


# Attack Resilience Hyperproperties: Formal Security Analysis of (Automotive) Network Architectures under Active Compromise

Julius Figge <sup>1,2</sup>, David Knuplesch<sup>2</sup>, Andreas Maletti<sup>1</sup>, and Dragan Zuvic<sup>2</sup>

<sup>1</sup> Institute of Computer Science, Leipzig University, 04109 Leipzig, Germany  
{julius.figge, andreas.maletti}@uni-leipzig.de

<sup>2</sup> Mercedes-Benz Tech Innovation GmbH, 89081 Ulm, Germany  
{julius.figge, david.knuplesch, dragan.zuvic}@mercedes-benz.com

**Abstract.** Automotive architectures are evolving with increased computational power and network connectivity, including mandatory over-the-air updates. This transformation enlarges the attack surface and increases security risks that impact safety. This contribution investigates the security of automotive network architectures (ANAs) and application protocols via formal modeling and verification approaches. The focus is on resilience against a strong adversary that is capable of partial component and network compromise. The key questions that are addressed include the formal modeling of ANAs, the impact of compromised components, and the comprehensive evaluation of security properties (SPs). Attack Resilience Hyperproperties (ARHs) are introduced to model complex SPs that involve compromised components. A prototype tool, EXACT, is provided for the analysis of ARHs with the Tamarin Prover. Additionally, a scoring system is proposed to evaluate the robustness of architectural variants. The real-world applicability of these approaches is demonstrated through a case study involving an ANAs and a secure log file upload protocol.

**Keywords:** Formal Verification · Hyperproperties · Automotive Security · Network Architectures

## 1 Introduction

**Motivation** Automotive architectures have evolved from numerous electronic control units (ECUs) [26] with limited computing capabilities to network connected, centralized high-performance computer platforms [12]. Modern vehicles connect to numerous external entities via the Internet, using mobile communications and Wi-Fi through their telematic control unit (TCU) [26]. This transformation is driven by legislation and regulations like UNECE R156 that mandate software update management systems and over-the-air updates [25]. However, the increasing connectivity challenges the development of secure automotive network architecture (ANA) and expands the attack surface. Previously isolated

ANAs are now internet-connected, including backend systems [27]. For instance, updates can reprogram ECUs firmware in safety-critical areas over the internet, which introduces new threats. An attacker that manipulates updates could gain control of the most safety-critical ECUs and pose significant risks [21] to passenger safety by, for example, detonating the airbags [10].

Countermeasures are being implemented to reduce the attack surface. In-vehicle functionality is segmented into domains [5,15] based on safety-criticality [13]. Complementing segmentation, zero trust architecture (ZTA) and trust boundary (TB) approaches enhance component security and are already being incorporated into ANAs [17,13]. Demonstrating the security of an ANA and its protocols is challenging due to the increasing complexity and connectivity. Even so, the high safety, security criticality as well as laws and regulations [6] mandate security against active attackers in partial compromise scenarios. Formal verification enables to unambiguously prove or disprove security aspects. [8,28].

**Research Questions and Contributions** Our investigation into secure ANAs is based on specific research questions. We focus on automotive architectures with network segments following ZTA and TB approaches. Protocols of interest facilitate communication between the vehicle and external partners via the internet and involve numerous participants. Of particular relevance is the resilience against active attackers and partial compromises of the architecture. For this contribution our primary objective is to validate the security properties (SPs), e.g. secrecy of message content, of these protocols and architectures using formal verification. We derive the following research questions:

- RQ1** *How can ANAs, their application protocols, SPs and aspects such as ZTA & TB be formally modeled?*
- RQ2** *How does compromising architectural components (protocol participants and network segments) affect overall protocol security?*
- RQ3** *Can we identify critical points for the attack resilience concerning the desired SPs of a protocol or the architecture as a whole?*
- RQ4** *Is it feasible to provide a comprehensive evaluation of the architecture concerning the validity of SPs relevant to its protocols?*

We propose 3 main contributions that extend existing approaches for formal verification of security protocols to address the identified research questions. **C1** We propose a *structured approach* for formally modeling ANAs and associated application protocols with a strong focus on internet-connected automotive architectures. Our versatile approach is also applicable to cyber-physical systems and general network architectures. It is based on the symbolic and Dolev-Yao model (DY-model), which is also used by the Tamarin Prover that we utilize for verification. Unlike the computational model, these models assume perfect underlying cryptography [4], but the adversary controls the network and can drop, inject, modify, and intercept all messages in that network [1]. We extend the classical DY-model (Fig. 1) to better align with our unique network topology and domain-specific setup. Therefore, we introduce a new adversary model, referred

to as the “Compromised Automotive Realm” model (CAR-model) (Fig. 2).<sup>3</sup> As a novel feature it allows us to consider fine-grained active component and network subsegment compromise<sup>4</sup>. In our approach communication paths and areas can be partially compromised, which allows us to study the impact of ZTA and TB approaches, which are used to prevent data breaches and limit internal lateral movement [23]. TBs are logical separations between components, for which the trust status of subjects changes. They essentially segment the network [17]. We provide design patterns and structured modeling abstractions for Tamarin to support these modeling efforts for *RQ1*.

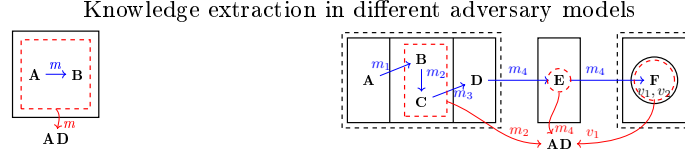


Fig. 1: Dolev-Yao model    Fig. 2: “Compromised Automotive Realm” model

**C2** Our second contribution is a *framework* for evaluating ANAs under active compromise scenarios, based on the validity of SPs in their application protocols. Building on the modeling approach in **C1**, we extend security protocol verification to analyze and evaluate active compromise scenarios using hyperproperties (HPs), which are system properties that extend formal verification by quantifying over multiple execution traces to specify complex security and privacy properties [1]. They generalize relationships between different traces instead of focusing on individual execution traces [22]. As the first of two main contributions, we introduce new security HP in the form of Attack Resilience Hyperproperties (ARHs) (**C2.1**) to address *RQ2-3*. We extend the security verification that is possible with the Tamarin Prover (Tamarin) to analyze an adversary’s capabilities to break desired SPs in all possible compromise scenarios. Grouping traces into sets based on compromises enables the verification of HPs and allows us to verify ARHs to derive previously unattainable insights. This facilitates the analysis of how entities (e.g. ECUs, protocol participants) and domains (e.g. network segments, TB, or subnetworks) contribute to the invalidation of SPs. We propose four ARHs, with the first two delineating *minimal compromise sets* and *single points of failure* critical in breaking SPs. We can identify components and network segments whose compromise does not affect the validity of SPs and are *not responsible for compromise*. The final ARH represents *necessary but not sufficient* compromise scenarios. While necessary, these components alone cannot invalidate the tested SP and further compromises are needed. Insights from the systematic evaluation of ARHs enable comprehensive assessment of architecture and protocol combinations regarding the desired SPs. We propose a score (**C2.2**) for this assessment that facilitates comparisons between

<sup>3</sup> In this paper, only the extraction of content is addressed. Manipulation and injection are carried out analogously.

