Exploiting Partial Order of Keys to Verify Security of a Vehicular Group Protocol

Felipe Boeira and Mikael Asplund
Dept. of Computer and Information Science
Linköping University, Sweden

***Abstract*—Vehicular networks will enable a range of novel applications to enhance road traffic efficiency, safety, and reduce fuel consumption. As for other cyber-physical systems, security is essential to the deployment of these applications and standardisation efforts are ongoing. In this paper, we perform a systematic security evaluation of a vehicular platooning protocol through a thorough analysis of the protocol and security standards. We tackle the complexity of the resulting model with a proof strategy based on a relation on keys. The key relation forms a partial order, which encapsulates both secrecy and authenticity dependencies. We show that our order-aware approach makes the verification feasible and proves authenticity properties along with secrecy of all keys used throughout the protocol.**

I. Introduction

Security of cyber-physical systems are increasingly becoming a societal concern, as both the attack surfaces and the potential consequences of attacks have increased considerably in recent years. In the automotive domain, connected and automated vehicles (CAV) promise considerable improvements in safety and efficiency, but also require very strict security processes to be trustworthy. Today, the most advanced and complex collaborative vehicular application comes in the form of vehicular platooning where a group of vehicles are jointly controlled by a leader. In this paper we consider the security of platooning as a starting point to investigate automated security proofs, partially ordered key structures, and the process of transforming informal and semi-formal standardisation text to formal models.

Our work was inspired by Basin et al. [1] who in their security analysis of the 5G AKA protocol make a compelling argument for the need of formal models and security specifications to complement protocol standards. In particular, the authors point to under-specification of security properties and assumptions that can in some cases lead to vulnerabilities. Our analysis is focused on a cyber-physical protocol that is currently in the pre-standardisation phase and described in the European Ensemble project, and which also builds on the existing ETSI ITS-G5 and IEEE vehicular networking standards (including security). Together, these form an interesting study object since (i) they will have a real and significant impact on the way future commercial vehicles are operated and controlled, (ii) they represent a typical standardisation product composed of multiple cross-references documents (in our case 8 documents and 617 pages), and (iii) the protocol and the associated security specification describe a complex system with dynamically joining and leaving nodes and a non-trivial key structure.

We perform a structured and formal security analysis of the Ensemble platooning protocol. To perform this analysis several interesting challenges must be overcome. One such challenge is the transformation from informal and semi-formal standard documents - where descriptions are often spread out over multiple documents, contain optional parts, and sometimes overrides previous statements - to a formal model. We show how ASN.1 specifications significantly improve this process and discuss potential benefits of a fully automatic transformation process. Moreover, our analysis also identifies the lack of a corresponding language to capture the behaviour of security checks since the specifications currently included only capture the *structure* of security information, not the *mechanisms* that make use of that information. This gives rise to potential weaknesses depending on the interpretation of the standard.

The second major challenge is of course the model complexity which causes a state space explosion. In our case, even a simple fact such as the secrecy of a long-term key could not be proven on a large computing cluster without manual intervention. Previous works have shown how this process can be aided with so called helper lemmas and oracles. The difficulty with such mechanisms is that they are inherently specific to the problem at hand, and hard to generalise. In this work we explore how the structure of the model can be used to guide the proof strategy. In particular, by considering the ordered structure of the cryptographic keys in the model we make the problem tractable, thereby allowing a more generic proof guidance mechanism.

Secure protocols are often created so that multiple secret keys form dependency chains where the secrecy of one key is dependent on the secrecy of another. This naturally allows formulating provable properties in a way that together satisfy the overall security specification (e.g., see [2]). While existing research has investigated dependencies between keys [3], [4], we show how a resulting partial order of keys is leveraged to tackle proving complexity and provide an automatic key dependency extractor for Tamarin models.

We formulate relevant security properties of vehicular communication protocols and instantiate our model and proof guidance[1] in the Tamarin verifier tool [5]. In our case study there

[1]https://gitlab.liu.se/ida-rtslab/public-code/2022_csf_platooning

are 10 classes of keys with up to a dependency depth of six, and potentially infinitely many instances of keys on a protocol run. Our improved proof strategy allows us to identify the security properties that are met by current protocols and under what circumstances. Overall, our assessment is that security standards for vehicular networks can provide strong security properties, but that ambiguities and implicit assumptions in the standards potentially give room for implementations with lacking security checks.

To summarise, the contributions of this paper are as follows.

- Formulation of a joint secrecy and authenticity relation on a set of keys that potentially forms a partial order, together with an automated proof strategy, a key hierarchy extractor for TAMARIN protocol models that exhibit such partial orders of keys.
- An assessment of the state of security for current vehicular networking protocols and a formal and structured security analysis of vehicular platooning.
- A structured approach for interpreting security standards in order to create models that allow formal reasoning, identification of what is lacking in current standards, and recommendations for future vehicular security standardisation activities.

The remainder of this paper is organised as follows: Section II introduces the area of vehicular networking protocols at large, the Ensemble platooning protocol in particular and formal verification of security protocols. Section III introduces our model of the platooning protocol and the security properties that we verify. Section IV introduces the partial order of keys, and explains how this ordering relation is used to create the proof guidance. The outcome of verifying security protocols under different settings and assumptions is presented in Section V. Finally, Section VI describes related work and Section VII concludes the paper and outlines future work.

II. BACKGROUND

In this section we introduce the standards for vehicular communication and security that are employed in Ensemble. We describe the overall design of the protocol, and provide an overview of how the protocol security properties are verified using the TAMARIN security verification tool.

A. Vehicular network and security standards

In recent years, organisations such as the Institute of Electrical and Electronics Engineers (IEEE) and European Telecommunications Standards Institute (ETSI) have been actively working towards standardisation of vehicular network protocols and applications. The Ensemble protocol [6], [7] is built on top of these existing standards and makes use of their services, so one must understand how these fit together to understand the security implications for vehicular platooning.

Fig. 1 represents the protocol stack and the associated standards for each layer. In Europe, the physical layer and the data link layer are grouped together into the *Access layer* and are part of ITS-G5. More recently, the 5G standard has emerged as a potential replacement at the access layer. Ensemble is designed to work on ITS-G5 together with the Geonetworking (GN) protocol and the Basic Transport Protocol (BTP) which compose the network and transport layer. A set of security profiles for vehicular applications is defined by ETSI based on the primitives and message types defined by IEEE, and are applied to GN and Ensemble messages. We proceed to describe each of these standards briefly (starting from the lowest layer).

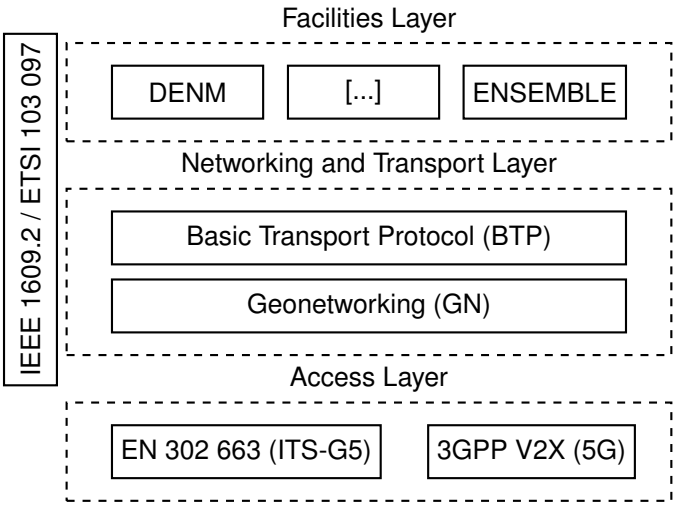

Fig. 1. ENSEMBLE protocol stack.

ITS-G5 (ETSI EN 302 663). At the bottom of the vehicular networking stack we find the access layer, which is of limited interest from a security standpoint, but still relevant to provide context. In Europe and US, the dominating access layer standard for vehicular networks has been based on a flavour of IEEE 802.11 (i.e., essentially wifi) operating at 5.9GHz.

3GPP V2X (5G Release 15+). There are ongoing efforts to integrate 5G as an alternative access layer in the vehicular networking stack [8]. Provided that GN is compatible with V2X as intended [9], we argue that our analysis will still be relevant as the cryptographic operations occur from the networking and transport layer upwards.

GeoNetworking (ETSI EN 302 636-4-1). The GN protocol [10] provides packet routing in vehicular networks with the use of geographical locations for packet transport. In Ensemble only the single hop broadcast (SHB) mode is used.

BTP (ETSI EN 302 626-5-1). BTP [11] defines two header variants: BTP-A for interactive packet transport and BTP-B for non-interactive. Given that BTP does not carry relevant information for our analysis and is included in higher layers' cryptographic operations, we have decided to omit it in our model.

