Efficient and Timely Revocation of V2X Credentials

Gianluca Scopelliti
Ericsson Security Research
DistriNet, KU Leuven
gianluca.scopelliti@ericsson.com

Christoph Baumann
Ericsson Security Research
christoph.baumann@ericsson.com

Fritz Alder
DistriNet, KU Leuven
fritz.alder@acm.org

Eddy Truyen
DistriNet, KU Leuven
eddy.truyen@kuleuven.be

Jan Tobias Mühlberg
Université Libre de Bruxelles
DistriNet, KU Leuven
jan.tobias.muehlberg@ulb.be

***Abstract*—In Intelligent Transport Systems, secure communication between vehicles, infrastructure, and other road users is critical to maintain road safety. This includes the revocation of cryptographic credentials of misbehaving or malicious vehicles in a timely manner. However, current standards are vague about how revocation should be handled, and recent surveys suggest severe limitations in the scalability and effectiveness of existing revocation schemes. In this paper, we present a formally verified mechanism for self-revocation of Vehicle-to-Everything (V2X) pseudonymous credentials, which relies on a trusted processing element in vehicles but does not require a trusted time source. Our scheme is compatible with ongoing standardization efforts and, leveraging the Tamarin prover, is the first to guarantee the actual revocation of credentials with a predictable upper bound on revocation time and in the presence of realistic attackers. We test our revocation mechanism in a virtual 5G-Edge deployment scenario where a large number of vehicles communicate with each other, simulating real-world conditions such as network malfunctions and delays. Results show that our scheme upholds formal guarantees in practice, while exhibiting low network overhead and good scalability.**

I. Introduction

Intelligent Transport Systems (ITS) are a key technology for innovation in transportation that link different modes of transportation and address safety, congestion, emissions, and efficiency problems. Increased connectivity in vehicles, called Vehicle-to-Everything (V2X) communication, incorporates digital interactions with other road users with potentially safety-critical impacts on the vehicle and its passengers. As highlighted by Sedar et al. [40] there are many possible incentives for attackers to target V2X systems, which range from personal interest (e.g. tracking or insuarance fraud) to geopolitical conflict (e.g. causing congestion or accidents).

A large body of academic research and technical standards for V2X communication focuses on privacy-preserving protocols to enroll and manage pseudonymous identities [46], [12], [30], [19], [20], [6]. The key argument for such protocols is that special care must be taken to ensure vehicle routes and patterns in their use of V2X services must not be observable by other V2X participants in the network. Yet, the central concern in transportation has always been road safety. In V2X scenarios, road safety requires that misbehaving network participants, e.g., faulty or malicious vehicles that report inaccurate road and traffic conditions, can be identified and prevented from causing harm. For example, the US Security Credential Management System (SCMS) defines revocation *"an essential design objective"* [6].

In a practical working system, it must be possible to 1) detect, and 2) punish such misbehaving participants by removing them from the network. We identify a common shortcoming in existing privacy-preserving V2X approaches, with respect to their capability to punish malicious or misbehaving actors after misbehavior connected to a participant's pseudonym is detected: Ensuring that this participant will not be able to communicate with other vehicles in the future and effectively *revoking* all their existing credentials in a timely manner is either not guaranteed and avoidable by the attacker, or is inefficient and not scalable due to bandwidth and computational overheads. Indeed, recent surveys [48], [45] highlight that existing schemes based on active [6] or passive [20] revocation suffer from scalability issues, low reactiveness, or both.

To address these challenges, we build upon decentralized revocation schemes based on self-revocation [46], [12], [30], which utilize a Trusted Component (TC) embedded in vehicles for secure credential management. Self-revocation enables vehicles to self-certify and also self-revoke pseudonymous credentials, obsoleting most of the centralized public key infrastructure of current ITS designs. This paper addresses a number of shortcomings of previous work, summarized by the following contributions:

1) We design a secure self-revocation scheme that, in contrast to earlier work, can guarantee revocation with an upper bound on revocation time, without relying on a trusted time source in TCs (Sect. V);
2) We formalize and verify our design using the Tamarin prover [34], showing that it provides strong guarantees on the revocation of credentials even in presence of a

Network and Distributed System Security (NDSS) Symposium 2024
26 February - 1 March 2024, San Diego, CA, USA
ISBN 1-891562-93-2
https://dx.doi.org/10.14722/ndss.2024.24017
www.ndss-symposium.org

powerful adversary who drops and/or delays revocation requests (Sect. VI);
3) We evaluate the security, performance, and scalability of our design, showing that our approach guarantees prompt revocation and low utilization of resources, even in the presence of network malfunctions and a high number of attackers, in difference to related work (Sects. VII and VIII).
4) We discuss applicability of our revocation scheme to state-of-the-art V2X protocols, showing that it is compatible with current standards in the EU and US, as well as Direct Anonymous Attestation (DAA) (Sect. VIII-B).

We provide the full formal specifications and an implementation of our design with this publication [39]. The most recent version of this artifact is also available on GitHub[1].

II. Background

In the EU, performance and security requirements for ITS and advanced use cases (such as vehicle platooning or remote driving) are laid out in [15] and [17]. For example, it is specified that ITS need to support "high connection density for congested traffic" with "3,100 to 4,300 cars per square kilometer." Vehicles in these scenarios communicate at rates of up to 50 messages/s with an end-to-end latency of no more than 20 ms (platooning), or even via continuous data streams of up to 25 MBit/s with an end-to-end latency of no more than 5 ms (remote driving) [17].

To maintain security in these scenarios, infrastructure and vehicles must implement efficient protocols that facilitate credential management, including revocation, and signing and verifying message content while upholding privacy objectives and legal requirements on data protection. As summarized by Hicks and Garcia [25], the requirements for the management of V2X credentials involve communicating parties to verify the authenticity of messages, guarantee unlinkability, minimize the repercussions of a compromise of central trust management systems, and enable revocation of vehicles upon the detection of policy violations, malfunction, or malicious interaction. Therefore, vehicles must be capable to timely manage the validity of credentials, which involves changing their own pseudonym certificates and validating those of communicating parties. Importantly, communications and security management protocols must also account for extended periods without connectivity, where a vehicle is unreachable or powered off.

Certificate Management and Revocation. In Europe, the architecture for credential management in V2X is defined by ETSI [19], [20], while the US follows the SCMS design [6]. In both architectures, ITS participants possess two types of credentials: 1) *long-term credentials*, for authenticating a vehicle towards the ITS infrastructure, and 2) *pseudonym credentials* (*pseudonyms* in short), for secure and privacy-preserving peer-to-peer communication, such as Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I), or Vehicle-to-Pedestrian (V2P). Long-term credentials can be either private keys stored in secure elements of vehicles at manufacture time, or certificates obtained during enrollment with the ITS. Pseudonyms are typically retrieved from the infrastructure upon successful authentication and authorization of a vehicle (via its long-term credential), and are usually short-lived (with lifetime of up to weeks [16]) and rotated regularly to prevent vehicle tracking.

Revocation aims at excluding a participant from interacting with others. We distinguish between *identity-based revocation* and *message-based revocation*, which differ in the way that the decision to revoke is made: The former is based on a vehicle's *canonical identity*, and is necessary to enforce legal/policy requirements (e.g., to have only insured vehicles on the road), in case of private-key compromise (e.g., a leak), or when a vulnerability affecting a certain component or vehicle vendor becomes public, requiring revocation of multiple vehicles at once. The latter, which is the focus of our work, is based on peer-to-peer traffic, where a misbehaviour detection system is in place to report faulty or malicious messages to a Revocation Authority (RA), which eventually mandates the revocation of the vehicle that signed such messages. When a vehicle gets revoked, its long-term credentials are blacklisted by the ITS in order to deny it future requests, e.g., for re-enrollment or for obtaining new pseudonyms. Additionally, all current vehicle's pseudonyms should be made unusable, to prevent the transmission of malicious messages to other peers. In this regard, SCMS and ETSI use two opposing strategies, i.e., active and passive revocation, respectively (cf. Sect. III).