<sup>4</sup> I.e. the active (partial) takeover of network segments and control devices, as opposed to participation in network traffic.

different architecture and protocol variants. The score is derived from sub-scores for application protocols and SP validity by considering necessary compromises relative to protocol and architecture complexity. This evaluation addresses *RQ4*.

**C3** We complement the theoretical contributions of **C1** Structured Approach and **C2** Framework by our prototype implementation *Extended Adversary Compromise Tool (ExACT)*. It enables evaluating relevant application protocols and associated SPs for given ANAs. Our tool extends the Tamarin Provers capabilities for HPs and enables the examination of architectures and protocols under active compromise scenarios concerning ARHs. Tamarin natively supports HP verification only for observational equivalence [14]. To enable further HP capabilities for Tamarin, we implemented ExACT as a wrapper. ExACT includes a preprocessor for input data, automated verification of protocols and SPs under compromise scenarios, evaluation and extraction of ARHs, and a web interface for result visualization. Thus, we automate all the steps that follow system modeling and graphically present the results. We assess our contribution and verify its success in addressing the identified research questions using a case study from the automotive domain. We use an exemplary ANA with real-world relevance and a sample protocol for the secure upload of log files from a radar ECU<sup>5</sup> to the backend. We utilize our proposed framework to transform the ANA and relevant protocol of our case study into a Tamarin model that incorporates our extension properties and rules. We preprocess the Tamarin model with *ExACT*, automatically verify the ARHs, calculate a score for the architecture, and post-process and visualize the results. An analysis of the results confirms that our contribution facilitated the resolution of the research questions.

**Structure** We first present our *Case Study* (Sect. 2), examining the proposed exemplary network architecture and protocol. In *Formal Verification Framework* (Sect. 3), we introduce our modeling framework, verification conceptual basis, and result evaluation methodology. In *Prototype Implementation* (Sect. 4) we validate our framework and implementation using our prototype *ExACT*. *Related Work* (Sect. 5) compares and distinguishes existing approaches and concepts from our contribution. The *Conclusion and Outlook* (Sect. 6) reflects on our contribution, addresses research questions, and provides an outlook.

## 2 Case Study: Automotive Network Architecture

**Network Architecture** We present a simplified ANA (see Fig. 3) to serve as a case study and to illustrate our contributions. This architecture removes non-essential components and focuses on core aspects like vehicular segmentation and external connectivity [12,19,29]. The abstraction level is based on our CAR-model (Fig. 2).

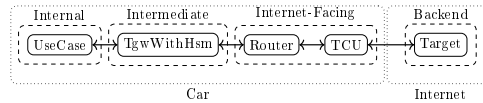


Fig. 3: Exemplary simplified ANA

<sup>5</sup> The radar ECU exemplifies a safety-critical unit related to vehicle control.

The architecture is divided into two components: the *Car* and the *Internet*.<sup>6</sup> Communication flow is bidirectional between protocol participants (entities) connected (by arrows) along the network segments (domains). This allows for communication filtering between domains in accordance with ZTA approaches [23]. The *Backend* is an independent, encapsulated network segment. Entities can communicate within and with adjacent segments. Communication pathways are abstracted from the underlying technology (e.g., Ethernet or CAN). Security and susceptibility to compromise are relevant for our verification framework. The *Internet* contains the *Backend*<sup>7</sup>, including the *Target*, as the protocol communication partner. The *Car* is divided into three domains, based on ZTA principles, with varying levels of safety criticality. Criticality decreases from left to right (Fig. 3), with the most critical components, like radar ECU, on the left and the infotainment system on the right. The least critical domain, *Internet-Facing*, contains two ECU entities: the *TCU*, which acts as a modem communicating with internet endpoints like the backend via cellular network, and the *Router*, which directs data to the appropriate communication partner. Communication occurs openly within segments. Domain compromise grants the adversary abilities in our CAR-model analogous to the DY-model. Communication flows via the transmission gateway (TGW), situated in the *Intermediate* domain, connected to the router. This ECU features an integrated hardware security module (HSM), which holds key material.<sup>8</sup> The TGW transmits communication to the most safety-critical *Internal* domain. Within this domain is the *UseCase*-ECU that initiates the protocol with the *Target*.

**Protocol Description** Our proposed protocol (see Fig. 4) exemplifies communication between an ECU in the most safety-critical domain (*Internal*) and the *Backend*. Specifically, this protocol uploads log files ( $\sim lf$ ) containing telemetry data from a radar ECU (*UseCase*) to the corresponding application in the backend (*Target*). The protocol design is inspired by real-world scenarios but simplified for illustrative purposes (e.g., by reducing it to a single communication direction). The principals of the protocol are the participants introduced within the architecture framework (Figure 3). The security goal of the protocol is the secrecy of the transmitted file.<sup>9</sup> We require that a public-key infrastructure exists, through which the protocol participants can obtain key pairs for asymmetric encryption.

During the initialization phase, a symmetric key ( $\sim sk$ ), which is previously stored in the HSM, is distributed by the *TGW* to both *UseCase* and *Target*. The *TGW*, which is in the *Intermediate* domain, encrypts the key with the public key of the respective receiving entity and transmits these encrypted keys to the entities in

<sup>6</sup> Currently, these are logical groupings, which may become relevant for modeling overall vehicle compromise.

<sup>7</sup> Various entities and domains can be included in the backend to model complex protocols and compromise scenarios analogous.

<sup>8</sup> Key provisioning can occur during production or through advanced protocols.

<sup>9</sup> Additionally, it is possible to verify other relevant properties, such as integrity.

the *Internal* and *Backend* domains. The entities decrypt the received message using their private keys to gain access to the symmetric key.

