Paper TP0798

Design and Verification of Secure Autonomous Vehicles

Letitia W. Li[12*], Ludovic Apvrille[1], Annie Bracquemond[2]

1. LTCI, Telecom Paristech, Université Paris Saclay, France, letitia.li@telecom-paristech.fr, ludovic.apvrille@telecom-paristech.fr

2. Institut VEDECOM, France, annie.bracquemond@vedecom.fr

Abstract

The rising wave of attacks on communicating embedded systems has exposed their users to risks of information theft, monetary damage, and personal injury. Unprotected remote access to connected features on conventional vehicles already can provide attackers operational control. With the impending introduction of autonomous vehicles, their increased connectivity of autonomous vehicles will offer even more avenues for attack.

In this paper, we present a survey of previous attacks on connected vehicles, and the prospective security risks threatening autonomous vehicles. To address these vulnerabilities, previous projects on connected vehicle security, such as EVITA, proposed countermeasures including key distribution and Hardware Security Modules. The expense and overhead of these solutions can be high, leading us to place them only where necessary. We propose developing systems using our toolkit TTool, which locates insecure communications, and then determines performance impact of these countermeasures through modelling and formal verification.

Keywords:

Autonomous Vehicles, Formal Verification, TTool, Model-Driven Engineering, Vehicle Security

Introduction

The increasing connectivity of vehicles offers conveniences for drivers, such as remote diagnostic apps, automatic toll systems, and mobile connections to the entertainment center. Future innovations, such as V2X systems, emergency braking, and ultimately autonomous driving are expected to revolutionize driving; by reducing traffic, accidents, and human error. However, the increase in connectivity also offers more avenues of attack.

Possible attacks on vehicles range from loss of privacy, such as being able to remotely track a vehicle, to misleading sensors, and most concerning, accessing a vehicle's control system remotely.

This paper discusses modelling and verification of vehicular systems in the context of secure communications. First, we present a chronological survey of publications of attacks on connected cars, and then potentially autonomous cars. Next, we discuss projects and efforts to design secure

architectures for future connected cars. Since the monetary and performance costs of these security solutions may be prohibitive, we discuss how to make use of modelling and formal verification to evaluate those solutions when selecting an architecture and mapping, and add only necessary security. We introduce how our toolkit TTool performs automatic evaluation of both security properties, and the impact of security mechanisms on the safety and performance.

Vulnerabilities

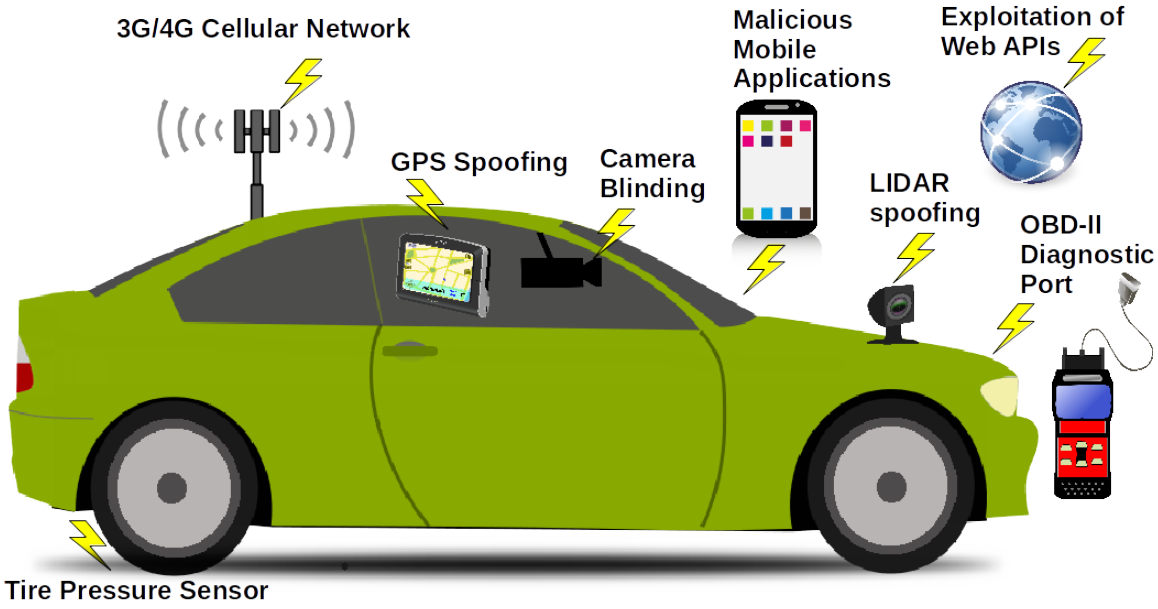

Figure 1: Attacks on Connected and Autonomous Vehicles

Survey of attacks on connected vehicles

We first present a survey of existing and potential attacks on connected vehicles, organized chronologically, to establish the types of attacks we might need to protect against. Figure 1 shows all proof-of-concept attacks on both connected and potentially autonomous vehicles.