Trusted Execution and DAA. A common approach to establish a root of trust in distributed systems is by relying on Trusted Components (TCs), such as Trusted Platform Modules (TPMs) or Trusted Execution Environments (TEEs) [33], which provide the means to securely store and use long-term credentials, perform key generation, remote attestation, and sealing. Several papers propose the use of trusted components inside vehicles to manage cryptographic credentials and increase security and privacy [23], [44], [28]. Based on the security features provided by TCs, Direct Anonymous Attestation (DAA) has been developed as a privacy-preserving protocol to remotely authenticate a system [7], [26]. In difference to ongoing standardization efforts in ITS and SCMS, which rely heavily on centralized public key infrastructure [19], [20], [6], the application of DAA in an ITS context obsoletes most central PKI by relying on secure processing elements (notably TPMs), which enable a decentralized credential management. For example, some DAA schemes allow vehicles to derive pseudonyms autonomously from long-term credentials [46], [12]. These pseudonyms are self-certified using DAA signatures that can be verified by other ITS participants.

III. Related Work

Traditional message-based revocation schemes take two opposing directions: *Active revocation*, where the information of revoked pseudonyms is distributed to vehicles in real time, and *passive revocation*, where pseudonyms have a short lifetime and are left to expire. Additionally, recent schemes propose a novel approach based on *self-revocation*.

Active revocation. Active revocation is employed by SCMS and consists in the distribution of Certificate Revocation Lists (CRLs) [11] to all vehicles in the network, containing the list of pseudonyms that have been revoked and thus should not be trusted. Vehicles use this information to decide whether or not to accept or discard a message received from other

[1] https://github.com/EricssonResearch/v2x-self-revocation

peers. CRL updates are issued regularly by the RA to indicate newly revoked pseudonyms, and must be received as quickly as possible by all vehicles to ensure the safety of the system. However, as CRLs grow large over time, the associated computation and communication overheads make these mechanisms less suitable for real-time systems. Several papers propose optimizations to reduce the size of CRLs and the verification overhead [41], [29], though recent surveys show that these optimizations are still far from ideal [45], [48]. The SCMS design relies on *linkage values* to identify all pseudonyms of a vehicle with a single value, effectively reducing the size of the CRL but significantly increasing the delay to decide whether to accept or not a network message [6].

Passive revocation. As an alternative to active revocation, some schemes propose issuing pseudonym certificates that have a fixed and short lifetime, forcing vehicles to request new pseudonyms regularly to continue participating in the network [25], [43], [1]. Internal CRLs of long-term credentials are used by the infrastructure to deny such requests to malicious vehicles, thus negating the need for pseudonym revocation. Passive revocation is also adopted in the ETSI standard [20].

Since pseudonym certificates are not revoked, vehicles detected as malicious can continue their operation until the last of their pseudonyms has expired. As such, malicious vehicles still have a non-negligible time window after their detection in which they can spread false information and cause safety-related incidents [16], and has been identified as an open issue [48]. Thus, to increase security, pseudonym lifetimes would need to be quite short. However, with shorter lifetimes and increasing participants in the network, the traffic between infrastructure and vehicles would increase significantly, along with the risk of Sybil attacks [14].

Self-revocation. Some DAA protocols adopt a scheme called self-revocation, where revocation is performed with the active cooperation of the vehicle to be revoked [46], [12], [30]. The concept of self-revocation was initially proposed by Förster et al. in their REWIRE protocol [21], as a privacy-preserving revocation scheme for vehicular networks that leverage a hardware-based TC in vehicles. In such a scheme, the RA broadcasts an Order for Self-Revocation (OSR), containing the pseudonym identifier to be revoked. The idea is that, upon reception of such message, the targeted vehicle will detect that the revoked pseudonym is one of its own and then proceed with the deletion of all its credentials, such that the vehicle will be unable to generate malicious messages any further. This approach eliminates the need for CRLs or short pseudonym lifetimes, providing substantial benefits in terms of performance and scalability [3].

This simple scheme works with the assumption that the TC is a trusted entity and will behave as expected, i.e., that it will delete the credentials as soon as the OSR is received. In case such messages are delayed or dropped, e.g., due to a network disruption or a malicious host, Förster et al. propose the use of *keep-alive messages* that are periodically broadcast by the RA to vehicles and used to detect network interruptions, causing the automatic deletion of all credentials in a TC if no messages are received for an extended period. However, this approach is described only informally, and does not address an attack scenario where only OSRs are dropped, but keep-alive messages are not. The REWIRE scheme was formally verified in [47], though the possible disruption of OSRs was not addressed. Furthermore, a similar *heartbeat* mechanism is later proposed in subsequent work [46], [12], [30], which suffers from the same limitations. Finally, all solutions seem to implicitly rely on a notion of trusted synchronized time to check the freshness of revocation requests and compute timeouts, but such a feature may not be available on all vehicles due to the limitations of current technologies [2], [4], neither do previous papers discuss possible issues related to clock drifting between vehicles and the RA.

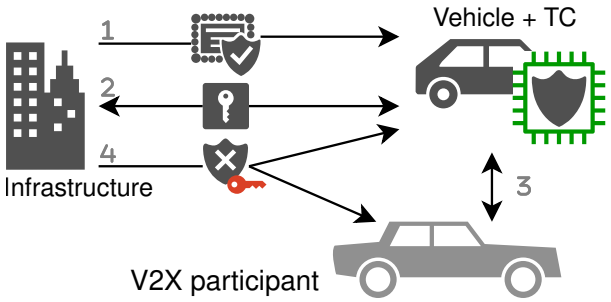

Fig. 1: Lifecycle of a V2X architecture. An infrastructure enrolls vehicles (1) and equips them with pseudonymous credentials (2) which they can use to take part in V2X communication (3). On-demand, the infrastructure revokes pseudonyms or certificates (4).

IV. MOTIVATION AND SYSTEM MODEL

Section III highlights that, while active and passive revocation schemes present some drawbacks, self-revocation of pseudonyms seems a promising solution to remove misbehaving or malicious participants from the V2X network efficiently and in a timely manner. However, related work on the topic ignores several challenges posed by realistic attackers and does not provide strong guarantees or upper time bounds for revocation. None of the previous papers effectively solved the problem of attackers actively dropping/delaying revocation requests, and the *heartbeat* idea was only informally introduced but never developed [46], [12], [30]. To the best of our knowledge, there are no self-revocation schemes available that work in practice, making these protocols unsuitable for real-world deployment in V2X systems. Earlier formal verification work [47] does not address this problem, and no related paper provides an upper-bound analysis nor a comprehensive evaluation. Additionally, previous work hints at the requirement of trusted timestamps in network messages, but none provides a full design that integrates a trusted time source in TCs. In fact, under a realistic attacker model, current confidential computing technologies, and TEEs particularly, cannot rely on a trusted time source [2]. Thus, previous works that implicitly assume trusted synchronized time cannot currently be implemented securely.

In this work, we aim to address these issues by providing a reliable, formally verified revocation mechanism that can be employed in V2X systems and beyond. We propose a design that considers the challenges of having a reliable time source in TCs, and we discuss optional extensions that may bring additional advantages. We strive to minimize the computational and communication overheads caused by revocation schemes,

making our design efficient and suitable for any kind of system and network. Below, we overview our system model, discuss attacker capabilities, and introduce our objectives. Our design is then detailed in Sect. V.