The core protocol for transmitting the log file is then executed. The *Use-Case* wraps the log file using the previously distributed symmetric key and encrypts it with the router’s public key. During transit the encryption is partially removed, and the wrapped log file is re-encrypted. TBs, as domain borders, serve as checkpoints to inspect the payload and filter or forward messages as necessary, rather than using continuous end-to-end transport encryption. Our exemplary implementation is simplified for clarity by omitting verification at domain transitions. The encrypted and wrapped log file is transmitted from the *Internal* domain along the communication chain to the *Router* in the *Internet-Facing* domain. It is decrypted

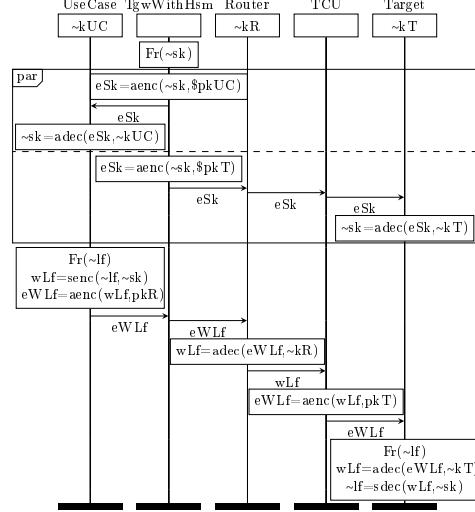


Fig. 4: Logfile-Upload Protocol

using the private key and passed within the domain to the *TCU*, which re-encrypts the symmetrically wrapped log file with the *Target*’s public key and transmits it to the *Backend*. The *Target* receives and decrypts the wrapped log file using first its private and then the symmetric key. The goal is to verify whether the log file remains secure (i.e., not exposed to adversaries) despite potential compromise. The threat model assumes that the attacker can compromise entities, which takes over these protocol participants, their messages, and their internal knowledge. The attacker can also control individual network segments, which allows him to gain control over the transmitted messages.

### 3 Formal Verification Framework

We extend the formal verification of SPs in protocols of ANAs with a focus on active partial compromise. To this end, we incorporate ZTA concepts and controlled compromise of protocol participants as well as network segments. We derive ARHs from verifying SPs under active compromise by externalizing and iterating over compromise scenarios. Our method provides insights into the involvement and interaction of components in invalidating SPs and allows concrete assessments of architectures.

**Approach Formalization** We define the structural components relevant to architecture and protocol modeling, including the segmentation of entities within the network, communication channels, and the extension of the attacker model. We then present our framework for the formal verification of ARHs.

The *architecture* serves as the fundamental component and is represented as a directed graph  $A = (E, L, D)$ , in which the finitely many nodes  $E$  denote *entities* (e.g., ECUs and backend services), the edges  $L \subseteq E \times E$  model the (directed) communication *links* between them, and the partition  $D$  of the entities  $E$  represents the *domains* or specific network segments. We assume that  $(e, e) \in L$  for every  $e \in E$ , so each entity has a link to itself, which we utilize to model internal computation using messages to itself. We additionally include a special node  $e_d \in E$  for every domain  $d \in D$ , that possesses no links, as it will be useful for modeling compromise later. Since nodes now represent entities as well as domains, we introduce *object* as a collective term. A *message*  $m = (l, c) \in L \times N$  consists of a link  $l \in L$  along which it is sent and its contents  $c \in N$  (usually a cryptographic nonce from a set  $N$  of random numbers). The finite set of all messages is denoted by  $M$ . As already indicated internal computations of an entity  $e \in E$  (e.g. en- and decryption of messages, generation of nonces, and hashing) are modelled by messages of the form  $((e, e), c)$  and can lead to new stored values or knowledge from a finite set  $K$ .

The dynamic possession of knowledge is modelled by a Deterministic Finite-State Automaton (DFA) [31] with special state set  $Q = S \times \mathcal{P}(K)$  for a finite set  $S$ . The first and second component of each such state  $(s, K') \in Q$  model the internal state  $s \in S$  and the currently possessed knowledge  $K' \subseteq K$ , respectively. Such a DFA is a tuple  $(Q, \mathcal{A}, q_0, \delta, F)$  with  $Q = S \times \mathcal{P}(K)$  for some finite set  $S$ , an alphabet of *actions*  $I \subseteq M$ , an initial state  $q_0 = (s_0, k_0) \in Q$  indicating the initial internal state  $s_0$  and initial knowledge  $k_0$ , a transition function  $\delta: Q \times \mathcal{A} \rightarrow Q$ , and set of final states  $F \subseteq Q$ . As usual, the transition function  $\delta$  is extended to  $\Delta: Q \times \mathcal{A}^* \rightarrow Q$  via  $\Delta(q, \varepsilon) = q$  and  $\Delta(q, aw) = \Delta(\delta(q, a), w)$  for every  $q \in Q$ ,  $a \in \mathcal{A}$ , and  $w \in \mathcal{A}^*$ . We will not utilize the final states  $F$ .

To model the entities' dynamic behavior in the form of message content, we associate to each entity  $e \in E$  such a DFA  $T(e)$ . For the domains  $d \in D$ , we do not utilize those automata  $T(e_d)$ , i.e. their state and knowledge is not influenced by any messages. Normally, transitions in  $T(e) = (Q, \mathcal{A}, q_0, \delta, F)$  only occur on messages that involve  $e$  as sender or receiver; i.e., the entity only reacts to messages that involve themselves. However, we avoid that distinction and assume that universally  $\mathcal{A} = M$  and  $\delta(q, a) = q$  for all messages  $a \in \mathcal{A}$  that do not involve entity  $e$ . In other words, the state (and knowledge) of the automaton  $T(e)$  associated to object  $e$  simply does not change for messages that do not involve  $e$  as sender or receiver. For theoretical convenience we assume a special content tick = 0 that when sent as a message  $(l, \text{tick})$  along any link  $l \in L$  causes no reaction whatsoever by any object (i.e., no entity or domain changes state or knowledge).

A *protocol* is now simply a finite sequence  $w \in M^*$  of messages. Each entity  $e \in E$  reacts to this protocol in the following manner. Domain entities  $e_d$  with  $d \in D$  do not possess links and therefore do not occur directly in messages; i.e., they have no relevance for the regular protocol execution. Let  $T(e) = (Q, \mathcal{A}, q_0, \delta, F)$  be the DFA associated to  $e$ . The knowledge possessed by  $e$  after executing protocol  $w$  is  $K(e, w) = K'$ , where  $\Delta(q_0, w) = (s, K')$ . In other words, it is the knowledge

encoded in the state attained after reacting to the messages constituting the protocol. The protocol  $w$  is *valid* if for every prefix  $w'((s, r), c)$  of  $w$  with final message  $((s, r), c)$  we have  $c \in K(s, w')$ . In other words, a sender can only send part of its current knowledge as contents, but all entities can generate new knowledge (e.g., new nonces or keys) as part of their transition function.