The authors of [19] presented privacy and security risks of a Tire Pressure Monitoring System. By monitoring RF signals from the sensors, the authors found the unique sensor ID for identification of the vehicle, and spoofed packets to send fake tire pressure warnings. This attack could force a driver to pull over thinking he had a flat tire, and also the ability to track cars by their wireless signals is a privacy concern.

Telematics units attached to cars, often used for insurance purposes, have been an avenue of attacks on the operation of vehicles themselves. The authors of [11] performed an attack through the OBD-II port using the CAN-to-USB interface. The authors built a CAN packet sniffer/injector and determined how to control units, many commands discovered through fuzzing. Furthermore, the attack restarts the ECU and erases any evidence of the attack code. However, the attack required physical access to the OBD device.

A recent minor attack [24] used a malicious app on a paired smartphone and a car with an OBD-II scan tool. When an OBD-II and smartphone with diagnostic app are connected through

bluetooth, an attacker can send malicious CAN frames to the vehicle. While it required the user to accidentally install the malicious app and for the car to have a telematics unit installed, it demonstrates another avenue of attack.

The authors of [6] analysed the Metromile TCUs, an aftermarket device interfacing with the OBD-II port. The ssh keys were common to all TCUs and could be acquired by dumping the NAND flash. Updates to the TCU were sent through SMS messages, which the authors used to create a new console starting a remote shell to obtain access to the device through *ssh*. The authors note that the update did not have the vehicle verify the server's identity, which is a major vulnerability.

New phone apps allow users to control comfort settings. However, vulnerabilities in these apps, such as lack of authentication protocols, can be a problem. One such example is the WebAPI for the Nissan Leaf, which required only a VIN number to access certain climate control and status of a vehicle [10]. While they did not offer an avenue for attack on the car operation itself, except for draining the battery, researchers were able to recover driving history, which may be a privacy concern.

Most notably, Miller and Valasek's hack is completely remote and prompted a recall and change in Sprint's network [15]. The authors connected to the built-in telematics unit Uconnect's Diagnostics Bus, open to any 3G device using Sprint. They reprogrammed the unit with custom firmware to send CAN messages and control ECUs. This attack is unique as it could attack a vehicle anywhere on the Sprint network and did not require that the vehicle have more than the default setup.

Most recently, researchers presented how to gain control of a Tesla that connected to their malicious wifi hotspot [8]. They reported that a vulnerability in the Tesla's browser would allow them to run code if it visited their page. The malicious code helped them gain access to the car's head unit, which they used to overwrite the gateway to the CAN bus, and subsequently inject their own commands onto the CAN bus. Tesla has reportedly responded by requiring that firmware of components writing to the CAN bus be code signed.

Survey of identified attacks on future autonomous vehicles

Unlike conventional vehicles, autonomous vehicles rely on their sensors to perceive the world around them. To blind or any of these sensors could lead to a grave impact on vehicle safety. While autonomous vehicles remain in the development phase and there do not exist production models to attack, researchers have proposed proof-of-concept attacks on their sensors.

In [18], the authors blinded the camera with a laser, strong enough that the camera cannot adjust exposure to render background visible. On LIDARS, it was possible to cause the LIDAR to detect fake objects with spoofing or replay attacks.

GPS spoofing was demonstrated by researchers from the University of Texas [9]. The authors demonstrated how they created false GPS signals to redirect a GPS-guided drone, and then later misdirected a yacht. Autonomous vehicles greatly rely on GPS for both their location and determination of surrounding environmental data, so this threat could greatly affect vehicle navigation.

Vehicular Ad hoc NETworks (VANETs) offer a collaborative exchange of data on traffic, environmental hazards, etc, which may decrease accidents and traffic jams. However, as predicted,

malicious participants may track the location of a vehicle, or send fake traffic data to clear a road for themselves. In addition, V2X involves connections to numerous vehicles or roadside units. This connective interface offers opportunities for an attacker to access the interior control systems and possibly gain control of the vehicle itself [16]. Countermeasures for anonymity include use of pseudonyms [23], while other solutions in the following section discuss protection of internal vehicular control when possessing wireless communication interfaces.