A. Vehicles and Trusted Components

In V2X, participants can be vehicles, pedestrians, traffic lights, and so forth. For readability, in this paper we only discuss vehicles, but our system model can be generalized to *any* participant in the network. We consider vehicles as integrated systems composed of a number of Electronic Control Units (ECUs) connected by a local in-vehicle network such as a CAN bus. These ECUs may also be connected to a wide variety of sensors and actuators, used to retrieve information from the vehicle itself or its surroundings (e.g., speed, position), or to perform actions (e.g., accelerate, brake). Typically, all such components are managed by a central unit called On-Board Unit (OBU), which is also responsible for processing network messages sent to and from the vehicle [31].

Additionally, we assume that vehicles contain a secure processing element called the Trusted Component (TC), responsible for credential management and cryptographic operations, analogously to DAA protocols (cf. Sect. II). In our system model, the TC is a passive device that exposes an interface to the OBU for tasks such as enrollment/attestation, pseudonym generation, and message signing/verification. Therefore, all sensitive operations are performed on the TC itself, and the OBU cannot directly access cryptographic keys or other sensitive material. Examples of TCs are the TPM, ARM TrustZone, or TEEs such as Intel Security Guard Extensions (SGX) and ARM Confidential Compute Architecture (CCA). Nowadays, trusted components are ubiquitous in modern hardware [38], and with recent developments in the automotive domain, starting with Infineon's automotive-certified Optiga TPM [27] (released in 2018) and automotive security processors such as the NXP NCJ38A [37] (available since 2020), TEE solutions are available and working groups within the Trusted Computing Group and Global Platform are active in designing automotive security services based on TEEs [42], [22].

In our attacker model (cf. Sect. IV-C), we consider all components of the vehicle potentially malicious or compromised, except the TC. Therefore, at high level, we can define a vehicle as the sum of two main components: the *untrusted host* and the TC. Just like sensors, timing components used for precise timestamping are also part of the untrusted host, such that the TC cannot rely on a trusted notion of time by itself [2], [4]. Therefore, our design does not assume a trusted time source in vehicles to guarantee a timely revocation, but relies on trusted timestamps (or "epochs") distributed by the ITS infrastructure to TCs (cf. Sect. V).

B. The Vehicle Lifecycle in a V2X Network

The lifecycle of a vehicle in a V2X network is depicted in Fig. 1 and, in short, is composed of four main phases: 1) *enrollment*, 2) *pseudonym generation*, 3) *operation*, and 4) *revocation*. Although our paper mainly focuses on the last phase, every revocation scheme, to be effective, has some dependencies on the other phases. For example, vehicles that do not properly check the validity of signatures might erroneously accept messages signed with invalid or revoked pseudonym certificates, making the revocation process useless. Thus, in this section we first compile a set of functional requirements that a V2X protocol should implement in order to use our revocation scheme effectively. These requirements are based on the architectures proposed in state-of-the-art V2X credential management systems, and are discussed at *high level*: Our goal is to make our revocation scheme protocol-agnostic in order to be compatible with most existing protocols such as DAA or technical standards such as ETSI and SCMS (cf. Sect. VIII-B).

Enrollment. Each vehicle is equipped with a TC which exclusively stores credentials and performs cryptographic operations with them (cf. Sect. IV-A). In the initial phase, a vehicle enrolls to the V2X network by authenticating to the infrastructure, e.g., by presenting a valid certificate [20]. With trusted components such as TEEs, *remote attestation* also becomes an essential step to make sure that the TEE is configured correctly and is up to date, and the TC is running genuine software [38]. Successful enrollment should result in the issuance of a *long-term credential*, used by the vehicle as a token to authenticate subsequent requests to the infrastructure and to obtain pseudonym certificates (cf. Sect. II).

Requirement 1: Vehicles correctly enrolled to the network are issued a long-term credential, stored in the TC.

Pseudonym generation. In this phase, a vehicle uses the long-term credential obtained during enrollment to get a new pseudonym certificate. This process can be done either *online*, where the pseudonym is issued by the infrastructure upon successful authorization [20], [6], or *offline*, where the vehicle's TC performs a key derivation from the long-term credential [46], [12]. Either way, only non-revoked vehicles should be able to obtain new pseudonyms.

Requirement 2: Pseudonym certificates are obtained using (valid) long-term credentials through online or offline protocols.

Operation. Pseudonyms are then used to authenticate messages that vehicles exchange with network peers. For readability and without loss of generality, this paper focuses on the communication between vehicles (V2V), but peers could also be pedestrians (V2P), smart city infrastructure (V2I), and so forth (cf. Sect. II). V2V messages may include information about the vehicles themselves (e.g., speed, location, direction) and the surrounding environment (e.g., traffic, accidents). These messages need to be integrity-protected and authenticated to prevent message spoofing or tampering, and time-stamped to provide freshness; Both actions are performed by the TC, using pseudonym credentials for the former and its internal time now for the latter (we elaborate more on now in Sect. V).

Requirement 3: V2V messages are signed by the TC using its own valid pseudonyms and timestamped according to an internal TC variable now, representing the TC's current time.

We then require that vehicles correctly check the authenticity of V2V messages to ensure that they have been signed using a valid pseudonym certificate. Furthermore, we demand that messages older than a predefined age are discarded, to prevent processing outdated information.

Requirement 4: Vehicles should check the signature of received V2V messages to ensure that they have been generated

using a valid pseudonym. A message with timestamp t_{v2v} should only be accepted by a receiving TC if not older than a predefined window T_{v} w.r.t. its internal time *now*:

$$t_{\mathsf{v2v}} \geq \mathit{now} - T_{\mathsf{v}} \tag{1}$$

Revocation. We expect the infrastructure to operate mechanisms for misbehavior detection and reporting, where pseudonyms can be flagged, e.g., for spreading messages that contain false information [6], [18]. Based on such reports, the RA can decide to revoke the corresponding vehicle's credentials using the reported pseudonym. This is the concept of *message-based revocation* introduced in Sect. II.

Requirement 5: The RA makes decisions on revocation by employing misbehavior detection mechanisms in the network, to identify and report misbehaving pseudonyms.

Moreover, we require that *only* non-revoked vehicles can enroll to the network, otherwise it would be trivial for attackers to bypass revocation by simply re-enrolling to the network. This can be achieved by keeping an internal blacklist of long-term credentials [20], [6]. Revoked vehicles may be reinstated at the discretion of the ITS infrastructure, e.g., after a factory reset.

Requirement 6: Only vehicles whose credentials have not been revoked can (re-)enroll to the V2X network.

We observe that a vehicle in a V2X network is operational iff two conditions are fulfilled: 1) the vehicle has *at least* one valid pseudonym, and 2) the TC's internal time *now* is loosely synchronized with the others, i.e., within window T_{v}. If either of these two conditions is not met, a vehicle cannot generate valid and fresh V2V messages, according to Reqs. 3 and 4. Hence, we define *effective revocation* as follows:

*Definition 1 (*effective revocation*):* A vehicle is considered effectively revoked when either all its pseudonyms are made unusable (or expire), *or* when its TC gets permanently de-synchronized, i.e., when *now* cannot be updated anymore.

C. Attacker Model

We consider a powerful attacker that has control over a vehicle with valid credentials and wants to disrupt the V2X network by spreading malicious V2V messages, avoiding revocation of their pseudonyms as long as possible. Examples of attack vectors include faulty sensors, software bugs in ECUs, root access to the attacker's own car, etc.