A *compromise* is simply a subset  $C \subseteq E$  of the entities and domains that are assumed to be compromised and controlled by an adversary. We utilize our proposed CAR-model (Fig. 2). Every message sent to or from the compromised entities  $e \in E$  can be intercepted by the adversary. Also, all messages  $((s, r), c)$  sent to or from a compromised domain  $d \in D$  (i.e.,  $e_d \in C$  and  $\{s, r\} \cap d \neq \emptyset$ ), can be intercepted by the adversary. Additionally, all information available to all compromised entities can be compromised by the adversary. In other words, all contents of messages received or sent by a compromised entity as well as from or to a compromised domain are immediately known to the adversary. In addition, the adversary knows the internal state and knowledge possessed by the compromised entities. In the following, we fix a compromise  $C$ . Although we will only discuss secrecy in this contribution, we prepare the formalization to handle active attacks utilizing the compromised entities as well. We plan to extend our approach to these scenarios and thus allow adversarial actions already in our formalization. To keep the formalization simple, we assume that the adversary will not change the behavior of any entity, rather disabling compromised entities completely and reacting on their behalf.

In other words, once an entity is compromised, it performs no further actions on its own besides receiving messages, whose content is then immediately known to the adversary. Instead, the behavior of the compromised entities is now completely controlled by the adversary that can inject messages using any compromised entity as sender. Additionally, the adversary can inject arbitrary messages from and into compromised domains. For the generation of these injected messages it can utilize any knowledge that it possesses; i.e., it is not limited to the knowledge possessed by the sending entity.

To allow the adversary to generate these injected messages out of turn, we define an *execution trace* as an even-length sequence  $w = m_1 m'_1 m_2 m'_2 \dots m_n m'_n \in M^*$  of messages such that all messages  $m'_1, \dots, m'_n$  are completely controlled and injected by the adversary. If the adversary does not actually want to inject a message, then it injects the special message  $(e, e, \text{tick})$  for any entity  $e \in C$  that it compromised. Since this special message causes no changes, it can be used to model that the adversary does not act at this time. The adversary controls all the messages at even positions in an execution trace, but in addition it is also responsible for all messages  $m_i = ((s, r), c)$  such that  $s \in C$ ; i.e., all regular messages with a compromised sender, and  $\{s, r\} \cap d \neq \emptyset$  for some  $e_d \in C$  with  $d \in D$ ; i.e. spoofing sender or receiver in compromised domains. As expected the non-compromised entities are oblivious to the distinction between injected and regular messages and act normally according to their dynamic behavior (i.e., perform the expected state transitions according to their assigned DFA). Thus



an execution trace  $w$  is *valid* for  $C$  if

$$c \in \begin{cases} K(s, w') & \text{if } s \notin C \\ \bigcup_{c \in C} K(c, w') & \text{otherwise} \end{cases}$$

for every prefix  $w'(s, r, c)$  of  $w$ . Once again a normal entity can only send contents that it knows, whereas compromised entities can utilize the knowledge of all compromised entities.

Given a compromise  $C$ , security property  $S$ , and an execution trace  $w$  that is valid for  $C$  we write  $w \models_C S$  if the property  $S$  is fulfilled for the execution trace  $w$ . Moreover, for a given set  $L$  of execution traces that are valid for  $C$ , we write  $L \models_C S$  if  $w \models_C S$  for every  $w \in L$ . Subsequently, for every language  $L \subseteq M^*$  we let

$$C(S, L) = \{C \subseteq E \mid w \not\models_C S, \text{ valid execution trace } w \in L \text{ for } C\}$$

be the set of compromises that yield a valid execution trace of  $L$  that violates the SP  $S$ . Its complement is  $\overline{C(S, L)} = \mathcal{P}(E) \setminus C(S, L)$  and these scenarios do not permit a violation of the SP  $S$ .

### 3.1 Conceptual Basis and Result Evaluation

The analysis of protocols with Tamarin involves an adversary model, which defaults to an abstraction of the symbolic- and DY-model, in which malicious behavior and communication access is allowed to take place over one cohesive network. In the context of verifying ANAs, which rely on segmentation and delineation, such as in ZTA, it is necessary to extend the communication schema. Instead of using an open network we introduce communication channels that are analogous to the domains in the architecture and model communication within and between these domains. Additionally, the capability of compromising network segments is introduced to re-enable adversary behavior in communication. To model communication of ANAs with non-public communication paths and segments using Tamarin, specific rules for modeling channels are required. To still allow the attacker access to the channel messages, we must also model rules for compromise of the channel communication, which extends the attacker model. The standard adversary model in Tamarin involves network interaction to compromise protocols and verify invalidating SPs, but does not incorporate the compromise of protocol participants themselves. In automotive architecture models, it is expected that ECUs can be compromised [11]. This is particularly relevant because ECUs may possess secret knowledge, such as keys, which can assist the attacker in invalidating SPs. Therefore, beyond the compromise of communication flow in the network, the compromise of entities should be included in the attacker's threat model and thus in the modeling. To examine ANAs under partial compromise scenarios, an extension of the capabilities of the adversary is necessary. We extend these capabilities to include active control of components, which allows the attacker to extract and modify internal knowledge and states

from the compromised components and utilize the internal functionality of these entities. We introduce these attacker capabilities extensions, compared to the DY-model, in the form of our CAR-model (Fig. 2). To limit the complexity, the current attacker model is restricted to the extraction of knowledge within the compromised scope.

The systematic analysis of possible compromise variants enables the derivation of ARHs concerning the SPs of the protocols.

**Attack Resilience Hyperproperties** In this section, we detail the identified ARHs. We extract these based on the evaluation of SP  $S$  in regard to protocols for a given architecture  $A$  under all possible compromise scenarios  $C$ . We can examine the validity of the evaluation results  $C(S, L)$  concerning a ARH in relation to (all) other result elements. We identify four relevant ARHs for evaluating the role of individual components on SPs and identifying critical points for protocols. For the properties at hand, monotonicity largely applies concerning their evaluation results, meaning that an existing compromise invalidating  $S$  will lead to all supersets invalidating  $S$  as well. This does explicitly not hold for *Necessary but not sufficient* (Definition 1), making algorithmic optimization of the evaluation non-trivial.

The first ARH identified is *Necessary but not Sufficient for Compromise*. This describes compromise scenarios necessary as part of a multi-step compromise to invalidate the SP. These are particularly useful in complex protocols for identifying critical components. Intuitively they are those minimal compromise scenarios  $A$  that do not invalidate a security property but are subsets of invalidating scenarios  $C$ .