Table 1 summarizes notable wireless attacks and the targeted component.

Table 1: Summary of Vulnerabilities

Attack Outcome	Vector of Attack	Reference
Control of vehicle	Wifi	Tencent (2016) [8]
Falsified sensor readings	Camera/LiDAR	Petit (2015) [18]
Control of vehicle	3G network	Miller (2015) [15]
Control of vehicle	Smartphone App	Woo (2015) [24]
Loss of privacy	Tire sensor	Roufa (2010) [19]
Misdirected navigation	GPS signals	Humphreys (2008) [9]

Proposed Solutions

To create a standardization for securing automobiles, some efforts led by European manufacturers, suppliers and research labs have defined hardware and software architectural solutions to secure the automotive systems: SEVECOM, SHE, and EVITA. SEVECOM (Secure Vehicle Communications) is a European collaborative project focused on securing communications between vehicles [13]. SHE (Secure Hardware Extension) is interested in adding hardware modules to embedded automotive architectures to accelerate cryptographic processing [22]. Finally EVITA (E-safety vehicle intrusion protected applications), another a European collaborative project completed at the end of 2011, has defined a complete secure automotive architecture, with hardware accelerators and security protocols in particular [20]. The results of EVITA were subsequently widely adopted by several OEMs, and several "EVITA-compatible" products are currently available on the market. We focus on the solutions of the EVITA project in the following section.

The EVITA project assumed a "Dolev-Yao" attacker who can listen to all traffic as well as inject data on the buses. In order to avoid that the memory bus of a processor can be thus spied on, EVITA places both the processor and embedded memory on the same chip. All data that leaves the chip, however, are encrypted. Data exchange are carried out using formally proven security protocols. Security protocols are furthermore used for remote communication, such as for updating firmware or ECU start-up protocol. Finally, the domains are isolated using a firewall whose configuration with real-time updates according to vehicle conditions. The firewall can be used for intrusion detection and raise alerts: a protected mode should engage when an attack is detected.

A dedicated secure module, Hardware Security Module (HSM), can be added to ECUs for

better performance in executing cryptographic protocols. The encryption use previously described requires the frequent execution of cryptographic algorithms, use of keys, generation of random numbers, etc. To reduce the cost of HSMs, three versions have been defined: a light version, an intermediate version, and a full version. The lightweight version is for simple ECUs (a sensor with a basic microcontroller). In all three cases, it is recommended to place the HSM and processor/microcontroller on the same chip.

Periodic distribution of session keys between ECUs is required. These session keys have a lifetime of 2 hours: they are first created at the start of the corresponding service, then renewed as necessary. Keys expire due to their use in MAC. Since the CAN bus supports only relatively short messages, and each message includes a shortened MAC, a brute-force attack could create a collision over time. On newer Ethernet buses, however, this key renewal process could be modified.

All security mechanisms have a downside in terms of hardware costs, energy consumption and performance. Security inevitably slows the exchange of data, increasing latency. For critical messages, such as a braking order, the increased delay may be dangerous. To minimize both monetary and performance costs, it is important to place security mechanisms only where needed. In the following section, we describe how we model and verify security with our toolkit. Unlike other toolkits focusing on performance or functional correctness in architectural exploration, our toolkit includes support for automatic formal verification.