We assume a Dolev-Yao style attacker that controls the network stack and can tamper with, rearrange, resend, and stall network messages but not break established cryptography [13]. The attacker is assumed to have full control over all hardware and software on vehicles except for functionality implemented by their TC, which is protected and fully trusted. Entities in the ITS infrastructure (e.g., the RA) are also trusted.

Besides the aforementioned cryptographic limitations, our attacker is not able to perform physical attacks on the TC. We stress that voltage tampering attacks against the hardware can still be a real threat and cannot be tackled by network protocols, but need to be addressed on lower levels of the stack [9], [35]. Side channel attacks against the TC are also out of scope and will have to be mitigated through the usual means of hardware and software defenses [32], [8].

D. Objectives

Based on our system and adversary model, and the open issues of existing schemes (cf. Sect. III), we define objectives for a practical and secure real-world revocation mechanism:

- **O1 Guarantee of revocation:** Revocation should always succeed within a deterministic upper bound, even in case of delayed or dropped revocation requests. In particular, the *effective revocation time* is a time period starting with the revocation decision made by the RA and ending when the vehicle is effectively revoked.
- **O2 No assumptions on trusted time:** Synchronizing trusted time across multiple vehicles with a strong attacker is challenging and no assumptions should be made on having access to a local trusted time source.
- **O3 Scalability:** Any revocation mechanism suitable for V2X scenarios must be scalable in terms of vehicles and share of attackers in the network. This covers both computational as well as network overhead.
- **O4 Adaptability:** The revocation mechanism needs to be configurable for different network characteristics and security requirements. It must be able to tolerate varying degrees of network latency.

V. Design

In a nutshell, our design has the RA send periodic messages containing the pseudonyms of vehicles to be revoked, as well as information used for a loose time synchronization between all participants. Vehicles need to relay these messages to their TCs in order to be able to communicate with the network, otherwise they will eventually get de-synchronized. Hence, attackers cannot postpone revocation indefinitely, and eventually they become effectively revoked after a fixed amount of time that can be configured by system parameters (**O1**, **O4**). Below, we describe our design in detail.

Time synchronization. Our definition of effective revocation introduced in Sect. IV, together with Reqs. 3 and 4, suggests that TCs need a reliable representation of time (i.e., *now*). Since a trusted and synchronized local time source is not always available in TCs (**O2**), we rely on a trusted server (i.e., the RA) that distributes timing information to all vehicles in order to roughly synchronize them to a reference time frame, which could be the actual time and date or an epoch counter [5]. Note that vehicles can still use a local time source for precise timestamping in real-time communication, though it is considered part of the untrusted environment just as other sensors (cf. Sect. IV-A).

The heartbeat. The RA periodically distributes authenticated messages called heartbeats (HBs). HBs include a so-called *Pending Revocation List (PRL)*, i.e., a list of pseudonyms pending revocation. When the RA decides to revoke a pseudonym ps, based on Req. 5, it calls an internal REVOKE function to add the pseudonym identifier (e.g., a hash of its public key) to the PRL, such that subsequent HBs will contain this revocation request. As we will discuss in Sect. V-A, each pseudonym needs to stay in the PRL only for a limited period. In addition, every HB contains a timestamp t_{hb}, which is used to provide freshness information to the HB and to distribute timing information to the vehicles, used for time synchronization. In

summary, the format of a HB is the following:

$$HB := PRL \parallel t_{\mathsf{hb}} \parallel \mathit{sig}_{\mathsf{RA}}(PRL \parallel t_{\mathsf{hb}}) \quad (2)$$

Here, $\mathit{sig}_{\mathsf{RA}}()$ is a digital signature made by the RA. HBs can be either fetched manually by a vehicle or periodically distributed via *broadcast* to reach multiple vehicles at once. Additionally, they can also be exchanged in a peer-to-peer fashion when connectivity is limited (e.g., rural areas).

Processing a heartbeat. Vehicles should forward received HBs to their TCs. Similar to V2V messages (cf. Req. 4), the TC firstly verifies its digital signature to ensure that it is an authentic message generated by the RA, and then checks its freshness by ensuring that the timestamp t_{hb} is not lower than its current time now minus the validity window T_{v}, as follows:

$$t_{\mathsf{hb}} \;\geq\; \mathit{now} - T_{\mathsf{v}} \quad (3)$$

Upon successful verification of the HB, the TC synchronizes its time as below:

$$\mathit{now} := max(\mathit{now}, t_{\mathsf{hb}}) \quad (4)$$

Then, the TC performs *self-revocation* if any of its pseudonyms (represented by the set $Certs$) appears in the PRL:

$$PRL \cap Certs \neq \emptyset \;\Rightarrow\; \mathit{self\_revoke}() \quad (5)$$

Self-revocation puts the TC in a revoked state, where all subsequent signature requests made by the vehicle's OBU are denied. Credentials may also be deleted from memory. A TC can only be restored upon successful authorization by the infrastructure (e.g., via vehicle reset and re-enrollment).

Missing reception of heartbeats. Self-revocation is effective as long as HBs are correctly delivered to all TCs. However, attackers in the network, as well as compromised OBUs, might prevent this from happening in order to keep TCs operational for as long as possible (Sect. IV-C).

We note that a TC needs to process a HB in order to synchronize its time. In other words, if no valid HBs were processed for an extended period, the internal TC time now would stand still at the last processed HB, according to Eq. (4). This would lead to vehicle de-synchronization, in which the revoked vehicle would not be able to generate *fresh* messages, even though its TC is still operational (cf. Req. 3). In such case, other vehicles would start discarding all messages coming from the revoked vehicle as soon as Eq. (1) is not satisfied anymore. Thus, our security objective is guaranteed even when a revoked vehicle's TC does not process HBs.

Automatic revocation. To make such de-synchronization permanent, a revoked vehicle should not be able to process any more HBs without causing self-revocation. This can be trivially achieved by permanently keeping pseudonyms in the PRL. However, the PRL would become bigger and bigger, making the distribution of HBs as slow and inefficient as traditional CRLs (cf. Sect. III). To prevent this, we make TCs automatically perform self-revocation should they detect de-synchronization. We call this process *automatic revocation*, which is triggered when the timestamp t_{msg} of any received message (either a HB or a V2V message) is much bigger than the current time now, according to the equation below:

$$t_{\mathsf{msg}} > \mathit{now} + T_{\mathsf{v}} \;\Rightarrow\; \mathit{self\_revoke}() \quad (6)$$

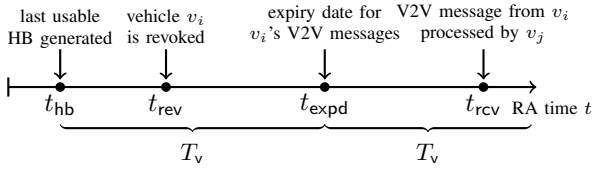

Fig. 2: Timeline diagram showing times from Sect. V-A and their relations. When estimating the worst-case revocation time, t_{hb} would coincide with t_{rev} and t_{rcv} would occur T_{v} after t_{expd}, i.e., t_{rcv} is $2T_{\mathsf{v}}$ after t_{rev}.

That is, vehicles de-synchronized for more than T_{v} are automatically revoked as soon as they process any *fresh* message. This, however, may also cause benign (i.e., unrevoked) vehicles to trigger revocation accidentally: If this happens, they can simply be restored by the infrastructure, though this may cause temporary service interruption and should be avoided when possible. See Sect. VII-C for a deeper discussion of the issue.