**Definition 1.** *Necessary but not sufficient*

$$NBNS = \min_{\subseteq} \{A \in \overline{C(S, L)} \mid \exists C \in C(S, L): A \subseteq C, C \setminus A \in \overline{C(S, L)}\}$$

The next identified ARH is *Never Responsible for Compromise*, which are compromise scenarios  $A \in \overline{C(S, L)}$  that do not invalidate the SP and  $C \setminus A \in C(S, L)$  for every superset  $C \in C(S, L)$  that invalidates the SP. In other words, the difference  $C \setminus A$  is responsible for the invalidation. This describes scenarios, in which a given compromise never causes or affects the invalidation of the SP, which allows the identification of non-critical components.

**Definition 2.** *Never responsible for Compromise*

$$NRFC = \{A \in \overline{C(S, L)} \mid \forall C \in C(S, L): A \subseteq C \text{ implies } C \setminus A \in C(S, L)\}$$

The next ARH *Minimal Compromise* describes the minimal compromises necessary to invalidate the SP (i.e., no proper subset of them still invalidates the property). This enables the identification of critical components for the security property within the protocol and the architecture.

**Definition 3.** *Minimal Compromise Subset*

$$MCS = \min_{\subseteq} C(S, L)$$

A special case of this property is the *Single Point of Failure*, which is characterized by a singleton compromise that invalidates the SP. This ARH is significant because a single entity alone is responsible for invalidating the SP. These are typically the sender and the receiver in sender-receiver protocols.

**Definition 4.** *Single Point of Failure*

$$SPOF = \{C \in C(S, L) \mid |C| = 1\}$$

**Score based Evaluation** The ARH evaluation provides meaningful evaluation for architectures and their protocols. We offer a score to compare different architectures or modifications under consistent or evolving protocol requirements and associated SPs. This summarizing score comprises individual scores for the protocol's SPs within the architecture, forming a comprehensive basis for evaluating application protocols against desired SPs. The architecture's score is based on the same data as the ARH evaluation. Our contribution extends component compromise to parameterized controlled evaluation of all compromise scenarios, crucial for creating and calculating the overall score.<sup>10</sup>

Our score  $\bar{s}$  enables the evaluation of all SPs for all protocols of a given architecture  $A$  using a modified form of the weighted average mean, to account for protocols with different amounts of steps and corresponding attack surface. The overall score intuitively is the sum of the architectures protocols scores  $sp_x$  weighted by  $w_x$ . The score  $sp_x$  for a protocol is defined as  $sp_x = \frac{1}{|S|} \sum_{j=z}^n s_{x,z}$ , where every  $s_{x,z}$  is a SP's score of the protocol. Or in other words a protocol score  $sp_x$  is the average of its SP scores.

The score  $s_{x,z}$  for a SP  $S$  is defined as  $\min_{c \in MCS} |c|/p_x$  where  $\min_{c \in MCS} |c|$  is the smallest number of objects (i.e. entities and domains) necessary to invalidate the SP. Essentially the cardinality of the smallest set of the ARH minimal compromise scenario. We define the involved parties for the protocol as  $p_x = |e_x \cup d_x|$ .  $e_x$  defines the relevant entities  $e_x = \{s_2, \dots, s_n, r_1, \dots, r_{n-1}\}$  for protocol  $P = ((s_1, r_1), c_1) \dots ((s_n, r_n), c_n)$ , i.e. the protocols entities excluding sender and receiver. We define  $d_x = \{d \in D \mid \exists e \in e_x : e \in d\}$  with  $d_x$  containing all domains, that occur in the protocol. This value thus refers to the ratio of compromise elements needed for invalidating the SP to the total number of components (i.e. entities, domains) involved in the protocol, excluding sender and receiver.

The individual protocol score  $sp_x$ , the protocol SP's score  $s_{x,z}$  and the overall score  $\bar{s}$  are real numbers confined to the interval  $[0, 1]$ . An overall score value  $\bar{s}$  closer to 1 indicates higher security of the examined architecture concerning its protocol's SPs under active component compromise. The evaluation is based on the components necessarily compromised to invalidate the SP, considering the total number of components in the architecture.

<sup>10</sup> Reviewing all compromise scenarios is necessary because monotonicity of results for subsets and supersets does not apply to all ARHs.

### 3.2 Limitations

Our approach for modeling and verifying security architectures is not protocol-agnostic. The security of the architecture depends on the SPs of its communication protocols. Thus, this limitation is a logical consequence, not a hindrance. Compared to existing adversary models, particularly when using Tamarin (defaulting to DY-model), actively modeling component compromise is necessary for the CAR-model. We propose design patterns within our structured modeling approach to address this. In addition, we currently use a passive attacker that only extracts knowledge. For the validation of security properties such as integrity, the extension to an active attacker is necessary.

Our approach to evaluating ARHs is relatively naive, involving the assessment of all possible compromise scenarios. Although verifying all possible compromise scenarios affects performance, this straightforward method facilitates the prototypical implementation and practical validation of our contribution.

If a SP is invalidated by a counterexample, Tamarin’s heuristics do not necessarily provide a minimal trace [3].<sup>11</sup> A local minimum for invalidating SPs can be found, minimal in terms of necessary compromises but not in trace length. Finding the global minimum is not guaranteed, but it is not required for ARH analysis. Thus, it is reasonable to analyze ARHs based on SPs. Evaluations requiring comparative analysis of traces are not practically feasible at this technical state of development.

## 4 Prototype Implementation “ExACT”

We demonstrate the practical applicability of our contribution regarding the structured modeling and framework for analysing ARHs in ANAs and associated protocols (see Sect. 3), achieved through our prototypical tool ExACT. Our case study serves as the basis for the evaluation (see Sect. 2).

```
rule Channel_Send:
  [ ChOut(fD,tD,fE,tE,m) ]
  -->[ !MOnCh(fD,tD,fE,tE,m) ]

rule Channel_Receive:
  [ !MOnCh(fD,tD,fE,tE,m) ]
  -->[ ChIn(fD,tD,fE,tE,m) ]

/* analogous: tE, From-/tD */
rule Channel_CompromiseFrom:
  [ !Compr(fE), !MOnCh(fD,tD,fE,tE,m) ]
  -->[ ComprHappened(m) ]->[ Out(m) ]
```

Listing 1: Channel Rules

```
rule Tgw_SecretKeySetup:
  [ Fr(~sk), !Tgw($Tgw) ]
  --[ Sk(~sk), InitSK(~sk), ...]->
  [ !PrivSt_Tgw_GenSk($Tgw,~sk) ]

rule Tgw_LeakSecretKey:
  [ !PrivSt_Tgw_GenSk($Tgw,~sk), !Compr($Tgw) ]
  -->[ Out(~sk) ]
```

Listing 2: Internal Knowledge Compromise Rules Excerpt

**Formal Architecture and Protocol Model** We present our structured modeling approach for the case study, focusing on aspects necessary for analyzing ARHs in ANAs with ZTA aspects and CAR-model based compromises. To accurately represent and model the necessary network segmentation due to various physical entities (e.g., the vehicle versus the backend) and the use of ZTA and

<sup>11</sup> A minimal trace refers to a counterexample with the fewest necessary steps.