Security Services for Applications and Management Messages (IEEE 1609.2). The primitives for providing security capabilities to vehicular messages are primarily defined by IEEE 1609.2 [12] and its two amendments [13], [14]. The standard defines several data structures for encapsulating data based on the type and origin of the key that is used to encrypt

it, and a `SignedData` structure for storing signature information. The 1609.2 standard uses the concept of recipients to transfer data encryption keys to other nodes, and a sequence of recipients may be included in a message. For instance, a recipient of type `pskRecipInfo` is used whenever the data encryption key is pre-shared between the participants. Alternatively, a node might use an ephemeral data encryption key and a key encryption key to protect the ephemeral. In particular, two recipient types that employ ephemeral keys are used in Ensemble:

- `symmRecipInfo` specifies that the ephemeral data encryption key was encrypted using a symmetric key.
- `rekRecipInfo` specifies that the ephemeral data encryption key was encrypted using a public response encryption key that was not obtained from a `SignedData` structure.

Security header and certificate formats (ETSI TS 103 097). While IEEE 1609.2 defines low-level primitives and data structures to build vehicular messages, the ETSI TS 103 097 [15] specifies message profiles based on the definitions from 1609.2. For instance, ETSI specifies that Decentralized Environmental Notification Messages (DENMs) shall include the certificate as the signer information instead of its digest only. The Ensemble project extends the definitions contained in this standard to reflect the requirements of the messages exchanged in the protocol.

B. Platooning protocol

Ensemble works as a group formation protocol with key establishment and distribution. It consists of four operational modes: `idle`, `join`, `platoon`, and `leave`. In `idle` mode, vehicles announce their interest to form a platoon through the flag `isJoinable`, which is included in a Cooperative Awareness Message (CAM). Neighbouring vehicles may send a request to join the platoon and, if accepted, will receive a join response. These two messages compose the `join` mode, which enables the joiner to start receiving and sending control messages in the `platoon` mode. The latest vehicle to join the platoon may flag `isJoinable` to allow other neighbours to join, up to a certain length of the platoon. Finally, the `leave` mode is activated whenever a vehicle wishes to depart from the platoon or if no control messages have been received for a predetermined period of time. More details on the protocol are provided in Section III where we describe how it is formally modelled.

C. Protocol security verification

To formally prove security properties of a protocol such as Ensemble, there are at least three things to consider, how to model the protocol, how to model the attacker, and what security properties to verify.

There are two protocol models generally considered for creating cryptographic protocol representations: computational and symbolic [16]. In the computational model [17], [18], terms are represented as bitstrings, cryptographic primitives are functions on these bitstrings, and the adversary is any probabilistic Turing machine. In the symbolic model (which we consider in this work) bitstrings are abstracted to algebraic terms, and cryptographic primitives are represented by an equational theory. Messages are terms of these equations, for instance, consider the symmetric encryption/decryption equation where the term m is encrypted/decrypted using the key k in Equation 1.

$$sdec(senc(m,k),k) = m \tag{1}$$

The symbolic model allows the reasoning to be automated, although complex protocols usually require the solver to be guided with some proof strategy as we will discuss in the later sections of this work. Given proper heuristics, TAMARIN has been shown to work with protocols that exhibit complex state machines that may include loops and agent memory [1], [19]–[22]. For these reasons, the symbolic model and TAMARIN tool are well-suited to our work.

To represent an attacker that acts throughout the execution of the protocol, a threat model defines capabilities on computation and on observing and manipulating the network communication. A Dolev-Yao [23] threat model assumes a powerful attacker who is able to tamper with public communication channels, knows public constants, has unbounded computational and communication resources, and is able to employ cryptographic primitives as long as the required terms are known.

We now provide a brief overview of modelling in TAMARIN. Protocols and adversary actions are modelled as multiset rewriting rules and security properties are defined through (temporal) first-order logic. Rules are composed by **p**remise, **a**ction, and **c**onclusion facts as follows:

$$[p_1, ..., p_i] \; -[a_1, ..., a_j]\!\!\rightarrow \; [c_1, ..., c_k]$$

The solver maintains a multiset of facts that can be consumed as premises to activate the execution of a rule (there are also persistent facts that can be consumed an arbitrary number of times and are defined with a starting '!'). The special facts $\mathsf{In}()$ and $\mathsf{Out}()$ are used to model receiving and sending messages over the network (which can be intercepted by the attacker), and $\mathsf{Fr}()$ to generate fresh terms. In addition, the action fact $\mathsf{KU}()$ logs terms that are known by the attacker. The execution of rules creates a trace of action facts, and the security properties are formulas that reason about possible traces of the protocol. For instance, consider an action fact $\mathsf{Secret}(x)$ that marks the term x as secret whenever the corresponding rule is executed. The following formula formalises the property that an adversary cannot know a secret term x at any time point j (an action fact a that occurs at time point j is denoted as $a@j$).

$$\forall x \; i. \; \mathsf{Secret}(x)@i \Rightarrow \neg(\exists \; j. \; \mathsf{KU}(x)@j)$$

To construct traces for which formulas will be checked, TAMARIN uses a backwards constraint solving approach that checks all possible sources for a given constraint. For a more

detailed discussion and presentation of TAMARIN we refer the reader to [24]–[28].

III. PROTOCOL MODEL

This section describes our approach to analysing the standards and interpreting semi-formal descriptions to create a formal protocol model. We describe how we leverage ASN.1 (Abstract Syntax Notation One) specifications to support this process and present the resulting models (we model the platoon formation statically and dynamically), the verification goals, and assumptions we have considered.

A. Protocol messages interpretation

Some of the complexities to model protocols lie in collecting information that is scattered across different documents and connecting information that is defined sparsely, as well as interpreting possibly ambiguous specifications with regards to, for example, whether to include certain optional fields in a message. In our analysis, basic data structures and message types are defined by IEEE 1609.2, which is extended by two amendments, and ETSI 103 097 defines profiles based on these definitions. Finally, the protocol specification itself uses and extends the profiles in distinct ways (Ensemble).

We have used ASN.1 specifications that are included in several of the standards that we analyse to guide our modelling of the protocol messages. First, we collect the required modules from distinct standards (data structures and message types can be defined and imported as modules): ITS Container, CAM, IEEE 1609.2, ETSI 103 097, and Ensemble. Then, we employ an ASN.1 compiler to generate sample packets of the data structures we are interested in modelling. With the final sample packet, we can refer back to the standards so that the expected behaviour of the agents towards the data included those packets can be modelled. We refine and choose what to model in a message type if a given data structure transmits or modifies cryptographic material, and whether it affects the way the agents handle the messages (e.g., the presence of a node identity in a message can be matched with expected senders by a receiver).

In this work, the transition from ASN.1 packet descriptions to a formal model was performed manually. Given that large portions of the TAMARIN model are based on the content of messages as they are exchanged between different nodes, much of this process should be possible to automate. If properly implemented, this would significantly simplify the process of model validation.

However, even if ASN.1 formalises data structures and its encoding/decoding operations, there are no formal specifications of the sanity/security checks to be performed on received data. This makes standards susceptible to ambiguous or misinterpretations that may lead to vulnerabilities in the implementation. Therefore, even though the sample packets provide an overview of the contents of messages, it is still necessary to carefully analyse natural language in standards so that the expected behaviour of the agents towards data contained in messages is captured. Having a well-specified language for expected behaviour and security checks as part of the standardisation process would both mitigate problems related to vulnerabilities in implementations as well as improve the ability to formally verify security properties.

B. Model overview

The security verification of Ensemble was performed through two TAMARIN model variants which we define as static and dynamic. Fig. 2 depicts the main steps of the protocol operation. Our **static model** has a fixed sequence of actions from steps S_1 to S_7 between three agents that includes the announcement of a platoon, a join operation followed by a second join to form a platoon with three members, a leave from the third member and a re-keying process to update the group key. The **dynamic model** supports the same steps, however with an unlimited number of vehicles that can form an unbounded number of platoons (although with the limitation that each vehicle engages in only one session). This model enables the flexibility of different scenarios, which include unbounded number of joins, a leave from any of the followers, a key update request to the leader, and propagation of the new group key through a series of key update messages until the interaction ends. We have not analysed the growth of a platoon after a leave has occurred (however, this growth would involve the same interactions from S_2 to S_4 as shown in Fig. 2).

To support several platoons in the dynamic model, we follow the Ensemble ASN.1 definitions to include a platoon identifier in messages subsequent to the join response, and leverage such identifiers to claim the honesty of vehicles in a certain platoon (i.e., restrict the revealing of keys in the platoon).

Our models consist of rules that represent the public key infrastructure, initialisation of the vehicles and platoons, sending and receiving messages, and the revealing of keys by the attacker. The rules and properties take approximately 900 lines of text to be defined, and we use `let` bindings to make the messages more readable and clearer to maintain.