Eq. (6) ensures that, at any time, either: 1) the TC is in sync with the RA, being behind the current time t at most T_{v}, or 2) the TC is behind t by more than T_{v}, and any received message with $t_{\mathsf{msg}} \geq t$ will cause automatic revocation. Usually, only compromised vehicles that ignore received messages remain in case (2) without being revoked automatically. If an honest vehicle ends up being isolated from the network like that, it is most likely due to targeted jamming attacks or lacking network coverage. We discuss this issue further in Sect. VIII-C but rule it out here. Therefore, in what follows we assume for all honest vehicles in the network:

$$\mathit{now} \;\geq\; t - T_{\mathsf{v}} \quad (7)$$

A. Properties

The design elaborated above guarantees two important properties of revocation:

(i) *If a pseudonym ps belonging to a vehicle v_i is revoked by the RA, then after a fixed time period T_{eff} v_i will be effectively revoked;*
(ii) *Each pseudonym can be removed from the PRL after a fixed time period T_{prl} since its insertion time.*

Property (i) is crucial as it implies effective revocation, guaranteeing that the REVOKE function always succeeds within a fixed time period, even if HBs are dropped or delayed (**O1**). Property (ii) ensures that the size of the PRL (and consequently the HBs) does not keep growing over time, guaranteeing scalability (**O3**). Below, we give an intuition of properties (i) and (ii), while in Sect. VI we discuss our formal verification work to prove their correctness. Fig. 2 illustrates the general timeline assumed below.

Revocation time. Let us assume that, at a time t_{rev}, the RA orders revocation for a pseudonym ps owned by a malicious vehicle v_i. That vehicle would drop all HBs generated after t_{rev} to prevent TC_i from locking its credentials. Given that a TC only updates its internal time when processing a HB (Eq. (4)), we observe that t_{rev} is the highest value that TC_i would store as internal value now. Thus, all V2V messages

signed by TC_i would have a timestamp $t_{\mathsf{v2v}} \leq t_{\mathsf{rev}}$. Putting this information into Eq. (1) for a receiver v_j, we obtain:

$$\mathit{now}_j \leq t_{\mathsf{rev}} + T_{\mathsf{v}} \tag{8}$$

Eq. (8) shows that TC_j will discard all V2V messages signed by TC_i as soon as its internal time now_j exceeds $t_{\mathsf{rev}} + T_{\mathsf{v}}$. Conversely, for a time period T_{v} since t_{rev}, TC_j is still accepting messages coming from v_i. Assuming all honest vehicles in the network are at most T_{v} behind the RA time t (Eq. (7)) and using Eq. (8) in the inequation, we get:

$$t - T_{\mathsf{v}} \;\leq\; \mathit{now}_j \;\leq\; t_{\mathsf{rev}} + T_{\mathsf{v}} \tag{9}$$

Estimating the upper bound for $t - t_{\mathsf{rev}}$, we can then define the *effective revocation time* T_{eff} as follows:

$$T_{\mathsf{eff}} = 2T_{\mathsf{v}} \tag{10}$$

Time in PRL. As 1) t_{rev} is the highest value for now_i that TC_i can have after the revocation order for ps, and 2) Eq. (6) ensures that HBs containing timestamp t_{hb} greater than $\mathit{now}_i + T_{\mathsf{v}}$ would trigger revocation in TC_i, we can conclude that ps can be removed from the PRL as soon as RA time t passes $t_{\mathsf{rev}} + T_{\mathsf{v}}$. Thus, we can define the minimum period of time T_{prl} that ps needs to remain in the PRL as follows:

$$T_{\mathsf{prl}} = T_{\mathsf{v}} \tag{11}$$

B. Extensions

Trusted time source. A trusted time source in vehicles would enhance V2V communication through precise and trusted timestamping, making **O2** redundant. In our design, if the TC has access to a trusted time source its internal time can advance automatically and Eq. (4) would be unnecessary. In such case, however, the condition for automatic revocation would never occur, because Eq. (6) cannot be satisfied. Thus, we can define an additional condition by measuring the time that has passed since the most recent HB $max_{t_{\mathsf{hb}}}$ that has been received: If such time is higher than T_{v}, the TC would trigger revocation:

$$\mathit{now} > max_{t_{\mathsf{hb}}} + T_{\mathsf{v}} \;\Rightarrow\; \mathit{self\_revoke}() \tag{12}$$

As we will discuss in Sect. VI and Appendix A, we demonstrated that the properties defined in Sect. V-A still hold, with the assumption that TC and RA clocks are perfectly synchronized. In reality, however, there may be a bounded clock drift Δ that needs to be added to Eq. (10) and Eq. (11).

Security enhancements. Although T_{eff} is a strict upper bound on revocation time (cf. Sects. V-A and VI), it is still possible to improve the *average* revocation time. We devised two techniques to make it harder for attackers to postpone revocation, both are evaluated in Sect. VII-A to show their effectiveness:

1) *Encrypted HBs.* we observe that the worst-case scenario illustrated in Sect. V-A can be easily achieved if a malicious OBU has complete access to the content of a HB. Here, the OBU can arbitrary delay and drop HBs in order to evade revocation and avoid de-synchronization for as long as possible. By leveraging encryption, instead, the attacker would be unable to look at the content of a HB, and would be forced to make a guess on whether or not to drop, delay, or relay to the TC a HB received from network. HB encryption can be achieved by means of a pre-shared symmetric key obtained during enrollment and rotated periodically. In case such key is leaked, it would only cause attackers to gain access to the content of HBs, but they would still be unable to forge valid ones.
2) *Active revocation.* The PRL included in HBs can be used to perform active revocation (cf. Sect. III). That is, a TC could store the most recent PRL received via HB and use it as a lightweight CRL in the message verification phase (Req. 4). As the size of the PRL is generally small (cf. Sect. VII-B), this allows to reduce the average revocation time for malicious vehicles without causing too much overhead in the message verification phase.

VI. Formal Verification

We formally verified the correctness of properties (i) and (ii), defined in Sect. V-A, using TAMARIN prover [34], a symbolic verification tool for security protocols based on multi-set rewriting rules and first-order logic. This section summarizes our work, while Appendix A contains additional details. The complete models and instructions to execute the proofs locally on TAMARIN version 1.6.1 are available online [39].

A. Introduction on TAMARIN

TAMARIN is a symbolic verification tool that supports verification of security protocols, based on multi-set rewriting rules and first-order logic. Cryptography constructs, such as symmetric/asymmetric encryption, digital signatures and hashing, are built into the tool together with a Dolev-Yao adversary.

A protocol in TAMARIN is modeled as a set of *rules*, each of them defining a specific protocol step. The system's state is expressed as a multi-set of *facts*, i.e., predicates that store state information. Each rule has a pre-condition, i.e., facts that are consumed by the rule, and a post-condition, i.e., facts that are produced after its execution. In addition, *restrictions* allow to specify additional constraints on the rule's execution.

Rules can also produce special facts called *action facts*, which are not part of the system's state but rather are used to notify that a certain event has occurred (e.g., a HB has been generated). Action facts are used to define properties on the model, called *lemmas* in TAMARIN, using first-order logic. To verify a lemma, TAMARIN uses a "backwards reasoning" approach where it starts from a later state and reasons backwards to derive information about earlier states. In particular, TAMARIN starts with a state that contradicts the lemma, and tries to construct an execution trace to check whether that state is actually reachable or not via the defined rules. If a complete execution trace that reaches that state can be constructed, then the lemma is falsified, otherwise it is successfully verified.