TB, we utilize channels modeled via rules (Listing 1). These extend the channel model proposed as a design pattern in the Tamarin manual [3].

```
rule TCU_PassEncrypted:
  let im = <'ENC_LF',wLf>
      nEWLf = aenc(wLf,pubkTarget)
      om = <'ENC_WRAP_LF',nEWLf>
  in
  [ ChIn(InternetFacing(),
    InternetFacing(),
    $Router,$TCU,im),
    !Router($Router),!TCU($TCU),
    !Target($Target),
    !Pubk($Target,pubkTarget)
  ]--[ ... ]->[
    !PrivSt_TCU_Intermediate(
      $TCU,wLf),
    ChOut(InternetFacing(),
      Backend(),$TCU,$Target,om) ]
```

Listing 3: Excerpt Rule for Protocol

```
lemma secrecy_except_key_reveal:
  " All lf #i . Secret(lf)@i &
    (not Ex e #r . PkiReveal(e)@r)
    ==> not Ex #k . K(lf)@k "
```

Listing 4: Lemma for Secrecy security property

protocol modeling using a sample rule (Listing 3), demonstrating the modeling of communication with the channel model for incoming and outgoing messages. Protocol verification and ARH extraction are based on systematic evaluation of SPs parameterized over compromise scenarios, represented in Tamarin using lemmas. We analyze the secrecy property for the transmitted log file (Listing 4). This lemma verifies whether the attacker can access the sent log file at any point.

Our extension includes defining the correct domain of the message sender and receiver, enabling modeling of message transmission between and within domains. We use a network channel with persistent messages, restricting the retrieval of messages by protocol participants and the external attacker to once per message through external restrictions. Attackers can extract messages if they control the sender or receiver entity or domain.<sup>12</sup>

In addition to compromising domains and messages, we extend adversarial capabilities to include compromising entities internal knowledge (Listing 2). Knowledge generated or obtained by entities through messages, such as keys and nonces, is stored in the system state as a fact. If an entity is compromised, a rule allows this knowledge to be extracted from the state and accessed by the attacker. We exemplify the

**Tool Structure and Algorithms** Our tool ExACT (Figure 5) enables fully automated ARH evaluation and comprehensive assessment for a provided structured architecture and protocols model. The ExACT wrapper facilitates the analysis of hyperproperties, allowing ARH analysis with Tamarin. The foundation for utilizing the ExACT tool is based on modeling done according to our structured approach. The result is a Tamarin *Architecture and Protocol Model* representing the case study, used as ExACT’s input. The processing of the model by ExACT begins with *Preprocessing*. In this step, individual models for all possible compromise scenarios are created through externalized compromise, extending the models with rules that allow the attacker to take over affected components and communication.

The result is a set of *Enriched Models*, used during *Controlled Verification* as input for *Tamarin*.

<sup>12</sup> The attacker capabilities are limited to passive extraction in this context; invalidation of security properties, for example, through the injection of session keys, is not considered in this example.

After execution, the *Verification Results* are stored in Tamarin-specific output formats and rendered as graphical overviews of SPs invalidating traces.

The parser structures the results into JSON format during *Postprocessing*. Structured results per compromise scenario from postprocessing are used for *ARH Extraction and Score Generation*. We use a prototypical implementation of ARH algorithms to generate JSON result artifacts.

Our web interface prototype visualizes the results. Our graphical interface is a web-based application (Figure 6). It allows evaluation of extracted ARHs and architecture scores concerning compromise scenarios. The prototype is limited to evaluating one protocol and one specific SP, with partial score evaluation implemented in the web interface.

The user interface features a core control element in the header, allowing restriction of the compromise set size, i.e., the minimum and maximum number of objects. The result view can also be limited by type, entity, or domain, and the start and target elements of the protocol can be hidden. The last control element filters the results based on SP invalidation and specific ARHs.

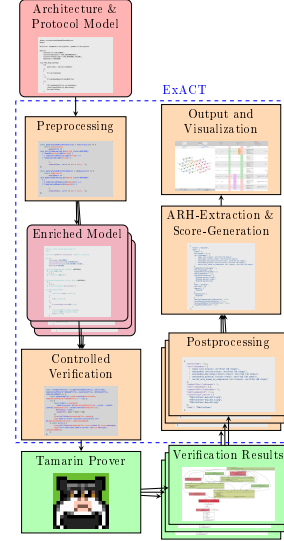


Fig. 5: ExACT

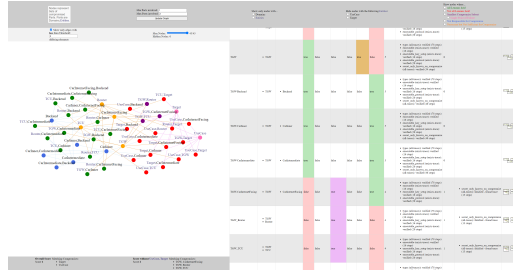


Fig. 6: Screenshot ExACT-GUI

Edges represent all sets for which the current node is a subset. The rendering library incorporates physics calculations to group nodes by their edges, helping identify patterns indicating the involvement of compromise scenarios invalidating SPs. Below the graph is the initial version of the architecture score evaluation, displaying the number and set of components necessary for compromising the protocol, but lacking the overall score calculation. On the right side is the evaluation of compromise scenarios in tabular form (Figure 5). The table displays ARH results for a given scenario, consisting of entities and domains, as well as verified and invalidated associated lemmas and relevant traces.

**Demonstration Example** Finally, we discuss the evaluation (Figure 6) of the architecture and protocol presented in Chapter 2. Using ExACT, we can assess

the analyzed SP, specifically the secrecy of the transmitted log file.

Based on the research question, the following questions arise:

- Is security maintained despite active compromise?
- What impact have components on security, and are there critical points?
- Does network segmentation improve a protocol’s security?

