 2 (2019), 1116–1127.

[49] Sherali Zeadally, Juan Antonio Guerrero, and Juan Contreras. 2020. A tutorial survey on vehicle-to-vehicle communications. *Telecommunication Systems* 73, 3 (2020), 469–489.

Received 12 January 2023; revised 12 August 2023; accepted 16 September 2023

ACM Transactions on Cyber-Physical Systems

https://tcps.acm.org/

Guide to Manuscript Submission

Submission to the *ACM Transactions on Cyber-Physical Systems* is done electronically through https://mc.manuscriptcentral.com/tcps. Once you are at that site, you can create an account and password with which you can enter the ACM Manuscript Central manuscript review tracking system. Proceed to the Author Center to submit your manuscript and your accompanying files.

You will be asked to create an abstract that will be used throughout the system as a synopsis of your paper. You will also be asked to classify your submission using the ACM Computing Classification System through a link provided at the Author Center. For completeness, please select at least one primary-level classification followed by two secondary-level classifications. To make the process easier, you may cut and paste from the list. Remember, you, the author, know best which area and sub-areas are covered by your paper; in addition to clarifying the area where your paper belongs, classification often helps in quickly identifying suitable reviewers for your paper. So it is important that you provide as thorough a classification of your paper as possible.

The ACM Production Department prefers that your manuscript be prepared in either LaTeX or MS Word format. Style files for manuscript preparation can be obtained at the following location: https://www.acm.org/publications/authors/submissions. For editorial review, the manuscript should be submitted as a PDF or Postscript file. Accompanying material can be in any number of text or image formats, as well as software/documentation bundles in zip or tar-gzipped formats.

Questions regarding editorial review process should be directed to the Editors-in-Chief. Questions regarding the post-acceptance production process should be addressed to the Associate Director of Publications, Sara Kate Heukerott, at heukerott@hq.acm.org.

Ceasing Print Publication of ACM Journals and Transactions

ACM has made the decision to cease print publication for ACM's journals and transactions as of January 2024. There were several motivations for this change: ACM wants to be as environmentally friendly as possible; print journals lack the new features and functionality of the electronic versions in the ACM Digital Library; and print subscriptions, which have been declining for years, have now reached a level where the time was right to sunset print.

Please contact acmhelp@acm.org should you have any questions.

Subscription and Membership Information.

Send orders to:

ACM Member Services Dept.
General Post Office
PO Box 30777
New York, NY 10087-0777

For information, contact:

Mail: ACM Member Services Dept.
1601 Broadway, 10th Floor
New York, NY 10019-7434
Phone: +1-212-626-0500
Fax: +1-212-944-1318
Email: acmhelp@acm.org
Catalog: https://www.acm.org/publications/alacarte

About ACM. ACM is the world's largest educational and scientific computing society, uniting educators, researchers and professionals to inspire dialogue, share resources and address the field's challenges. ACM strengthens the computing profession's collective voice through strong leadership, promotion of the highest standards, and recognition of technical excellence. ACM supports the professional growth of its members by providing opportunities for life-long learning, career development, and professional networking.

Visit ACM's Website: https://www.acm.org.

Change of Address Notification. To notify ACM of a change of address, use the addresses above or send an email to coa@acm.org.

Please allow 6–8 weeks for new membership or change of name and address to become effective. Send your old label with your new address notification. To avoid interruption of service, notify your local post office before change of residence. For a fee, the post office will forward 2nd- and 3rd-class periodicals.