Evaluation of Architectures

Methodology

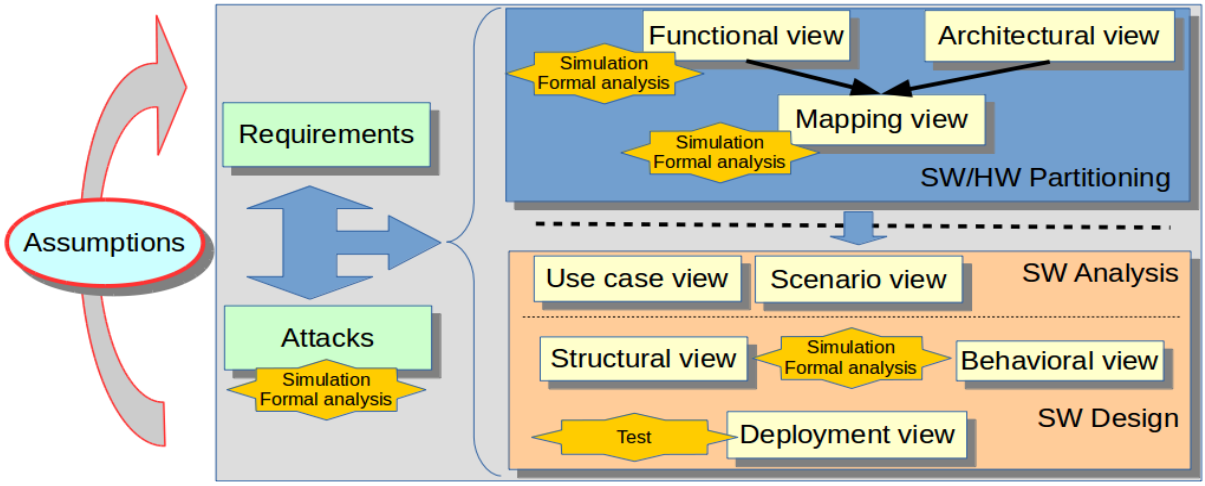

Figure 2: SysML-Sec Methodology

The SysML-Sec methodology was introduced for the design of complex systems, in terms of safety, performance, and security [2]. SysML-Sec, an extension of UML for the design of embedded systems, as shown in Figure 2, starts with Requirements and Attacks analysis, progressing into Hardware/Software Partitioning, and finishing with Software/System Design. The entire design process is supported by TTool [1], a free, open-source, multi-profile toolkit whose main strength is to offer a press-button approach for performing simulation and formal proofs from models. Simulation

and formal proof concerns safety, performance and security properties.

The requirements/attacks stage intends to identify and analyse both requirements and attacks together with the main application functions. Formally defined attack graphs [3] are used to capture attack scenarios, which commonly rely on the exploitation of a combination of vulnerabilities. Attacks can be linked together in order to assess the impact of a specific vulnerability and the need to address it at the risk assessment phase, e.g., once a mapping is under evaluation.

The SW/HW Partitioning phase follows the Y-Chart approach with the modelling of logical tasks/channels, candidate architectures, and mapping of tasks and channels onto the architecture. At this stage, the model does not concern actual algorithms to be implemented, or variable names and values. Tasks abstract the system behaviour, and define concepts of execution time and inter-task communications. The information passed between tasks are abstracted to consider only the size of the data transferred or stored. Once system partitioning has been completed, software design can begin.

The goal of the Software Analysis stage is to develop the software components implementing the functions mapped onto processors at the previous stage. Functions to be implemented are first analysed with SysML-based use case and scenario views to determine a software design in terms of safety or security-related SysML blocks and their interactions, e.g., security protocols.

Iterations over the complete method are assumption-driven [7]. More precisely, the system specification is first limited in scope, then progressively advanced to include further details. During design, software components/blocks are progressively refined until the point where executable code generation is feasible. This refinement also includes adding security-related functions, e.g., cryptographic algorithms, key management policies, and filtering policies.

HW/SW Partitioning

In this paper, we focus on the HW/SW Partitioning, which includes design of the application and architecture. The application modelling includes communicating tasks (functions), and their behaviour. The architectural modelling is displayed as a graph of execution nodes, communication nodes, and storage nodes. Execution nodes, such as CPUs and Hardware Accelerators, include parameters such data size, instruction execution time, and clock ratio. CPUs also must be defined by task switching time, cache-miss percentage, etc. Communication nodes include bridges and buses. Buses connect execution and storage nodes, and bridges connect buses. Buses are defined by parameters such as arbitration policy, data size, clock ratio, etc, and bridges are characterized by data size and clock ratio. Storage nodes are Memories, which are defined by data size and clock ratio.

Mapping involves specifying the location of tasks on the architectural model. A task mapped onto a processor will be implemented in software, and a task mapped onto a hardware accelerator will be implemented in hardware. The exact physical path of a data/event write may also include mapping channels to buses and bridges. Alternatively, if the data path is complex (e.g., DMA transfer), channels can be mapped over communication patterns [5].

Case Study

Previous work [12][17] presented an example of how to verify security protocols such as key

exchange, and how to evaluate the impact of security over performance [21]. In this paper, we focus on the design and evaluation of architecture and partitioning. We present an application and architecture model, locate insecure communications, and then evaluate the impact on performance of securing that communication with cryptographic protocols.