These points can be fully addressed using ExACT. The secrecy of the log file, and thus the security of the protocol and architecture, is not maintained under active attack scenarios. Both *UseCase* and *Target* are critical points and *single points of failure* because the start and end of the protocol possess the relevant file in plaintext (which is typical and expected). The entities *TgwWithHsm*, *Router*, and *TCU*, as well as the domain *CarInternetFacing*, are *necessary but not sufficient* conditions for compromise. The *smallest compromise subset* consists of tuples, always involving *TgwWithHsm*, highlighting *TgwWithHsm*’s critical role in the protocol. The *CarInternetFacing* domain, as the only network segment involved in a compromise, demonstrates effective segmentation into different TB. All other domains are *never responsible for compromise*.

## 5 Related Work

Our contributions expand on existing research through various approaches and concepts. We distinguish our work from other publications.

[2] extended the Dolev-Yao approach to stronger adversary models, including compromising participant states. Our extension to the classic DY-model introduces a strong adversary model, compromising network subsegments and protocol entities, tailored to the automotive domain, unlike existing literature.

[20] model automotive network architecture variants using CTMC, incorporating exploitability scores and component patching rates, enabling the analysis of SPs through probabilistic model checking. [10] formalize attacker privileges in vehicle networks and derive attack trees. In contrast to existing work analyzing attacks on ANAs, our contribution enables the analysis of HPs focusing on SPs.

[9] propose a method to generate sets of counterexamples for invariants of FSMs, analyzing safety properties in the automotive domain. [30] propose counterexample classification to summarize violating traces, providing a prototype for the Alloy Analyzer. Our contribution enables a new approach by systematically analyzing counterexamples for all compromise variant SP invalidations.

[16] generate a security score based on Bayesian networks, attack graphs, and CVSS scores. Our contribution is more specific, assigning a comprehensive score based on formal verification results regarding compromise resilience.

[22] propose DY-HPs for reasoning about HPs involving active adversaries, focusing on theoretical foundations. [1] extend the applied  $\pi$ -calculus to analyze timed HPs of cryptographic protocols, providing proofs for timed safety properties using Tamarin. In contrast our contributions enable the practical analysis of security HP in the form of ARH, through the proposed approach, framework and prototypical tool (ExACT).

[24] developed a prototypical extension of Tamarin for formal analysis of security ceremonies, focusing on automatically generating input files using mutation rules to model human behavior. In contrast, our prototype ExACT is a Tamarin wrapper that controls adversary capabilities, focusing on compromise scenarios. [7] analyze SPs in multi-component systems such as Power Grid Communication and 5G under partial compromise. Unlike our approach, they integrate the compromise directly into their formal model and do not analyze hyperproperties.

## 6 Conclusion and Outlook

In this paper, we propose a novel approach for formal security protocol verification of automotive architectures and protocols.

We developed a structured approach (**C1**) for formally modeling network architectures and application protocols, particularly for internet-connected automotive systems (**RQ1**). With the CAR-model our approach includes an adversary model for component and network subsegment compromise, considering ZTA and TB.

We introduced a formal framework (**C2**) for evaluating network architectures under compromise scenarios and the concept of ARHs (**C2.1**). These hyperproperties allowed us to analyze the impact of compromised components and network segments on the overall security of the protocols (**RQ2, RQ3**). To enable a deeper understanding of security dynamics within network architectures, we propose ARHs, including *necessary but not sufficient* conditions for compromise, *never responsible for compromise*, *minimal compromise sets*, and *single points of failure*. We proposed a scoring system (**C2.2**) to evaluate the validity of SPs relevant to application protocols in network architectures (**RQ4**). Derived from the evaluation of ARHs and the complexity of the architecture, this scoring system enables the comparison of different architectural variants and associated protocol implementations, providing a clear metric for assessing security robustness.

To validate our contributions, we employed a *case study* involving an automotive network architecture and a sample protocol for the secure upload of log files from a radar ECU to the backend, demonstrating the real-world applicability of our approach. We successfully transformed the case studies ANA and relevant protocol into a Tamarin model, incorporating our extension properties and rules. Using our prototypical tool *ExACT* (**C3**), we enabled the analysis of HPs, specifically ARHs, with Tamarin. We automated the verification, evaluation, scoring, and visualization processes. Our contributions effectively addressed the identified research questions. The structured modeling approach, evaluation framework for ARH, and scoring system provided a robust mechanism for assessing and validating the SPs of ANAs under compromise scenarios. We look forward to advancing the prototypical implementation of ExACT into a fully-fledged tool, offering security architects valuable support for their designs. We aim to integrate ExACT with domain processes such as TARA [18] and incorporate it into these workflows. To prepare for this, we plan to implement score



calculation and the capability to evaluate multiple SPs and protocols. We intend to extend the evaluation of compromise scenarios to services and more granular attacker capabilities. Optimizing algorithmic performance, either through native integration of modeling approaches and ARH verification into Tamarin or algorithm improvements, is promising. Finally, we aim to formalize and implement additional ARHs, such as transitive compromise, compromise containment, or compromise propagation.

**Disclosure of Interests.** The authors have no competing interests to declare that are relevant to the content of this article.

## Acknowledgement

This version of the contribution has been accepted for publication, after peer review (when applicable) but is not the Version of Record and does not reflect post-acceptance improvements, or any corrections. The Version of Record is available online at [https://doi.org/10.1007/978-3-032-10444-1\\_2](https://doi.org/10.1007/978-3-032-10444-1_2). Use of this Accepted Version is subject to the publisher's Accepted Manuscript terms of use <https://www.springernature.com/gp/open-research/policies/accepted-manuscript-terms>.

## References

1. Barthe, G., Dal Lago, U., Malavolta, G., Rakotonirina, I.: Tidy: Symbolic Verification of Timed Cryptographic Protocols. In: Proc. 2022 ACM SIGSAC Conf. Comput. Commun. Secur. pp. 263–276. ACM, Los Angeles CA USA (Nov 2022). <https://doi.org/10.1145/3548606.3559343>
2. Basin, D., Cremers, C.: Modeling and Analyzing Security in the Presence of Compromising Adversaries. In: Gritzalis, D., Preneel, B., Theoharidou, M. (eds.) Comput. Secur. – ESORICS 2010. pp. 340–356. Springer, Berlin, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-15497-3\\_21](https://doi.org/10.1007/978-3-642-15497-3_21)
3. Basin, D., Cremers, C., Dreier, J., Meier, S., Sasse, R., Schmidt, B.: Tamarin Prover (Feb 2025)
4. Blanchet, B.: Security Protocol Verification: Symbolic and Computational Models. In: Principles of Security and Trust, vol. 7215, pp. 3–29. Springer Berlin Heidelberg, Berlin, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-28641-4\\_2](https://doi.org/10.1007/978-3-642-28641-4_2)
5. Carlson, B.: The Rise and Evolution of Gateways and Vehicle Network Processing (Jul 2019)
6. Costantino, G., De Vincenzi, M., Matteucci, I.: A Comparative Analysis of UNECE WP.29 R155 and ISO/SAE 21434. In: 2022 IEEE Eur. Symp. Secur. Priv. Workshop EuroSPW. pp. 340–347 (Jun 2022). <https://doi.org/10.1109/EuroSPW55150.2022.00041>
7. Dehnel-Wild, M.: Component-Based Security Under Partial Compromise. Ph.D. thesis, University of Oxford (2018)
8. Dijkstra, E.W.: E.W. Dijkstra Archive: On the reliability of programs. (EWD303). <https://www.cs.utexas.edu/~EWD/transcriptions/EWD03xx/EWD303.html> (Jun 2005)