We model certificates as the persistent fact defined below, and it essentially captures the binding between an identity and a public key by a trusted certificate authority (CA with a long-term key ltk_{CA}). In our diagrams, the certificate for a public identity n is represented as $Cert_n$.

Definition 1. *A certificate for an identity n is modelled via the following persistent fact:*

$$!Cert(\langle n,\ pk(ltk_n),\ sign(\langle n,\ pk(ltk_n)\rangle,\ ltk_{CA})\rangle),$$

where $pk(ltk_n)$ is the public key for n and $sign()$ is the signing operation

We now proceed to explain the protocol model in two steps. First, we show the interaction that occurs when a new vehicle joins the platoon (Join procedure). Then we describe a full run of the protocol with two joins and one leave (Full run). The diagrams consider messages of the more expressive dynamic model, which includes platoon identifiers and vehicle position multisets.

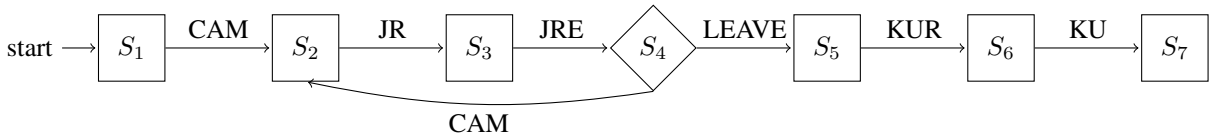

Fig. 2. Simplified diagram of protocol steps (Cooperative Awareness Message (CAM), Join Request (JR), Join Response (JRE), Leave, Key Update Request (KUR), and Key Update (KU)).

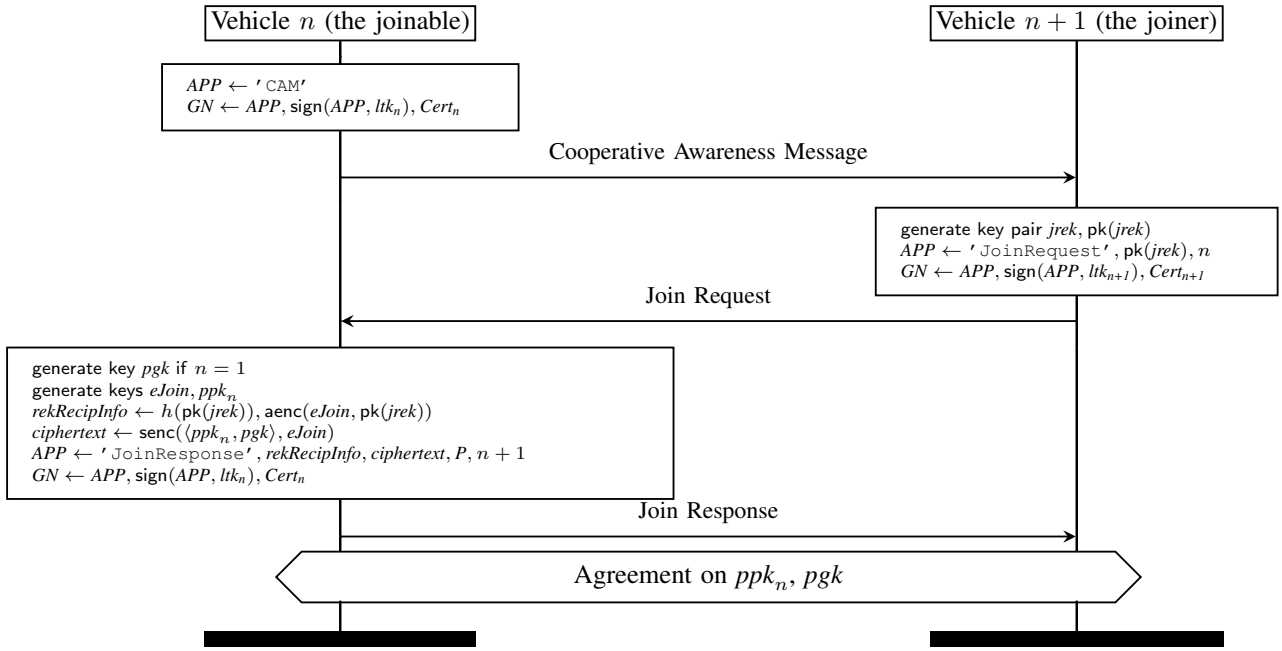

Fig. 3. The modelled Ensemble join operation.

1) Join procedure: The sequence diagram in Fig. 3 shows a `join` operation from Vehicle $n+1$ (the joiner) to Vehicle n (the joinable). The `join` begins with the joinable sending a CAM to neighbours announcing the availability to join the platoon (or to create one). The CAM is unencrypted at the application layer and follows the signing of messages in the GeoNetworking layer (all messages are signed and carry the vehicle's certificate).

Whenever the joiner receives a CAM advertising a platoon that it wishes to join, a `join request` message is prepared. The joiner generates a short-term asymmetric key pair *jrek*, $pk(jrek)$ and includes the public key in the request sent to the joinable along with the identity of the target joinable vehicle.

Once the joinable vehicle n receives the `join request`, it generates three keys: a platoon participant key (ppk_n), a platoon group key (*pgk* in case it is the first join, otherwise re-transmit previously generated), and an ephemeral join key (*eJoin*). Both ppk_n and *pgk* are encrypted with *eJoin*, and *eJoin* itself is encrypted with the public key of *jrek* which was sent by the joiner. Recall from Section II-A the use of `rekRecipInfo` when an ephemeral key is encrypted with a public key that was not obtained from a certificate.

Note that using the `rekRecipInfo` is discouraged by the security standards as it may introduce *misbinding* attacks. In Ensemble, the inclusion of an intended receiver in the `join response` mitigates such risk according to our analysis, however, we show in Section V-D the possible outcome if an intended receiver is omitted or not checked. The following is a quotation from the IEEE 1609.2 standard:

> **[IEEE 1609.2]** It is therefore recommended that secure data exchange entity designers who use public key encryption make use of either public keys in certificates or public keys in signed secured protocol data units (SPDUs), and avoid "raw" public keys because they do not mitigate this misbinding threat.

The `join response` transmits the keys along with the current platoon identifier P, the intended receiver, and the platoon position of the joiner. Note that we simplify the intended receiver and platoon position in the diagram denoted as $n+1$, whereas the model uses a public identity for the intended receiver and a multiset for the platoon position as previously discussed.

2) Full run: In a real case, there is no upper bound on the number of steps that can be taken in the Ensemble protocol. Even if there is a limit on the number of platoon members,

nodes can keep joining and leaving indefinitely. However, we consider it a full run when all message types have been sent. We now describe a scenario where two nodes join a leader node so that the platoon reaches a length of three. Once the third node has joined the platoon it initiates a leave procedure which causes a key update mechanism.

In Fig. 4 we simplify the diagram by omitting the `join` messages. In practice, each join procedure box can be interpreted as an instance of the interactions from Fig. 3. From then on, the sequence diagram represents a `leave` from Vehicle 3 and a `key update` procedure so that the remaining members agree on a new platoon group key (which we denote *pgkUpdate*).

A `leave` message contains the identity of the leaving vehicle, its position and reason to leave, and is encrypted with an ephemeral leave ($eLeave$) symmetric key. The $eLeave$ key is encrypted with the platoon group key (pgk) and is included in a `symmRecipInfo` recipient data structure. In our model, we represent the position as a multiset and reason as a constant given that no checks are performed on this term.

As soon as Vehicle 2 receives the `leave` broadcast, it prepares a `key update request (KUR)` message so that the leader instantiates a new group key. The leader proceeds to generate a $pgkUpdate$ key and includes it in a `KeyUpdate` message. Both `KUR` and `KeyUpdate` messages employ `symmRecipInfo`, however, while the former uses pgk to encrypt the ephemeral key, the latter uses ppk_n which is private to every pair of adjacent vehicles. Note that in both `KUR` and `KeyUpdate` we simply encrypt the message name constants since the contents of these messages are currently under specified in the documentation. The agreement on $pgkUpdate$ is shown at the end of the interaction in the figure.

C. Verification goals

The verification goals are divided into *liveness*, *secrecy*, and *authenticity*. Liveness ensures that our model can be executed as expected. See Section V for details of the liveness checks that are performed.

For every key (class of keys in the dynamic model) employed in the life cycle of the protocol, we introduce a secrecy verification lemma based on Definition 2. There is one important variation in the action facts and lemma construction between the static and dynamic models. Since the static model is composed of one static run, the nodes that participate in the protocol are marked as honest globally. In the dynamic model, since many platoons can be created, honesty claims are done in a per-platoon basis. This allows an attacker to compromise keys of vehicles that are not members of the platoon for which a property is being verified. In the following definitions we consider the use of a platoon identification. Our rules contain Honest(P, n) action facts that mark an identity n in a platoon P as a benign vehicle (participated in the protocol run and satisfied checks such as signature verification). Therefore, the secrecy formulas define that terms considered secret can not be deduced by an attacker unless it has revealed (through a reveal rule that contains a Rev() action fact) any subset of keys on which the secret depends. We explain such key dependency relations further in Section IV-A.