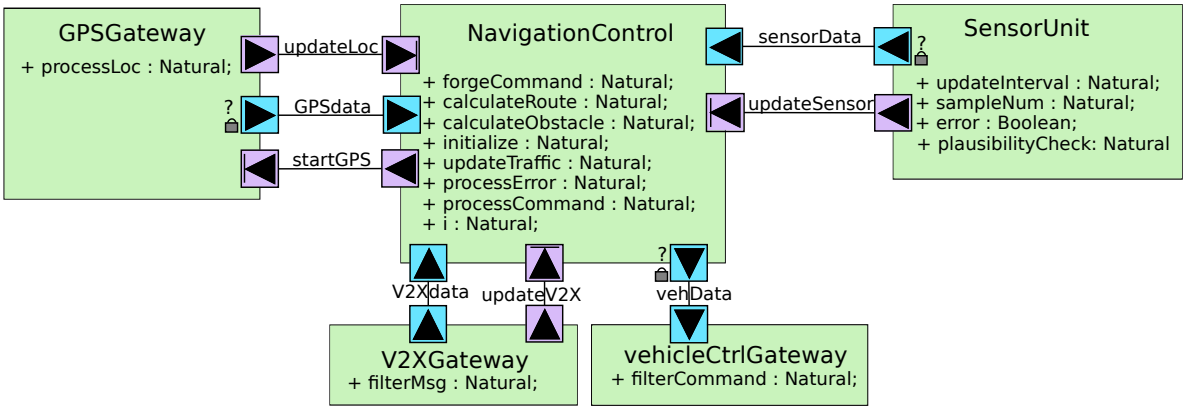

Figure 3: Application Model of Autonomous Vehicle Navigation System

Our case study is based on our real-world design of an Autonomous Vehicle Navigation System. The application model (Figure 3) contains 5 tasks, a main function controlling trajectory *Navigation Control*, route and current location calculation by *GPS Gateway*, environmental detection by *Sensor Unit*, traffic information from the *V2X Gateway*, and vehicle control with V*ehicle Control Gateway*. Sensor, V2X, and GPS data are sent to Navigation Control, which then determines the trajectory, and forges a command to direct the vehicle. The command is sent to *Vehicle Control Gateway*, which determines the validity and safety of the command, and actualizes the command through direct communication with the ECUs.

Based on provided constraints, our toolkit can assess multiple architectures and mappings to determine which fulfil performance constraints (simulation time, bus load, etc.). For example, the sample mapping shown in Figure 4 contains 4 CPUs on which tasks are mapped.

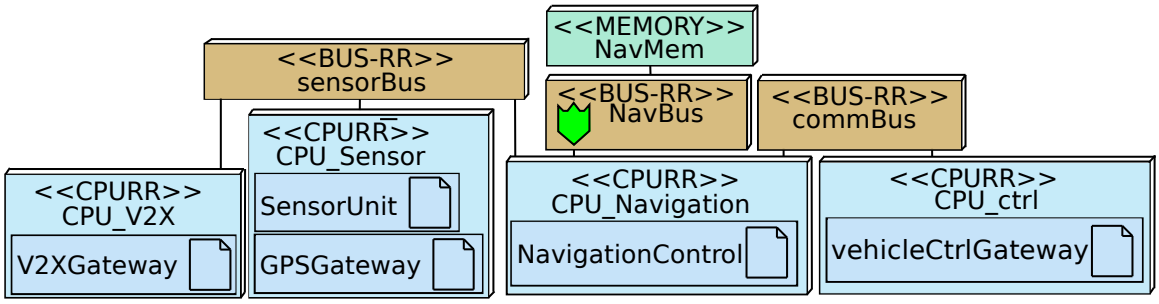

Figure 4: Candidate Mapping for Autonomous Vehicle System

In our context --- security in embedded systems ---, we focus on the properties of authenticity and confidentiality, adapted from the definitions provided by the EVITA project [20].

Authenticity (or integrity) confirms that a piece of data received by an entity really did originate from the expected sender, and that it was not tampered with. It means that, for a particular

exchange, a received message really does correspond to a message sent by the expected entity. In an embedded system, a failure to enforce authenticity may allow an attacker to forge messages, impersonate a trusted component (such as a remote controller), and change the behaviour of a system. For example, an attacker forging sensor data may cause an autonomous vehicle's Navigation Control to calculate trajectory based on the injected data, fail to detect an obstacle, and end up in an accident. Furthermore, we should protect against replay attacks and ensure that if Navigation Control receives the same message twice, this can only be because the sensor sent it twice.