B. Model

We modeled our revocation scheme considering both our main design and our extension when TCs have access to a trusted time source (Sect. V-B); The latter is discussed in Appendix A. Our models consider an arbitrary T_{v} and include three entities: 1) the RA, which generates HBs and revokes pseudonyms, 2) the TC, which processes HBs received from network and generates V2V messages, and 3) a generic third entity, which receives and verifies V2V messages generated by the TC. Since our focus is on verifying revocation properties

```
restriction IsLatestTime:
"All t #i . IsLatestTime(t)@i ==>
    not (
        Ex t2 #j . TimeIncrement(t2)@j & j<i
            & GreaterThan(t2, t)
        )"

rule advance_time:
[ !Time(t) ]
--[
    OnlyOnce(<'advance_time', t>)
    , TimeIncrement(t + '1')
]->
[ !Time(t + '1') ]
```

Fig. 3: Definition of the `advance_time` rule that increments the time counter by one, together with the `IsLatestTime` restriction that ensures the most recent value is used in a rule.

(cf. Sect. V-A), we did not focus on enrollment or credential management. We did, however, implement basic cryptographic functionalities for making and verifying digital signatures, using the *signing* built-in. This is enforced on both HBs and V2V messages. Additionally, we allow the TC to obtain and use an arbitrary number of pseudonym credentials.

Notion of time. Given that our goal is to measure the time since revocation, our model also requires a notion of progress. This was not trivial to implement in TAMARIN, as it does not provide any native support for time passing. Therefore, we leveraged the *multiset* built-in to model time as a logical counter that represents time steps, allowing our entities to perform any number of operations within the same time step. To increment the counter, we defined a rule that takes as pre-condition the current value and increments it by one (Fig. 3). The same built-in was also used to model the PRL.

Persistent facts. Time steps, timeouts, PRL and pseudonyms have all been modeled using *persistent facts*, i.e., facts that remain in memory persistently, even after being consumed by a rule. In our experience, this was a better choice than using *linear facts* which, once consumed, are deleted from the system's state. For example, persistent facts allow reusing the same `!Time(t)` fact in multiple rules, which reduces the number of possible states and thus the memory used by TAMARIN. We also experienced some undesired side-effects when using linear facts, such as infinite recursion due to the fact that rules were both consuming and producing `Time(t)` facts at the same time. However, the use of persistent facts cause multiple `!Time` facts to be in the system's state at the same time. For example, after the rule `advance_time` in Fig. 3 is executed for the first time, we would have both `!Time('1')` and `!Time('1'+'1')` facts in memory. This would potentially cause execution of rules using the old `!Time` fact instead of the new one. Although this inconvenience does not compromise the overall accuracy of the model, it does not represent a realistic scenario and it also causes a higher number of possible states. Hence, we added a restriction `IsLatestTime(t)` (cf. Fig. 3) in every rule to ensure that the latest time is always used. For the PRL, we used sequence numbers to restrict rules to use the most recent list.

Modeling T_v. We modeled T_v as a parameter initialized in the setup phase (`Init`). Our goal was to model T_v as an arbitrary value to prove our properties for any value greater than zero.

```
// property (i)
lemma effective_revocation [heuristic=o "oracle.py"]:
"All msg ps t #i . MessageAccepted(msg, ps, t)@i ==>
    Ex tv #j . SystemInitialized(tv)@j & j<i
    & not (
        Ex ps2 t_rev #k . RevocationIssued(ps2, t_rev)@k
        & GreaterThan(t, t_rev + tv)
    )"

// property (ii)
lemma no_heartbeats_processed_after_tolerance [heuristic=o "
    oracle.py"]:
"All prl t_hb t #i . HeartbeatProcessed(<prl,t_hb>, t)@i ==>
    Ex tv #j . SystemInitialized(tv)@j & j<i
    & not (
        Ex ps t_rev #k . RevocationIssued(ps, t_rev)@k
        & k<i
        & GreaterThan(t_hb, t_rev + tv)
    )"
```

Fig. 4: TAMARIN proof lemmas that prove properties (i) and (ii) in Sect. V-A.

To do so, we defined a first rule called `init_parameters`, which creates a linear fact `TvTmp` with initial value `'1'`. Then, we defined another rule called `increment_Tv` that consumes such linear fact and produces a new one with value incremented by one. Finally, the `Init` rule takes the current value and finalizes it, creating a persistent fact `!Parameters` used in other rules.

C. Lemmas

We defined three types of lemmas: 1) *Sanity lemmas*, to ensure that all rules can be executed at least once, i.e., that we defined the model correctly; 2) *Functional lemmas*, to ensure that the model correctly implements our design in Sect. V; and 3) *Proof lemmas*, to verify the properties in Sect. V-A. Table I provides the full list of lemmas and a short description.

Modeling systems that have an internal state and progress over time makes it extremely challenging to verify lemmas within a reasonable amount of time and memory. In order to build an efficient proof for some of our lemmas, we gave as input to TAMARIN a Python script called *oracle*, which provides a custom ranking for *goals*. Indeed, TAMARIN proves a lemma by attempting to solve the open goals of the constraint system, which are sorted based on some heuristics. By providing an oracle as input, goals can be ranked according to a custom heuristic. This was essential for us, as for some of the lemmas TAMARIN could not terminate with the default heuristics due to an inefficient ranking of goals. For example, solving `!Time` goals needs to be postponed as much as possible because, in an unconstrained system, time can have an arbitrary value and TAMARIN would get stuck in an infinite loop. By using a specific ranking, instead, we could give priority to other goals that add more and more constraints, allowing `!Time` goals to be solved efficiently. Our oracles guarantee that the verification only takes a few minutes in total. Table I indicates for which lemmas a custom oracle was needed.

Figure 4 shows the proof lemmas used to prove our properties. The lemma `effective_revocation` says that there is no state where *any* V2V message signed by a revoked ps is verified by another entity whose internal time is higher than $t_{\mathsf{rev}} + T_v$. Hence, the attacker is permanently

Lemma	Type	Oracle	Steps	Description
sign_possible	S		5	Ensures that the TC can sign V2V messages
generate_hb_possible	S		5	Ensures that the RA can generate and distribute HBs
issue_revocation_possible	S		5	Ensures that the RA can revoke pseudonyms, adding them to the PRL
processing_hb_possible	S	✔	7	Ensures that the TC can receive and process a HB
revocation_possible	S	✔	10	Ensures that the TC can self-revoke when receiving a HB
process_message_possible	S		7	Ensures that a third entity can process V2V messages signed by the TC
exists_par_tv_5	S		8	Used to verify that T_{v} is arbitrary
exists_par_tv_8	S		11	Used to verify that T_{v} is arbitrary
no_signing_after_revocation	F		2	Ensures that the TC cannot make signatures after performing self-revocation
all_heartbeats_processed_within_tolerance	F	✔	130	Ensures that the TC correctly discards outdated HBs, i.e., older than T_{v}
all_signatures_max_time_t_rev	F	✔	156	Ensures that, if revocation is mandated at time t_{rev}, the TC cannot make valid V2V messages with timestamp greater than t_{rev}
effective_revocation	P	✔	1964	Verifies property (i) in Sect. V-A
no_heartbeats_processed_after_tolerance	P	✔	100	Verifies property (ii) in Sect. V-A

TABLE I: Full list of lemmas proven with TAMARIN on our model. *Type* is either *S* (sanity), *F* (functional) or *P* (proof). *Oracle* indicates whether an oracle was needed, while *Steps* shows the number of steps required by TAMARIN to prove the lemma.

de-synchronized from the network and T_{eff} can be estimated assuming Eq. (7), as shown in Sect. V-A. The lemma no_heartbeats_processed_after_tolerance, instead, proves that a revoked TC cannot process *any* HBs whose timestamp is bigger than $t_{\mathsf{rev}} + T_{\mathsf{v}}$. There are two possible reasons why this happens: 1) The TC has processed a HB generated since t_{rev}, meaning that it is in a revoked state since then; 2) The TC has not processed any HB generated since t_{rev}, meaning that processing a HB with timestamp bigger than $t_{\mathsf{rev}} + T_{\mathsf{v}}$ would trigger automatic revocation. Either way, this can be directly translated to our second property: Pseudonyms belonging to the revoked TC can be safely removed from the PRL after $t_{\mathsf{rev}} + T_{\mathsf{v}}$, i.e., $T_{\mathsf{prl}} = T_{\mathsf{v}}$ (cf. Eq. (11)).