Definition 2. *Secrecy lemma with platoon identification:*

$$\begin{aligned} &\forall\ P\ x\ i.\ \textit{Secret\_key(P, x)}@i \Rightarrow \\ &\quad (\neg(\exists\ j.\ \textit{KU(x)}@j) \\ &\quad \mid (\exists\ n\ k\ r\ h.\ \textit{Rev('c', n, k)}@r\ \&\ \textit{Honest(P, n)}@h)) \end{aligned} \tag{2}$$

Intuitively, a secret x instantiated in a platoon with identifier P is either (1) not known by the attacker, or (2) an honest vehicle in the platoon P revealed a secret k of class 'c' on which x depends. We employ variations of this lemma structure to account for necessary dependencies.

For the authenticity properties we follow Lowe's hierarchy [29] and the standard formula definitions according to the TAMARIN manual. In the dynamic model, we also consider the platoon identification as part of the agreed data. Namely, we specify *aliveness*, *weak agreement*, and *non-injective agreement* properties. Since our models support a single session per vehicle, we have not analysed injective agreement properties in this work.

To verify these authenticity formulas we annotate the model rules with Running(m, n, t) and Commit(n, m, t) action facts that specify that vehicles n and m agree on their roles and the data represented by t. For a full description of the interpretation of these properties we refer to the TAMARIN manual.

D. Assumptions

In this section we describe assumptions related to the replay protection of messages, random numbers, and pattern matching used in the model.

In order to determine whether a vehicle should accept replayed messages, we have identified several places where this concept is mentioned in the relevant documents. The GeoNetworking layer could potentially be used to reject replayed messages. However, all Ensemble messages are single-hop packets which do not carry sequence numbers, so this cannot be used to prevent message replay, and ETSI 103 097 does not discuss replays. The IEEE 1609.2 standard describes a mechanism that can be used to avoid message replay attacks since it states that identical signed messages are not accepted for a given predetermined time as follows.

> **[IEEE 1609.2]** The replay detection service indicates that a signed SPDU is a replay if the entire encoded signed SPDU, including signature and other fields such as generation time inserted by the secure data service, is identical to a recently received SPDU.

Ensemble does not explicitly specify that the replay protection from IEEE 1609.2 must be enforced. Having optional security mechanisms is clearly a potential weakness since it delegates important security aspects to choices made in the

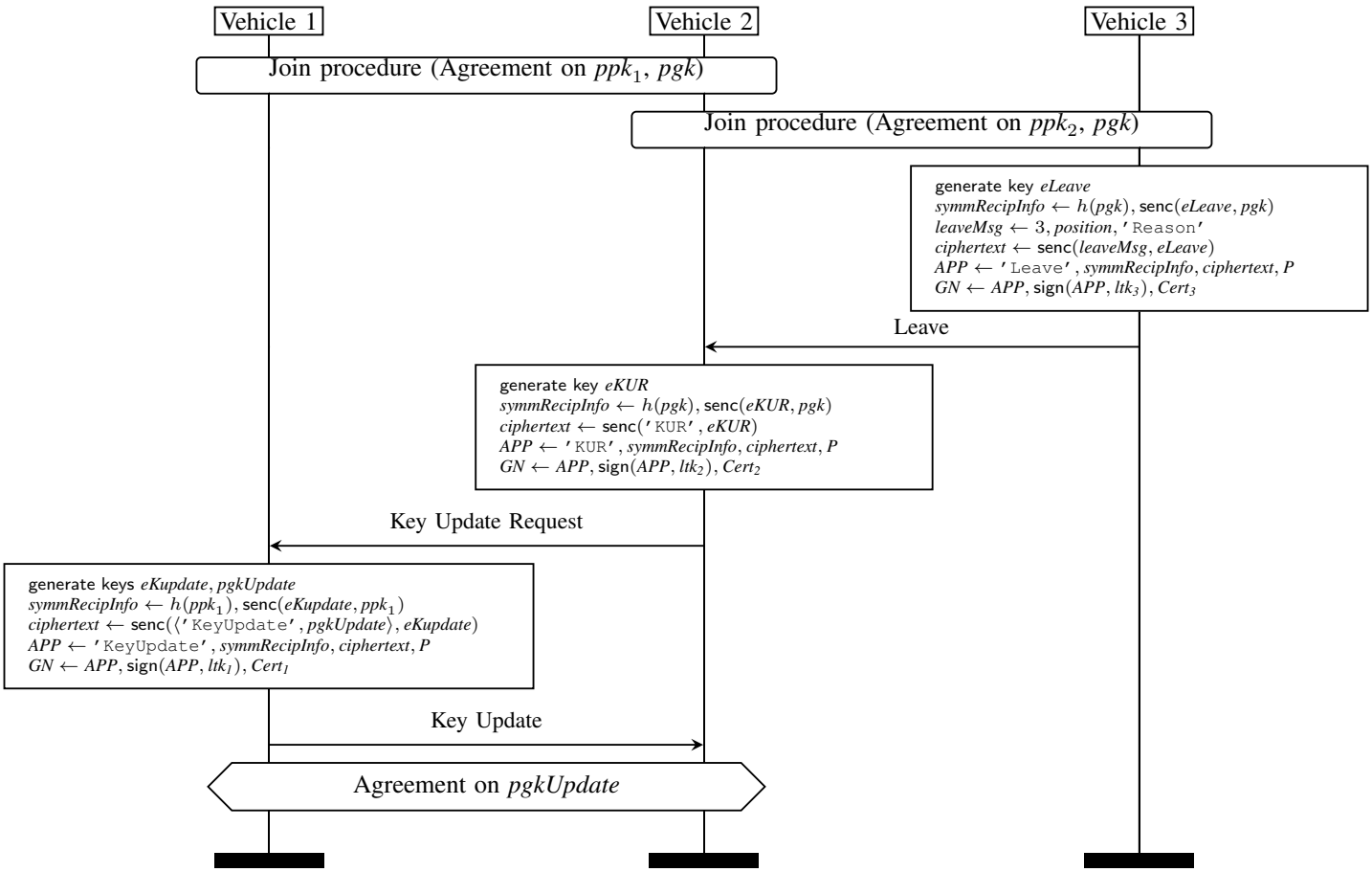

Fig. 4. The modelled Ensemble protocol interactions (the join messages from Fig. 3 are omitted).

implementation stage. Still, we believe that the most reasonable assumption according to this statement is that message replay is prevented by the protocol (valid re-transmissions of messages can be done by updating the timestamp, and agents check for uniqueness given a recentness parameter).

We formalise this replay protection as a restriction of traces (only consider those that satisfy the restriction formula) by annotating every message reception with $\mathsf{Message}(x, n)$ where x is a signed message and n is the identity of the receiver.

Definition 3. *Replay protection is modelled via the following restriction formula:*

$$\forall\ x\ n\ i\ j.\ \mathit{Message(x,\ n)}@i\ \&\ \mathit{Message(x,\ n)}@j\ \Rightarrow\ i = j$$

We include this restriction for completeness only since we have not identified any specific attack that could be launched if the restriction is not present (nor ruled it out). Such an analysis would require a model that allows entities to be present in multiple runs and is therefore out of scope for this paper.

Another assumption in the model is that vehicles verify the signature with the corresponding public key of the sender that was included in the certificate, and match the identity in the certificate with the identity stored in a state fact of that run of the protocol.

Moreover, we assume that the certificate authority is trustworthy, i.e., the attacker is unable to compromise its long-term key ltk_{CA}. In practice, one of the possibilities for the attacker would be to forge certificates with arbitrary identities and public keys in order to conduct identity theft of other vehicles.

Finally, our model assumes that fresh terms (for instance, ephemeral keys) are unique across all runs of the protocol. In addition, when receiving messages, we employ pattern matching instead of deconstruction. Deconstruction explicitly decomposes the terms by applying equations, selecting specific terms from tuples, and performing sanity checks on decomposed terms. Because of this, pattern matching implicitly checks for message formats and expected data types which must be done explicitly in real software.

IV. Proof strategy

In this section we define secrecy and authenticity relations between keys, and explain how a partial order of these

relations is employed in our proof strategy. In addition, we provide details of the goal prioritisation of the oracle that guides the constraint solver.

A. Key ordering

The combination of keys defined in Ensemble, the public key infrastructure, and ephemeral keys used in message profiles from the security standards considerably increases the complexity of our analysis. Our strategy towards making the analysis tractable is to define the relations between the keys and break the complexity into smaller parts that can be combined to prove the security properties. We first present how this can be done for a static case where all keys are known at design time, and then discuss the extension to the dynamic case where we know the classes of keys.