Confidentiality, in our context, deals with the privacy of sensitive data, such as personal information or credentials. Drivers may not want their driving history to be easily accessible, for example. It is also important to verify that any cryptographic keys distributed cannot be recovered by the attacker.

Assuming an external attacker who can spy on vehicle communications along public buses, we determine which communications are critical and need to be secured. In the provided model, the most critical communications are the GPS data, sensor data, and vehicle control commands. While forged V2X data could direct navigation to a non-optimal route, to minimize cost, we only secure safety-critical communications. Since all communication between the Sensor Unit, GPS Gateway, Vehicle Control Gateway, and Navigation Control occur across public buses, we assume that the attacker has complete control to spy on and modify the data. Vehicle commands and sensor data are not especially interesting to the attacker even when recoverable, but their authenticity is critical. However, GPS data indicating directions should remain both confidential and authentic (While the attacker could physically follow a vehicle to its final destination, passengers would likely notice a stalker on a longer trip).

Modelling of Security

Verifying security properties during the HW/SW Partitioning phase requires us to distinguish communications an attacker can intercept and manipulate with those that he can not, model security mechanisms, and determine if implemented protections for communications are sufficiently secure. Unlike in our work verifying security protocols, system models available during architectural/mapping exploration describe programs abstractly, without the use of named attributes and values. From the security requirements and possible attacks proposed during the first analysis phase, we instead determine which tasks will communicate sensitive data. The security of that communication will depend on both the underlying architecture, secure placement of cryptographic material such as keys, as well as the usage of security mechanisms, all of which we discuss in this section.

To model communications as either secure or insecure, we first specify physical locations accessible to attackers. For example, devices communicating on a WiFi network would be modelled as exchanging over a public bus, while the internal bus would be modelled as private. Private buses are marked with a green shield on the architecture diagram as shown in Figure 4.

Dedicated co-processors for security may be added to help a processor with encryption. We model them as Hardware Accelerator nodes that execute instructions in fewer clock cycles than a

regular CPU. The main processor sends data to the HSM to encrypt or decrypt. Since Navigation Control must encrypt and decrypt most of its communications with other tasks, we equip it with a dedicated co-processor. While the transmission and reception of data from the HSM include their own performance costs, the speedup should offset the additional processing.

Specialized security operators signal execution of a security operation. These operators can represent symmetric encryption, forging of a key, decryption, forging a nonce, etc. Encryption and decryption of data takes execution time to perform, increasing CPU usage. In addition, sending an encrypted message sometimes means sending more bits, as the message size can be increased by encryption. In such case, the execution time is increased due to the time required to send a larger message, and more memory is used to store the extra data. These additional latencies and bandwidth usage affect schedulability and performance. In addition to indicating additional overhead to the simulator, these security operators are used to tag channels secured with the corresponding security mechanism to indicate to the security verifier where data is secured.

Formal Verification

Security analysis is automatically performed with a SysMLSec-to-ProVerif model transformation. ProVerif is a verification tool operating on designs described in pi-calculus [4]. A ProVerif specification consists of a set of processes communicating on public and private channels. Processes can split to create concurrently executing processes, and replicate to model multiple executions (sessions) of a given protocol. Cryptographic primitives such as symmetric and asymmetric encryption or hash can be modelled through constructor and destructor functions. ProVerif assumes a Dolev-Yao attacker, which is a threat model in which anyone can read or write on any public channel, create new messages or apply known primitives.

ProVerif provides its user with the capabilities to query the confidentiality of a piece of data, the authenticity of an exchange, or the reachability of a state. The tool then presents a result to the user that is either *true* if the property is verified, *false* if a trace that falsifies the property has been found, or *cannot be proved* if ProVerif failed in asserting or refuting the queried property.

A TTool user indicates which channels send sensitive data on the application design diagram, as shown marked with the grey lock with a question mark as shown on Figure 3, and the information is passed to the ProVerif security verifier. The verifier returns whether the attacker can recover data from those channels, which can help the user decide where to encrypt data/place HSMs. After the placement of security mechanisms, the user can then check if their security measures are sufficient with a second verification.