VII. EVALUATION

In this section, we evaluate our scheme and give additional insights. Sect. VII-A focuses on property (i), showing experimental results on revocation time obtained from simulating different attackers and network conditions (**O1**). Sect. VII-B demonstrates that, thanks to property (ii), the PRL is kept small even with a high number of revocations, proving the scalability of our approach (**O3**). Finally, Sect. VII-C discusses how T_{v} should be chosen, based to the characteristics of the network and the desired security and efficiency properties (**O4**).

A. Revocation Time

We implemented a full prototype that realizes the revocation scheme from Sect. V and we evaluated it on a simulated V2X network running on Kubernetes [10]. The simulation environment allows to approximate the average revocation time for a range of network parameters in order to demonstrate the effectiveness of our scheme. Additionally, we simulated network malfunctions such as delays and packet losses, to show that our approach is resilient against real-world conditions. Code and configuration files are available online [39].

Implementation. Due to the lack of open-source V2X implementations, we implemented a custom V2X protocol based on [28], in which a group symmetric key is shared among all vehicles to sign and verify V2V messages. In addition, TCs are able to autonomously generate their own pseudonyms (similar to [46], [12]), which are random identifiers attached to messages as metadata. Thus, all V2V messages are linked to the (pseudonymous) identity that signed them, allowing for the revocation of bad actors' credentials. This protocol satisfies all requirements of our revocation scheme (cf. Sect. IV-B), but may not be practical for real-world use cases. Still, it allows us to obtain realistic average revocation times for different attacker models, numbers of vehicles, network conditions, and validity windows T_{v}.

In our prototype, vehicles are grouped into one or more edge areas, while a centralized infrastructure is responsible for enrollment (via an Issuer) and revocation (via a RA). Besides, on each edge area a Road-Side Unit (RSU) is responsible for fetching the latest HB from the RA and broadcasting it to all vehicles in the area. Furthermore, an edge area is divided into groups to simulate proximity, so that a vehicle broadcasts V2V messages only to its neighbors, i.e., other vehicles currently located in the same group. Each vehicle has two components: the untrusted host, or OBU, and the TC. The latter is implemented as a passive device that exposes the following API to the OBU: JOIN to enroll in the network, CREATE to generate a new pseudonym, SIGN and VERIFY to sign and verify V2V messages, and HEARTBEAT to process a HB. In our prototype, JOIN enrolls a vehicle in the network, after which the TC obtains: 1) the RA's public key, needed to authenticate the HBs, 2) the V2V group credential, needed to sign and verify V2V messages, 3) a long-term identifier, and 4) the current timestamp used for synchronization.

V2V messages are randomly-generated data, signed and verified using the symmetric group key. Revocation is triggered by reporting pseudonyms to the RA, who eventually adds them to the PRL via the REVOKE API. A *reporter* component inspects the V2V traffic in each area and periodically reports a random pseudonym.

Attackers. We implemented different behaviors for the OBU, depending on how HBs are processed and relayed to the TC. In particular, we simulated attackers trying to evade revocation by dropping or delaying the HBs. Our goal was to show that our properties actually hold in practice (Sect. V-A), but also that the average revocation time can be significantly improved with some extensions (Sect. V-B). To this end, we implemented three main attacker levels:

- *honest*: Simulates normal behavior as a baseline, i.e., the OBU relays all HBs to the TC as soon as they are received from the RSU;
- *smart*: The *smart* attacker delivers HBs normally as long as none of its pseudonyms are in the PRL. However, it attempts to evade revocation by dropping all HBs that contain any of its pseudonyms;
- *blind*: Simulates a scenario where HBs are encrypted (Sect. V-B), thus attackers have no access to their content and do not know if and when their pseudonyms will get revoked. Therefore, the *blind* attacker has to make a guess on whether to deliver or drop a HB. We tried several logics for this attacker, and the one with best results was the attacker relaying only one HB every T_{v}, to allow clock synchronization and prevent revocation to be triggered automatically (Eq. (6)).

Experimental setup. We ran our simulations on a Kubernetes [10] cluster with version 1.22.17. Up to 400 simulated vehicles are scheduled on 8 worker nodes, each with Intel(R) Xeon (R) CPU E5-2680 v4 @ 2.40GHz and two 10Gbps network interfaces, and 168 GB of total available memory.

We evaluated several scenarios with different values for T_{v}, with or without trusted time in TCs. In this paper we only discuss a scenario without trusted time where T_{v} is equal to 30 seconds, resulting in a T_{eff} and T_{prl} of 60 and 30 seconds, respectively, according to Eqs. (10) and (11). Complete simulation results are provided with our artifacts [39].

Each simulation ran for two hours and spawned 400 vehicles over a single edge area, divided into 20 groups. Of the vehicles, 10% were malicious, using one of the attacker levels described above. Each vehicle could have only two concurrent pseudonyms, disabling pseudonym rotation to not create bias on the revocation time. A new HB was generated every second by the RA, and every second the RSU fetched and distributed the latest HB to vehicles. In order to simulate network malfunctions and denial of service attacks, the RSU either dropped or delayed HBs, each with probability 0.4, such that only 20% of the HBs were delivered to vehicles without delay. Also, vehicles generated and distributed V2V messages every second. To increase the number of VERIFY events, messages from malicious vehicles had a 30% probability to be replayed. A pseudonym was reported and revoked every 10 seconds to gather sufficient experimental data (more than 600 revocations).

Results. The box plots in Fig. 5 illustrate the distribution of revocation times for each attacker level. An additional box plot labeled *smart-prl* shows simulation results under a *smart* attacker, where TCs use the PRL to perform active revocation, as described in Sect. V-B. Each value represents the time between the revocation of a pseudonym ps (REVOKE event) and the last V2V message signed with ps that was verified by a non-malicious TC (VERIFY event). Such time period represents the *effective revocation time* for that particular pseudonym (cf. Sect. IV-B).

It should be noted that we filtered out negative time values from the results: such values represent a situation where the last VERIFY event happened *before* the REVOKE event, for example because the TC owner of the revoked pseudonym was already in a revoked state due to the revocation of another of its pseudonyms, or simply because that pseudonym was not used to sign messages in the time after REVOKE. Either way, these values were not interesting for our evaluation.

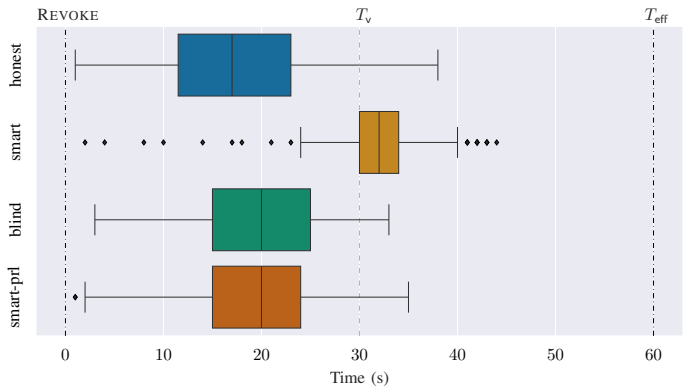

Fig. 5: Simulation results for $T_{\mathsf{eff}} = 60s$. The boxes show, for each revoked pseudonym, the time between REVOKE and the last VERIFY event, aggregating more than 600 revocations but excluding negative values.