9. Dominguez, A.L.J., Day, N.A.: Generating Multiple Diverse Counterexamples for an EFSM. Tech. Rep. CS-2013-06 (Sep 2013)
10. Dürrwang, J., Sommer, F., Kriesten, R.: Automation in Automotive Security by Using Attacker Privileges. In: ESCAR 2021 Eur. Frankfurt, Germany (Nov 2021)
11. Elkhail, A.A., Refat, R.U.D., Habre, R., Hafeez, A., Bacha, A., Malik, H.: Vehicle Security: A Survey of Security Issues and Vulnerabilities, Malware Attacks and Defenses. *IEEE Access* **9**, 162401–162437 (2021). <https://doi.org/10.1109/ACCESS.2021.3130495>
12. Figge, J., Knuplesch, D.: Applications of Formal Verification Techniques for Security in Automotive Diagnostics - a Literature-Survey (Nov 2024)
13. Frederic Ameye: Central E/E Arch ('single SoC'), How to ensure security & safety of hyperconverged SDV's? (Sep 2024)
14. Girol, G.: Formalizing and Verifying the Security Protocols from the Noise Framework. Master's thesis, ETH Zurich (2019). <https://doi.org/10.3929/ethz-b-000332859>
15. Huelsewies, M.: Server-Based Architecture - Transformation in Products (Jul 2021)
16. Lei, M., Madi, T.A., Nitschke, M., Zhao, L., Pourzandi, M.: Multi-target Risk Score Aggregation for Security Evaluation of Network Environments. In: 2024 IEEE Int. Conf. Cloud Comput. Technol. Sci. CloudCom. pp. 71–78 (Dec 2024). <https://doi.org/10.1109/CloudCom62794.2024.00023>
17. Macher, G., Sporer, H., Brenner, E., Kreiner, C.: An Automotive Signal-Layer Security and Trust-Boundary Identification Approach. *Procedia Computer Science* **109**, 490–497 (Jan 2017). <https://doi.org/10.1016/j.procs.2017.05.317>
18. Marksteiner, S.F., Schmittner, C., Christl, K., Nickovic, D., Sjödin, M., Sirjani, M.: From TARA to Test: Automated Automotive Cybersecurity Test Generation Out of Threat Modeling. In: 7th CSCS. pp. 1–10. ACM, Darmstadt Germany (Dec 2023). <https://doi.org/10.1145/3631204.3631864>
19. Mocnik, R., Fowler, D.S., Maple, C.: Vehicular over-the-air software upgrade threat modelling. In: Cenex-LCV and Cenex-CAM 2023. UTAC Millbrook, UK (Sep 2023)
20. Mundhenk, P., Steinhorst, S., Lukasiewicz, M., Fahmy, S.A., Chakraborty, S.: Security analysis of automotive architectures using probabilistic model checking. 52nd DAC pp. 1–6 (Jun 2015). <https://doi.org/10.1145/2744769.2744906>
21. Pedroza, G., Idrees, M.S., Apvrille, L., Roudier, Y.: A Formal Methodology Applied to Secure Over-the-Air Automotive Applications. 2011 IEEE VTC Fall pp. 1–5 (Sep 2011). <https://doi.org/10.1109/VETECF.2011.6093061>
22. Rakotonirina, I., Barthe, G., Schneidewind, C.: Decision and Complexity of Dolev-Yao Hyperproperties. *Proc. ACM Program. Lang.* **8**(POPL), 1913–1944 (Jan 2024). <https://doi.org/10.1145/3632906>
23. Rose, S., Borchert, O., Mitchell, S., Connelly, S.: Zero Trust Architecture. Tech. rep., National Institute of Standards and Technology (Aug 2020). <https://doi.org/10.6028/NIST.SP.800-207>
24. Sempredoni, D., Viganò, L.: X-Men: A Mutation-Based Approach for the Formal Analysis of Security Ceremonies. In: 2020 IEEE Eur. Symp. Secur. Priv. EuroSP. pp. 87–104 (Sep 2020). <https://doi.org/10.1109/EuroSP48549.2020.00014>
25. Seo, J., Kwak, J., Kim, S.: Formally Verified Software Update Management System in Automotive. *Veh. 2023* (2023). <https://doi.org/10.14722/vehiclesec.2023.23087>
26. Sommer, F., Kriesten, R., Kargl, F.: Survey of Model-Based Security Testing Approaches in the Automotive Domain. *IEEE Access* **11**, 55474–55514 (Jun 2023). <https://doi.org/10.1109/ACCESS.2023.3282176>

27. Stabili, D., Ferretti, L., Marchetti, M.: Analyses of Secure Automotive Communication Protocols and Their Impact on Vehicles Life-Cycle. In: 2018 IEEE Int. Conf. Smart Comput. SMARTCOMP. pp. 452–457 (Jun 2018). <https://doi.org/10.1109/SMARTCOMP.2018.00045>
28. Ter Beek, M.H., Gnesi, S., Knapp, A.: Formal methods and automated verification of critical systems. *Int J Softw Tools Technol Transfer* **20**(4), 355–358 (Aug 2018). <https://doi.org/10.1007/s10009-018-0494-5>
29. Vamour, C.: Security of Over-the-Air Software Update towards SDV (Sep 2023)
30. Vick, C., Kang, E., Tripakis, S.: Counterexample classification. *Softw Syst Model* **23**(2), 455–472 (Apr 2024). <https://doi.org/10.1007/s10270-023-01118-0>
31. Yu, S.: Regular Languages. In: Rozenberg, G., Salomaa, A. (eds.) *Handbook of Formal Languages: Volume 1 Word, Language, Grammar*, pp. 41–110. Springer, Berlin, Heidelberg (1997). [https://doi.org/10.1007/978-3-642-59136-5\\_2](https://doi.org/10.1007/978-3-642-59136-5_2)