Let $\mathcal{K}$ be the set of symmetric and asymmetric keys. We define a secrecy dependency relation $\rightarrow \subseteq \mathcal{K} \times \mathcal{K}$ such that for two keys $k_A, k_B \in \mathcal{K}$, $k_A \rightarrow k_B$ holds if revealing the key k_B allows the attacker to learn k_A. We consider that $k_A \rightarrow k_B$ whenever $\mathsf{senc}(k_A, k_B)$ or $\mathsf{aenc}(k_A, \mathsf{pk}(k_B))$ occurs in a message sent over the network **(rule 1)**. We note that the secrecy dependency relation is reflexive (i.e., $k \rightarrow k$ for all keys k since revealing a key means that the attacker knows it). Moreover, under the assumption that revealing/compromising a key is a stateless operation (i.e., it does not otherwise change any state in the system), then the secrecy dependency relation is also transitive. This means that if $k_A \rightarrow k_B$ and $k_B \rightarrow k_C$, then $k_A \rightarrow k_C$. In most applications this relation is also anti-symmetric, thus giving rise to a partial order of keys. To define our authenticity relation, we consider *compromising* a term x as either revealing it or being able to generate a x' that will be accepted by other nodes as x. We define an authenticity dependency relation $\dashrightarrow \subseteq \mathcal{K} \times \mathcal{K}$ such that for two keys $k_A, k_B \in \mathcal{K}$, $k_A \dashrightarrow k_B$ holds if compromising k_B allows the attacker to create another key k'_A that will be accepted by the other nodes as the legitimate k_A, which thereby becomes compromised **(rule 2)**. For instance, if node n generates a fresh term f and signs it with its long-term key ltk_n, then $f \dashrightarrow ltk_n$. The authenticity dependency relation is irreflexive (knowing a long-term key does not allow creating a new long-term key), transitive (proof in appendix), and should be anti-symmetric since otherwise the protocol has a cyclic authenticity dependency.

In Ensemble, the two key relations $\rightarrow$ and $\dashrightarrow$ are both anti-symmetric (there are no cases where two different keys depend on each other). By taking the union of the two relations (a relation is a set of pairs, so the union of two relations is the aggregation of all pairs from both relations) we arrive at a third relation whose transitive closure forms a partial order $\rightsquigarrow \subseteq \mathcal{K} \times \mathcal{K}$. Intuitively, whenever $k_A \rightsquigarrow k_B$, compromising k_B will allow the attacker to compromise k_A, either directly, or through a chain of learned/replaced keys in which the attacker appears as the legitimate entity that controlled k_B. Note that the joint dependency relation can in some cases be automatically deduced from a formal description of a protocol through rules 1 and 2, and we have implemented a proof-of-concept extractor presented in the next subsection.

The dependency relation we have described here assumes a static set of keys, and also that the relation itself is time-invariant. In reality, there are several situations where these assumptions do not hold. In the case of Ensemble, the static model can be immediately analysed as all keys are known a-priori. For the dynamic model on the other hand, we have to consider *classes* of keys by, for example, considering all *ppk* keys as if they were a single key. This means that if *some ppk* key depends on another key k, then the class of *ppk* will depend on the class of k. Another situation where the assumptions do not hold is if the reveal/compromise can be limited to happen only a finite number of times. In this case, transitivity is not guaranteed to hold so compromising a top-level key does not necessarily mean that all keys "under" also become compromised. Finally, the dependency relation does not specify when a key can be compromised, so it does not account for perfect forward secrecy formulations. Taken together, the dependency graph that we consider should be seen as an abstraction in which a dependency actually means a *possible* dependency. Since the purpose of the ordering relation is to guide the prover on which lemmas/goals to prioritise, having spurious dependencies does not cause erroneous results, but can potentially reduce its usefulness. An extension of our approach would be to let the prover maintain a dynamic key hierarchy at runtime which would at least account for a changing set of keys. We discuss this further in the future work section.

B. Key dependency extractor

The dependencies formalised through rules 1 and 2 can be automatically extracted from a TAMARIN model to support oracle and reusable lemmas construction. To implement this extractor we have extended the *Tamarin to alice&bob translator* [30] in order to parse the model, extract key dependencies, identify term equivalences (for instance, keys with different names across distinct rules) through unification, and then grouping equivalent keys to output a graph of the hierarchy. The process of extracting the key dependency is summarised as follows:

1) The model is parsed to instantiate an internal representation [30] and an empty directed acyclic graph (DAG) of keys is instantiated.
2) Terms are identified in each multiset rule of the TAMARIN model and subsequently added as nodes to the DAG of keys. Relations are added as edges according to rules 1 and 2 defined in the previous subsection (duplicates may be merged through unification in later steps).
3) Instantiate empty premise and conclusion lists. Premise and conclusion facts are added to the premise and conclusion lists, respectively.
4) Facts from the premise and conclusion lists with same name and arity (as well as inputs and outputs from the network) are unified if possible (we use maude for

this [31]). This results in a set of term substitutions (unifiers) which we use to find equivalent keys.

5) A new DAG of equivalent keys is instantiated given the prior DAG of keys and the resulting sets of equivalent keys. Its topological sort represents the key ordering.

We illustrate the partial order of keys in the Ensemble protocol extracted from the dynamic model using our tool in Fig. 5. We see that at the top of this partial order is the long-term key of the certificate authority, and at the bottom are the ephemeral keys as well as the *pgkUpdate* key. Edges that are covered by the transitivity of the relations have been omitted in the graph.

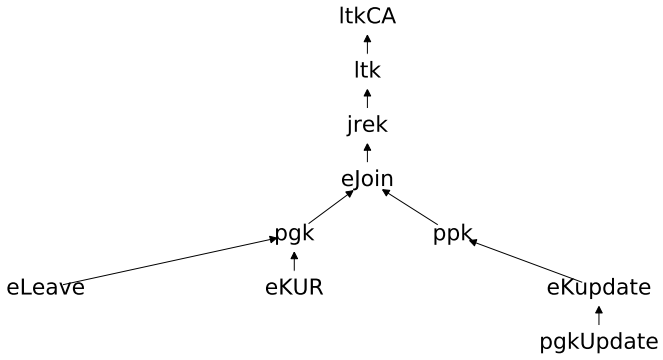

Fig. 5. Key dependency graph automatically extracted from our dynamic model.

Proving security properties involving keys at the bottom of this order requires that one is first able to prove the secrecy of the keys in the upper layers. More concretely, if $k_A \rightsquigarrow k_B$, then any security property (secrecy or authenticity) that depends on k_A also depends on k_B. Thus, one should first show the secrecy of k_B. In a simple protocol with few keys, this order matters little, but the more complex the key relationships become, the more important it is that the verifier is aware of the key ordering. Often this is implicit in the helper lemmas or oracle instrumentation made for each security property.

Due to limitations in the model parser, we remove union operations (enabled by the multiset built-in theory) from TAMARIN models and other theories are not currently supported (such as XOR and Diffie-Hellman). The relations for such theories could be derived from the message deconstruction rules used in TAMARIN, but are out of the scope of our current analysis. In addition, we note that some models might generate cyclic graphs which are currently not supported.

Despite these limitations, we have employed our proof-of-concept tool in the analysis of a recent work in 5G handover protocols [32]. These models also present a complex relation on keys, and required the modellers to consider the dependencies during the specification of lemmas and oracles. For instance, given that the model allows the revealing of some keys, the secrecy lemmas must restrict the revealing of dependencies. Furthermore, the oracles also seem to consider some dependencies in the prioritisation, similar to what we present in this work. While we have not enhanced the proving efficiency of these models that had already been carefully optimised by experts, we argue that applying the strategies described in this work could be useful during the analysis and modelling of such protocols. Examples of the extracted graphs and further discussions are included in Appendix B.

C. Inductive helper lemmas

The possibility to instantiate infinitely many vehicles and platoons associated to the fact that each platoon can grow indefinitely aggregates further complexity to our dynamic model. The model therefore enables loops that result in non-termination when using standard backwards search in TAMARIN. To handle such behaviour, TAMARIN allows the specification of inductive lemmas which we employ as intermediate helpers.

Recently, Cremers et al. have analysed IEEE 802.11's WPA2 protocol [22], which also contains complex state machines with loops and evolving states. In their work, they specify Wellfoundedness, Uniqueness, and Ordering lemmas. We follow their approach in the creation of such inductive intermediate helper lemmas, and in conjunction employ our key hierarchy prioritisation.