In our example, we demonstrate the capabilities of our toolkit to analyse architectures and mappings in terms of their security properties and performance. In the interest of space, we analyse the Autonomous Vehicle model on 3 representative architectures: with 4 CPUs shown in Figure 4, 4 CPU with HSM supporting Navigation Control, and with a single CPU. The lack of parallelism in the 1 CPU architecture will increase execution time and CPU usage, decreasing performance. However, all communication on the 1 CPU architecture can be considered secure since all tasks are mapped to that

CPU and communicate within it. We next analysed if the additional overhead due to securing communications causes the distributed architecture to demonstrate worse performance. We automatically generated a secured model with additional encryption/decryption operators, then modified it for all security operations to be performed by a dedicated cryptographic accelerator. By modelling the addition of a cryptographic accelerator, we analyse if the solutions presented in EVITA fulfil security and performance requirements before the selection of an architecture.

Performance Analysis

Table 2: Performance Analysis

Architecture	**Latency (cycles)**	**Nav. CPU usage (%)**	**NavBus usage (%)**	**SensorBus usage (%)**
1 CPU	7414	100	2	-
4 CPU unsecured	6772	72	0	2
4 CPU secured	15445	69	0	7
4 CPU HSM	10159	23	0.2	0.2

Performance analysis involves our custom simulation engine. We calculated execution time in cycles, bus and CPU load (measured as the active cycles/total cycles), for the 1 CPU architecture and 4 CPU architecture, then the secured architectures with each task performing their own security computations or by adding a HSM to the task Navigation Control, averaged over 1000 simulations. We notice that initially the distributed architecture executes faster than the 1 CPU architecture, but adding securing greatly increases time and bus/CPU loads due to encryption and renders it non-optimal. While use of a HSM decreases execution time, there is an increase in the usage of the bus between the Navigation Control CPU and the HSM. This analysis might lead us to further explore other architectures, or analyse how the further addition of HSMs might decrease execution time. We can consider if the 1.5x speedup justifies the additional hardware, and then analyse if additional HSMs would promote further performance improvement.

While our toolkit will perform security analysis and automotive exploration automatically, the analysis of the model itself for security risks requires designer expertise, for the toolkit does not know what data would be sensitive or the assumed capabilities of the attacker on its own. We are examining how our toolkit can connect the Requirements/Attacks phase with HW/SW Partitioning to ensure all security requirements and possible attacks are taken into account.

Conclusion

Connected and future autonomous vehicles offer great conveniences for users, but their security must be considered to ensure that they could not be remotely attacked and compromise the safety of the driver. In this paper, we first presented a survey of possible attacks on connected and autonomous vehicles. Next, we discussed proposed countermeasures. We have demonstrated how our toolkit analyses models for secure communication, automatically generates secured models, and

estimates performance overhead of security mechanisms at the architecture level.

Modelling and verification of security remains a continual subject of our research, as we consider how model real-world attacks and their countermeasures. For example, Miller and Valasek's hack through the cellular network re-wrote code of a different component. We plan to model the mapping of component code to memory, and the different protections to ensure its integrity, such as code signing. We also plan to expand security modelling to include access control, and add firewalls to moderate communication between tasks. If one component is compromised, the attack should not be propagated to other units. By limiting the length and contents of messages, the firewall makes an attack more difficult. These future work will offer designers increased support to design and verify systems during architectural and mapping exploration.

References

1. Apvrille, L. (2003). TTool. Retrieved from http://ttool.telecom-paristech.fr.
2. Apvrille, L., & Roudier, Y. (2013). SysML-sec: a SysML Environment for the Design and Development of Secure Embedded Systems. *APCOSEC, Asia-Pacific Council on Systems Engineering*, 8-11.
3. Apvrille, L., & Roudier, Y. (2015, July). SysML-Sec Attack Graphs: Compact Representations for complex attacks. In *International Workshop on Graphical Models for Security* (pp. 35-49). Springer International Publishing.
4. Blanchet, B., Cheval, V., Allamigeon, X., & Smyth, B. (2010). Proverif: Cryptographic protocol verifier in the formal model.
5. Enrici, A., Apvrille, L., & Pacalet, R. (2014, September). A UML Model-Driven Approach to Efficiently Allocate Complex Communication Schemes. In *International Conference on Model Driven Engineering Languages and Systems* (pp. 370-385). Springer International Publishing.
6. Foster, I., Prudhomme, A., Koscher, K., & Savage, S. (2015). Fast and Vulnerable: A Story of Telematic Failures. In *9th USENIX Workshop on Offensive Technologies (WOOT 15)*.
7. Genius, D., & Apvrille, L. (2016). Virtual Yet Precise Prototyping: An Automotive Case Study. In *8th European Congress on Embedded Real Time Software and Systems (ERTS 2016)*.
8. Greenberg, Andy. (2016, September 27). *Tesla Responds To Chinese Hack with a Major Security Upgrade*. https://www.wired.com/2016/09/tesla-responds-chinese-hack-major-security-upgrade/.
9. Humphreys, T. E., Ledvina, B. M., Psiaki, M. L., O'Hanlon, B. W., & Kintner Jr, P. M. (2008, September). Assessing the Spoofing Threat: Development of a Portable GPS Civilian Spoofer. In *Proceedings of the ION GNSS international technical meeting of the satellite division* (Vol. 55, p. 56).
10. Hunt, Troy. (2016, February 24). *Controlling vehicle features of Nissan LEAFs across the globe via vulnerable APIs.* https://www.troyhunt.com/controlling-vehicle-features-of-nissan