D. Oracle strategy

We now describe how the use of the linear extension of the partial order of keys is used to guide the solver. Recall from Section II-C that TAMARIN checks possible sources for constraints to generate traces that will be used to prove or find a counterexample for a given property. The choice of which constraint to solve (goal) at a given step can be tailored by using *oracles*. To perform the Ensemble verification, we structured the lemmas and developed an oracle so that they leverage the key ordering from our key dependency extractor.

Due to space constraints, we only present the main prioritisation activities performed by the oracle, which leverage the selection of reusable lemmas to avoid unnecessary case distinctions during proving.

Ordered helper lemmas iff a knowledge goal for the corresponding key exists in the constraint system: In order to create contradictions earlier, the helpers are prioritised if there is currently a goal for an attacker knowledge of the corresponding key k as $\mathsf{KU}(k)$. Algorithm 1 performs this prioritisation. The algorithm runs for each k according to the ordering of keys.

Signature of protocol messages: Following our hierarchical key approach, we introduce helper lemmas that prove that the attacker cannot obtain any long-term key unless it performs a reveal of those keys. Since the authenticity of messages depend on the secrecy of the respective long-term keys (because of the signatures), we prioritise these goals to determine that the attacker is not able to forge signatures or act on behalf of an honest node.

Data: Proof goals $\mathcal{G}$; Linearised key ordering $\mathcal{K}^{\rightsquigarrow}$
Result: Ordered list of goals $\mathcal{G}^{\rightsquigarrow}$
foreach $k \in \mathcal{K}^{\rightsquigarrow}$ **do**
  foreach g *in* $\mathcal{G}$ **do**
    if g *is a helper lemma for* k *and* $KU(k) \in \mathcal{G}$ **then**
      add g to $\mathcal{G}^{\rightsquigarrow}$
    end
  end
end

Algorithm 1: Oracle priority pseudocode for helper lemmas

V. Results and discussion

In this section we present the Ensemble verification results using our TAMARIN model variants and their respective proven properties. In addition, an evaluation of our proof strategy is conducted in order to show that leveraging an order-aware oracle is effective.

A. Security verification results

Considering the security verification goals described in Section III-C we prove three kinds of properties, model liveness, secrecy and authenticity. For each of these we describe the resulting security lemmas in the context of our model.

a) Liveness: To ensure protocol executability in our static variant, we prove that a full run with three vehicles exists. In the dynamic variant we verify that the following is possible: two platoons can be formed with four members each, there exists a leave from a member in a platoon, as well as a key update request and key updates are performed for remaining members.

b) Secrecy: We prove secrecy of all long-term and short-term keys. We use one lemma per key in the static model as they are instantiated in distinct rules, so there is a total of 15 secrecy lemmas. The dynamic model contains one secrecy lemma for every class of keys (e.g., one lemma proves the secrecy of all platoon participant keys).

c) Authenticity: We have one lemma each for the authenticity properties aliveness, weak agreement and non-injective agreement (cf. Section III-C).

The properties have been proven for both static and dynamic models. This required making use of all the verification strategies described in Section IV.

B. Verification strategy evaluation

To assess the effectiveness and impact of the proof strategies, we use two experiments: a synthetic protocol generator and a variation of configurations for proving the static model. The experiments are run on a cluster of the Swedish National Supercomputer Centre, where each compute node is equipped with Intel(R) Xeon(R) Gold 6130 CPU @ 2.10GHz with 32 cores and 96 GiB of main memory.

The synthetic protocol consists of a simple 'ping-pong' protocol in which two nodes communicate and in every interaction instantiate a new symmetric key which is encrypted with the previously received key (the first instance is derived from a pre-shared key). We use the standard TAMARIN heuristics and only provide annotations in the model to prioritise certain facts (state, pre-shared key, symmetric encryption, and attacker knowledge of secret). The lemmas are created according to the linear dependency of keys, and the proving is evaluated with and without the reuse of lemmas. In addition, we perform an experiment with the reuse of lemmas that are randomly ordered (for this, the prover is executed ten times with random orders for each key depth).

Each run was granted 8 cores of CPU for 30 minutes and 20 Gib of RAM. Table I presents the results. TAMARIN is able to automatically prove the secrecy of keys in a depth of 2 in all cases, and up to a depth of 8 when reusing ordered lemmas according to the key hierarchy, which shows how our strategy makes such proofs tractable. In some cases, the random ordering resulted in a possible automatic proof, but took significantly more time to terminate as it was not the optimal order. The key depth of 10 could not be proven and would require further manual intervention (e.g., through an oracle).

TABLE I
RESULTS OF PROVING THE SYNTHETIC MODELS WITH DISTINCT KEY DEPENDENCY DEPTH

Key depth	Without reuse (ordered by dependency)	With reuse (random order)	With reuse (ordered by dependency)
2	✔	✔	✔
4	✖	3/10	✔
6	✖	2/10	✔
8	✖	✖	✔
10	✖	✖	✖

In our second set of experiments, we run TAMARIN on the static model of Ensemble with distinct parameter configurations and measure how many of the security lemmas can be proven with these settings and the computational resources that are used.

We run the prover with four configurations, outlined as follows.

- **Bare TAMARIN** - In this configuration we try to prove the security properties of the protocol without any added proof strategies or helper lemmas.
- **Lemma reuse** - Lemmas that for example assert secrecy of keys are set as reusable so that the verifier can assume these lemmas to be true when searching for the proof.
- **Oracle only** - Here we make use of the order-aware oracle but do not reuse lemmas, and must therefore reprove all relevant subresults for every property.
- **Order-aware** - In this configuration we use both the order-aware oracle and reuse helper lemmas.

The first two configurations should be considered as baselines. The reason for including both the "Oracle only" and "Order-aware" configurations rather than just a single good strategy is to investigate the relative impact of the different aspects of the generated oracle as the ordering of reusable

lemmas according to the key hierarchy is an important aspect of its design (see Section IV-D).

In addition to the amount of successfully proven lemmas, we measure the resource consumption in terms of computation time and memory usage. Each lemma was run as a separate job in the cluster, and given an allocation of 2 hours on 8 cores and 22 GiB of memory. Jobs that exceeded either the time or memory limit were aborted.

C. Effectiveness of proof strategies

We now proceed to present the outcome of the second experiments. Table II shows an overview of how the four prover configurations performed in terms of proving the 20 security properties. The Oracle and Reuse columns summarise the key differences between the configurations (with or without the key-aware oracle, and with or without reusable lemmas). The final three columns show how many of the lemmas that could be proven in the three categories.

TABLE II
OVERVIEW OF PROVABILITY FOR DISTINCT PROOF STRATEGIES

Proof method	Oracle	Reuse	Liveness	Secrecy	Authenticity
Bare TAMARIN	N	N	1/1	0/15	0/3
Lemma reuse	N	Y	1/1	4/15	1/3
Oracle only	Y	N	1/1	15/15	3/3
Order-aware	Y	Y	1/1	15/15	3/3

The results clearly demonstrate the effectiveness of the key-aware oracle, which seems to be the deciding factor to making the model tractable for the verifier.

Another perspective on the performance of the strategies is shown in Fig. 6. The graph shows time on the X axis (logarithmic scale) and the number of lemmas proven within this time on the Y axis. There is a significant (and expected) performance difference observed when making use of, and ordering, previously proven lemmas. In particular, the fastest 15 lemmas were verified in 86 seconds by the order-aware strategy whereas it took over 10 minutes when lemmas were not reused (Oracle only).

D. Identity misbinding attack

An *identity misbinding attack* [33], [34], also referred to as an *unknown key-share attack* [35], [36] occurs when two honest parties establish a common session key without a consistent view of each other's identities. In IEEE 1609.2, the use of `rekRecipInfo` can possibly create a vulnerability to misbinding since the public key is not bound to an identity. In Ensemble, even though this data structure is used in a *Join Response*, misbinding can be mitigated because the identity of the intended receiver is included in the application payload. However, it is not stated explicitly that this information should be validated. The receiver must check that the intended receiver included in the message matches its own identity. In an implementation of the protocol where this is not done, some security properties will be violated. We ran the verification with a modified variant of the static model that captures this aspect. The results can be seen in Table III where we see that

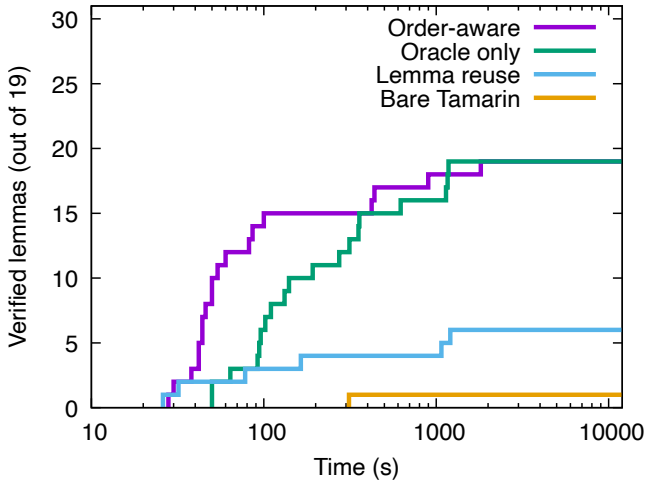

Fig. 6. The number of lemmas proven within a given time for the proof strategies (logarithmic x axis).

TABLE III
AUTHENTICATION FAILURE IN THE ABSENCE OF INTENDED RECEIVER

Security property	Intended receiver	No intended receiver
Aliveness	✔	✔
Weak agreement	✔	✖
Non-injective agreement	✔	✖

the weak and non-injective agreement authenticity properties are not satisfied.

The possible attacker behaviour is described as follows (see also Fig. 3). Provided that the check of intended receiver is absent, an inside attacker (who has a valid long-term key and certificate) could replay the CAM (from Vehicle 1) that advertises a joinable platoon to another Vehicle 2, which will send a `join request`. The attacker extracts the public key of *jrek* from that message and uses it in a join request signed with his own long-term key. Vehicle 1 will send a `join response` to the attacker, which will transmit it back to Vehicle 2. At the end of the procedure, Vehicle 1 believes that the attacker has joined, whereas Vehicle 2 believes it has joined Vehicle 1, and both share the participant and group keys (note that the attacker can not compromise the secrecy).

VI. RELATED WORK

Vehicular network security standardisation and its formal analysis is rather recent. Whitefield et al. [37] analyse V2X certificate revocation of malicious or misbehaving vehicles with the REWIRE scheme using TAMARIN. In their analysis, they are able to identify an authentication weakness and propose an extension to mitigate it. Li et al. propose a lightweight privacy-preserving authentication protocol that is verified with BAN logic and PROVERIF [38].

In mobile networks, Basin et al. [1] formalise the 5G authentication and key agreement protocol, and verify security properties using TAMARIN. The authors found in their analysis

that security goals and assumptions were under-specified or missing. We show a similar situation in our analysis of standards in the vehicular domain. While data structures are often well defined, under-specification of data checks and behaviour can lead to misinterpretation and potential security vulnerabilities.

With respect to verification and solving theory, Cremers and Mauw [39] employ partial order reduction to lower the number of traversed states in checking secrecy of terms in a cryptographic protocol in their tool SCYTHER. They build on the fact that exchanging two events in a trace might result in equivalent traces with respect to the verified property. In our work, we explore the fact that solving for the knowledge of some terms might not be relevant, and that solving for the knowledge of some terms before others is more efficient.

Schmidt et al. [40] develop an algorithm to verify protocol group key agreement protocols that can handle Diffie-Hellman exponentiation, bilinear pairing, and AC-operators. In their work they extend the operators set and provide constraint reduction rules in TAMARIN to support them. They argue for the analysis of dynamic join and leave operations in group protocols, which is also present in our model.

VII. Conclusion and Future Work

We have formally analysed the security of Ensemble, a protocol for vehicular group formation with key establishment and distribution which is currently in pre-standardisation. To conduct the verification, we define secrecy and authentication relations that are applied in a proof strategy based on their partial order. We automate the key hierarchy extraction from our TAMARIN models and create oracles to guide the prover based on the ordering of keys. To refine the model of the protocol messages we use ASN.1 definitions from standards and a compiler to generate sample packets, which was useful to avoid misinterpretations or ambiguities from multiple documents. Through our assessment of vehicular network security standards by IEEE and ETSI, we show that although they provide solid security message formats, the implementations may still be susceptible to weaknesses if the expected agent behaviour is not enforced. We show an example of such a weakness in the form of a misbinding attack when appropriate checks are not performed by the vehicles. An interesting point for discussion in the context of standardisation work lies in formally describing agent behaviour towards received data and appropriate security checks.

TAMARIN enables the formal analysis of several complex protocols, and may require manual tuning in some cases. We believe that, ideally, an automated security analysis should be able to derive, without manual intervention, the set of conditions for each cryptographic term in a protocol to remain secret and provide the corresponding proof (currently, modellers must identify such conditions and specify them in the lemmas). Our work to automatically extract key dependencies is a step towards this long-term goal, and many interesting challenges remain. An integration of the dependency analysis in TAMARIN at runtime (during proving) would allow much richer reasoning. For instance, this could allow the possibility to consider time and properties that involve forward secrecy. In addition, the extension of dependency relations to account for XOR, Multiset, Diffie-Hellman, and other equational theories are required to support a large class of models. We consider these challenges to be important contributions in future work.

References

[1] D. Basin, J. Dreier, L. Hirschi, S. Radomirovic, R. Sasse, and V. Stettler, "A formal analysis of 5g authentication," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 1383–1396. [Online]. Available: https://doi.org/10.1145/3243734.3243846

[2] L. C. Paulson, "Relations between secrets: two formal analyses of the yahalom protocol," *Journal of Computer Security*, 2001.

[3] D. Pavlovic and C. Meadows, "Deriving secrecy in key establishment protocols," in *Proceedings of the 11th European Conference on Research in Computer Security*, ser. ESORICS'06. Berlin, Heidelberg: Springer-Verlag, 2006, p. 384–403. [Online]. Available: https://doi.org/10.1007/11863908_24

[4] J. D. Guttman and F. Thayer, "Authentication tests and the structure of bundles," *Theoretical Computer Science*, vol. 283, no. 2, pp. 333–380, 2002, theoretical Foundations of Security Analysis and Design. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0304397501001396

[5] S. Meier, B. Schmidt, C. Cremers, and D. Basin, "The tamarin prover for the symbolic analysis of security protocols," in *Computer Aided Verification*, N. Sharygina and H. Veith, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 696–701.

[6] Ensemble, "D2.8 Platooning protocol definition and communication strategy," 2018.

[7] Ensemble, "D2.9 Security framework of platooning," 2019.

[8] 3GPP, "TR 21.915 Summary of Rel-15 Work Items (Release 15), v15.0.0," 2019.

[9] ETSI, "TS 102 636-4-3 Media-dependent functionalities for LTE-V2X, v1.1.1," 2020.

[10] ——, "EN 302 636-4-1 GeoNetworking; Part 4: Geographical addressing and forwarding for point-to-point and point-to-multipoint communications; Sub-part 1: Media-Independent Functionality, v1.4.0," 2019.

[11] ——, "EN 302 636-5-1 Basic Transport Protocol, v2.2.0," 2019.

[12] IEEE, "Std 1609.2 Security Services for Applications and Management Messages," 2016.

[13] IEEE, "Std 1609.2a Security Services for Applications and Management Messages - Amendment 1," 2017.

[14] ——, "Std 1609.2b Security Services for Applications and Management Messages - Amendment 2: PDU Functional Types and Encryption Key Management," 2019.

[15] ETSI, "TS 103 097 Security header and certificate formats, v1.3.1," 2017.

[16] B. Blanchet, "Security protocol verification: Symbolic and computational models," in *Principles of Security and Trust*, P. Degano and J. D. Guttman, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 3–29.

[17] S. Goldwasser, S. Micali, and R. L. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," *SIAM J. Comput.*, vol. 17, no. 2, p. 281–308, Apr. 1988. [Online]. Available: https://doi.org/10.1137/0217017

[18] S. Goldwasser and S. Micali, "Probabilistic encryption," *Journal of Computer and System Sciences*, vol. 28, no. 2, pp. 270 – 299, 1984.

[19] C. Cremers, M. Horvat, S. Scott, and T. van der Merwe, "Automated analysis and verification of tls 1.3: 0-rtt, resumption and delayed authentication," in *2016 IEEE Symposium on Security and Privacy (SP)*, 2016, pp. 470–485.

[20] R. Künnemann, "Automated backward analysis of pkcs#11 v2.20," in *Principles of Security and Trust*, R. Focardi and A. Myers, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 219–238.

[21] C. Cremers, M. Dehnel-Wild, and K. Milner, "Secure authentication in the grid: A formal analysis of dnp3: Sav5," in *Computer Security – ESORICS 2017*, S. N. Foley, D. Gollmann, and E. Snekkenes, Eds. Cham: Springer International Publishing, 2017, pp. 389–407.