11. Koscher, K., Czeskis, A., Roesner, F., Patel, S., Kohno, T., Checkoway, S., & Savage, S. (2010, May). Experimental Security Analysis of a Modern Automobile. In *2010 IEEE Symposium on Security and Privacy* (pp. 447-462). IEEE.
12. Lugou F., Li L., Apvrille L. and Ameur-Boulifa R. (2016). SysML Models and Model Transformation for Security. In *Proceedings of the 4th International Conference on Model-Driven Engineering and Software Development - Volume 1: MODELSWARD*, 331-338.
13. Leinmüller, T., Buttyan, L., Hubaux, J. P., Kargl, F., Kroh, R., Papadimitratos, Raya, M., and Schoch, E. (2006). SEVECOM - Secure Vehicle Communication. In *IST Mobile and Wireless Communication Summit* (No. LCA-POSTER-2008-005).
14. Mejri, M. N., Ben-Othman, J., & Hamdi, M. (2014). Survey on VANET Security Challenges and Possible Cryptographic Solutions. *Vehicular Communications*, *1*(2), 53-66.
15. Miller, C., & Valasek, C. (2015). Remote Exploitation of an Unaltered Passenger Vehicle. *Black Hat USA*.
16. Papadimitratos, P., Buttyan, L., Hubaux, J. P., Kargl, F., Kung, A., & Raya, M. (2007, June). Architecture for secure and private vehicular communications. In *2007 7th International Conference on ITS Telecommunications* (pp. 1-6). IEEE.
17. Pedroza, G., Idrees, M. S., Apvrille, L., & Roudier, Y. (2011, September). A Formal Methodology Applied to Secure Over-the-Air Automotive Applications. In *Vehicular Technology Conference (VTC Fall)*, 2011 (pp. 1-5). IEEE.
18. Petit, J., Stottelaar, B., Feiri, M., & Kargl, F. (2015). Remote Attacks on Automated Vehicles Sensors: Experiments on Camera and LiDAR. *Black Hat Europe*.
19. Roufa, I., Miller, R., Mustafaa, H., Taylor, T., Oh, S., Xu, W., Gruteser, M., Trappe, W., & Seskar, I. (2010, February). Security and Privacy Vulnerabilities of In-Car Wireless Networks: A Tire Pressure Monitoring System Case Study. In *19th USENIX Security Symposium, Washington DC* (pp. 11-13).
20. Ruddle, A., Ward, D., Weyl, B., Idrees, S., Roudier, Y., Friedewald, M., & Rieke, R. (2009). *Deliverable D2. 3: Security requirements for automotive on-board networks based on dark-side scenarios.* Retrieved from http://www.evita-project.org/deliverables.html.
21. Schweppe, H., Roudier, Y., Weyl, B., Apvrille, L., & Scheuermann, D. (2011, September). *C2X Communication: Securing the Last Meter. In The 4th IEEE International Symposium on Wireless Vehicular Communications: WIVEC2011, San Francisco, USA.*
22. Secure Hardware Extension (2012, February). *Embedded World*.
23. Wiedersheim, B., Ma, Z., Kargl, F., & Papadimitratos, P. (2010, February). Privacy in Inter-Vehicular Networks: Why simple pseudonym change is not enough. In *Wireless On-demand Network Systems and Services (WONS)*. IEEE.
24. Woo, S., Jo, H. J., & Lee, D. H. (2015). A Practical Wireless Attack on the Connected Car and Security Protocol for In-Vehicle CAN. *IEEE Transactions on Intelligent Transportation Systems*, *16*(2), 993-1006.