[22] C. Cremers, B. Kiesl, and N. Medinger, "A formal analysis of {IEEE} 802.11's wpa2: Countering the kracks caused by cracking the counters," in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 2020, pp. 1–17.
[23] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 198–208, 1983.
[24] S. Meier, "Advancing automated security protocol verification," Ph.D. dissertation, ETH Zurich, Zürich, 2013.
[25] B. Schmidt, "Formal analysis of key exchange protocols and physical protocols," Ph.D. dissertation, ETH Zurich, Zürich, 2012.
[26] J. Dreier, L. Hirschi, S. Radomirovic, and R. Sasse, "Automated unbounded verification of stateful cryptographic protocols with exclusive or," in *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, 2018, pp. 359–373.
[27] V. Cortier, S. Delaune, and J. Dreier, "Automatic generation of sources lemmas in tamarin: Towards automatic proofs of security protocols," in *Computer Security – ESORICS 2020*, L. Chen, N. Li, K. Liang, and S. Schneider, Eds. Cham: Springer International Publishing, 2020, pp. 3–22.
[28] J. Dreier, L. Hirschi, S. Radomirović, and R. Sasse, "Verification of Stateful Cryptographic Protocols with Exclusive OR," *Journal of Computer Security*, vol. 28, no. 1, pp. 1–34, Feb. 2020. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02358878
[29] G. Lowe, "A hierarchy of authentication specifications," in *Proceedings 10th Computer Security Foundations Workshop*, 1997, pp. 31–43.
[30] D. Kozmai, "Converting tamarin to extended alice&bob protocol specifications," Bachelor's Thesis, ETH, Zürich, 2016.
[31] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martı-Oliet, J. Meseguer, and J. Quesada, "Maude: specification and programming in rewriting logic," *Theoretical Computer Science*, vol. 285, no. 2, pp. 187–243, 2002, rewriting Logic and its Applications. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0304397501003590
[32] A. Peltonen, R. Sasse, and D. Basin, "A comprehensive formal analysis of 5g handover," in *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, ser. WiSec '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 1–12. [Online]. Available: https://doi.org/10.1145/3448300.3467823
[33] H. Krawczyk, "Sigma: The 'sign-and-mac' approach to authenticated diffie-hellman and its use in the ike protocols," in *Advances in Cryptology - CRYPTO 2003*, D. Boneh, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 400–425.
[34] W. Diffie, P. C. Van Oorschot, and M. J. Wiener, "Authentication and authenticated key exchanges," *Designs, Codes and cryptography*, vol. 2, no. 2, pp. 107–125, 1992.
[35] B. S. Kaliski Jr, "An unknown key-share attack on the mqv key agreement protocol," *ACM Transactions on Information and System Security (TISSEC)*, vol. 4, no. 3, pp. 275–288, 2001.
[36] S. Blake-Wilson and A. Menezes, "Unknown key-share attacks on the station-to-station (sts) protocol," in *Public Key Cryptography*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 154–170.
[37] J. Whitefield, L. Chen, F. Kargl, A. Paverd, S. Schneider, H. Treharne, and S. Wesemeyer, "Formal analysis of v2x revocation protocols," in *Security and Trust Management*, G. Livraga and C. Mitchell, Eds. Cham: Springer International Publishing, 2017, pp. 147–163.
[38] X. Li, T. Liu, M. S. Obaidat, F. Wu, P. Vijayakumar, and N. Kumar, "A lightweight privacy-preserving authentication protocol for vanets," *IEEE Systems Journal*, vol. 14, no. 3, pp. 3547–3557, 2020.
[39] C. J. F. Cremers and S. Mauw, "Checking secrecy by means of partial order reduction," in *System Analysis and Modeling*, D. Amyot and A. W. Williams, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 171–188.
[40] B. Schmidt, R. Sasse, C. Cremers, and D. Basin, "Automated verification of group key agreement protocols," in *2014 IEEE Symposium on Security and Privacy*, 2014, pp. 179–194.

APPENDIX A
AUTHENTICITY RELATION

In this appendix we prove that the authenticity relation defined in Section IV-A is transitive. We recall the following definitions:

- *Compromising* a term p means either revealing it or being able to generate a p' that will be accepted by other nodes as p. We assume that both of these actions (revealing and generating a new key) can be performed infinitely often.
- If $\mathcal{K}$ is a set of keys, then the authenticity dependency relation $\dashrightarrow \subseteq \mathcal{K} \times \mathcal{K}$ is is defined so that for two keys $k_A, k_B \in \mathcal{K}$, $k_A \dashrightarrow k_B$ holds if compromising k_B allows the attacker to create another key k'_A that will be accepted by the other nodes as the legitimate k_A.

We now proceed to prove that $\dashrightarrow \subseteq \mathcal{K} \times \mathcal{K}$ is transitive.

Proof. Assume that there exists keys $k_A, k_B, k_C \in \mathcal{K}$ such that $k_A \dashrightarrow k_B$ and $k_B \dashrightarrow k_C$. To prove transitivity, we then must show that $k_A \dashrightarrow k_C$. Assume that k_C has been compromised, then by the second definition above, the attacker can create another key k'_B that will be accepted by other nodes as k_B. By the first definition, this means that k_B is compromised. Since k_B is compromised and $k_A \dashrightarrow k_B$, then the attacker can create a key k'_A that will be accepted by other nodes as k_A. Thus $k_A \dashrightarrow k_C$. □

APPENDIX B
5G HANDOVER GRAPHS

In this appendix we present approximations of the key dependencies extracted automatically with our tool from a 5G handover model [32]. In addition to the steps described in Section IV-B, a custom key derivation function (KDF with arity 2) present in the 5G handover models must be considered. Given a term $k = \text{KDF}(a, b)$, then it holds that both a and b must be known by an attacker in order to construct k. This type of conjunctive dependency is not supported by our current dependency relation, hence we approximate it in a pessimistic (but safe) manner by creating two separate secrecy dependencies $k \rightarrow a$ and $k \rightarrow b$. Intuitively, we state that an attacker could construct k by learning either a or b, whereas in reality it must know both terms. This approximation (in addition to our secrecy and authentication relations presented in this work) of the (N2-based inter-RAN) 5G handover model resulted in the graph illustrated in Fig. 7, which is unlabelled for simplifying the presentation.

In order to give a concrete example of the structure of this relation, we present in Fig. 8 a subgraph of Fig. 7 which includes the all dependencies originating from a chosen term 'K-AMF3'. The secrecy of derived 'K-AMF' keys is one of the verified properties of the 5G handover analysis. A 'K-AMF' can either be derived directly from keys 'SUPI' and 'K-SEAF', or from another 'K-AMF' itself (we refer the reader to the paper [32] for more details). Fig. 8 shows classes of keys (there are several distinct classes of 'K-AMF') and their relations. In addition to the dependencies described earlier, the graph also includes SUPI $\rightarrow$ sk-HN due to the asymmetric encryption of 'SUPI' with the public key of the home network pk(sk-HN).

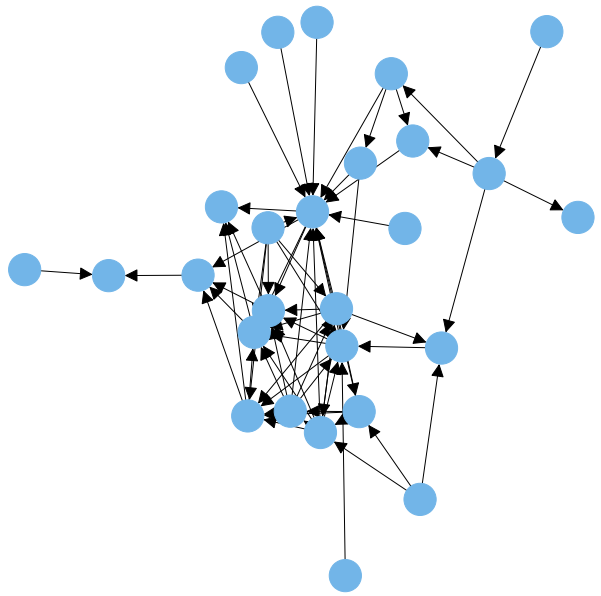

Fig. 7. Approximation of the dependency graph from the N2-based inter-RAN variant of 5G handover.

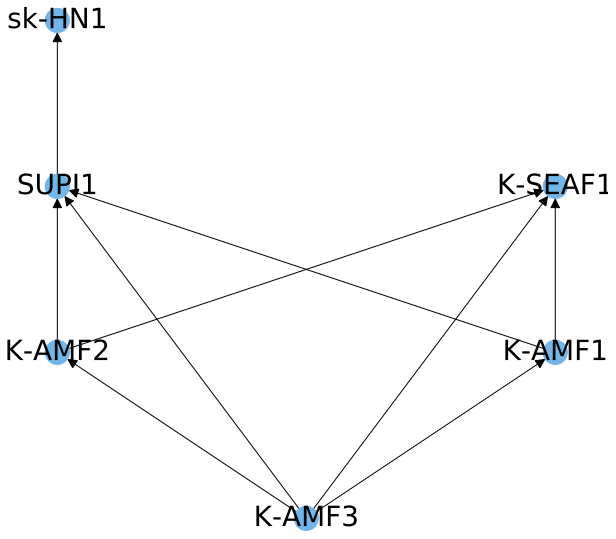

Fig. 8. Subgraph of dependencies extracted for target secret 'K-AMF3'.