Fig. 5 shows that pseudonyms belonging to *honest* vehicles are revoked rather quickly. Still, as V2V messages have a validity period of T_{v} and receiving vehicles' internal time may lag behind the RA within T_{v}, the effective revocation time varied widely, with a median of 17 seconds.

The simulation with the *smart* attacker level resulted in a median of 32 seconds and the majority of revocations between 24 and 40 seconds. This attacker is able to postpone revocation by dropping HBs, yet not relaying fresh HBs eventually causes de-synchronization in the TC. Therefore, in 99% of cases revocation was effective within 2/3 of T_{eff}, as honest vehicles were able to stay synchronized with the infrastructure even when simulating severe network malfunctions.

As for the *blind* attacker level, we experienced a higher number of negative values in our results. This can be explained by the fact that a *blind* OBU is not able to recognize which HBs can be dropped or delayed, which ones are fresh and which are old, and which ones can be delivered to the TC without causing revocation. As a result, and considering the network malfunctions we simulated, the TC is more likely to get de-synchronized and thus automatically revoked, even before revocation is mandated by the RA. This suggests that encrypting HBs makes it more difficult for an attacker to evade revocation. In the few cases where the vehicle was still operating after the REVOKE event, instead, revocation rarely took longer than T_{v}.

Finally, the *smart-prl* simulation showed a significant improvement over the *smart* one, with values more evenly distributed between 1.5 and 37.5 seconds and median of 20 seconds. However, these results were negatively affected by our simulated network malfunctions. As with traditional CRLs, the ability of a TC to discard messages coming from revoked vehicles depends on how quickly HBs are received and processed; Thus, with limited connectivity the effective revocation time increases. Indeed, we experienced better results when simulating scenarios with lower drop/delay rates for HBs.

Overall, our simulations showed the feasibility of our revocation scheme in a simulated though practical scenario. In the average case, revocation takes much less than T_{eff} to be effective, even when attackers attempt to evade it. Furthermore, simulating network malfunctions did not incur permanent de-synchronization of non-malicious vehicles, demonstrating the resiliency of our approach under real-world conditions.

B. Size of PRL

The size of a heartbeat is not constant. Instead, it depends on the size of the PRL, which varies over time according to the number of pseudonyms revoked and the parameter T_{prl} (Eq. (11)). If many pseudonyms are revoked at the same time, the PRL will be large until T_{prl} time has passed and they are evicted from the PRL again. To calculate the expected average PRL size, we approximate adding and removing pseudonyms from the list as a stochastic process and create a Markov model based on these parameters. The stationary distribution of this Markov chain then gives the probabilities for each given size of the PRL after an arbitrary number of steps.

Below, we elaborate on the core formulas and the expected values of the PRL size for a chosen set of parameters. Appendix B contains a complete explanation of the utilized probabilities, the Markov model, and a discussion of the nuances of modelling the PRL size as a probabilistic process.

PRL as a Markov Model. The process of adding and removing pseudonyms from the PRL can be seen as a finite state machine where the states are the possible sizes of the list. Transitions to a higher or lower state then happen on the addition or removal of a pseudonym, w.r.t. a base revocation probability p per pseudonym and time step. A discrete time Markov chain describes the probability of moving from state i to state j for each possible state [24]. As such, this Markov model represents the probabilities for gaining and losing pseudonyms from the PRL due to revocation or the removal of pseudonyms after T_{prl}, respectively. Assuming that the PRL contains i out of n possible pseudonyms at a time step, we can independently model the probabilities of gaining and losing k pseudonyms as two distinct binomial probabilities $G_{i,k}$ and $L_{i,k}$, respectively:

$$G_{i,k} = \binom{n-i}{k} \cdot p^k \cdot (1-p)^{n-i-k} \tag{13}$$

$$L_{i,k} = \binom{i}{k} \cdot (\frac{1}{T_{\mathsf{prl}}})^k \cdot (1-\frac{1}{T_{\mathsf{prl}}})^{i-k} \tag{14}$$

Observe that $G_{i,k} = 0$ if $i+k > n$ and $L_{i,k} = 0$ if $i < k$. We can then combine these binomial probabilities into a Markov matrix with the following property:

$$p_{i,j} = \sum_{l=Max(i-j,0)}^{i} L_{i,l} \cdot G_{i,l-i+j} \tag{15}$$

At its core, this Markov matrix contains all probabilities of moving from list size i to list size j as a combination of losing $l \leq i$ pseudonyms from the list and adding $l-i+j \leq j$ new ones, where in each time step, a pseudonym is removed from the list with probability $\frac{1}{T_{\mathsf{prl}}}$ and added with probability p. While this is not reasonable to calculate the effective list size after a specific number of steps, it suffices in a probabilistic model to determine the expected, average list size.

Average size of the PRL. In Markov chains, a stationary distribution is a state probability that maintains its distribution even after taking a step in the Markov chain [24], represented by the stationary vector of the Markov matrix. It is thus a probabilistic equilibrium that, once reached, will be stable forever. After startup, a Markov chain will eventually reach its stationary distribution after enough steps have been taken. We can calculate the equilibrium for our Markov chain for parameters that are useful in the real world and for different revocation probabilities. By accumulating the probabilities for each given state, we can thus calculate the maximum list size as a percentile of possible states.

We evaluate the impact of a growing share of attackers on the PRL size in the context of two baseline scenarios. In these scenarios, *honest* pseudonyms, i.e., pseudonyms in the network that do not belong to attackers, may still experience a seldom revocation due to erroneous behavior, e.g., due to a faulty sensor. In the first baseline scenario, we consider a 1% probability that a pseudonym gets revoked at least once within a day, i.e., a 1% chance for an expected revocation every 86400 time steps of one second each. In the second scenario, we raise this to the extreme case of a 99% probability of being revoked at least once a day or 86400 time steps.

Fig. 6 depicts the 75th, 90th, and 99.99th percentiles of the PRL sizes for these two scenarios. The plot uses the parameters of $n = 800$ pseudonyms and $T_{\mathsf{prl}} = 30$, to align with the evaluation described in Sect. VII-A. Each honest lifetime scenario is paired with a growing share of attackers in the network. Attackers are modeled to have a 75% chance of causing a pseudonym revocation every 30 minutes on average, which simulates a strain on the PRL that is by magnitudes larger than any honest pseudonym would normally incur.

In most situations, the PRL contains fewer than 10 and 12 pseudonyms for the two baseline scenarios, respectively. Even with a share of 20% of network participants that cause revocations heavily, the 99.99th percentile of PRL size still stays below 17. This means that even for coordinated peak revocation events, the PRL size will stay in very manageable margins. For comparison, we estimated the expected number of revocations over a day: they range from roughly 8 to 10500 for Scenario 1, and 3600 to 13500 for Scenario 2 – a worst case average of one revocation every 6-8s.

C. Choice of T_{v}

The choice of T_{v} is crucial in our scheme. We evaluated four reference values of 30, 150, 300 and 900 seconds and visualize the results in Fig. 7. Below, we discuss the impact of T_{v} across multiple dimensions.

Revocation time. The first dimension is related to the revocation time, and is shown in the top-left plot of Fig. 7. According to Eq. (10), the effective revocation time T_{eff} is linearly dependent on T_{v}. Therefore, the smaller T_{v}, the shorter it takes to revoke, resulting in a safer system.

HB frequency. The frequency with which HBs are generated and distributed by the RA is not fixed arbitrarily, but depends on T_{v}. Here, it is essential to ensure that honest vehicles

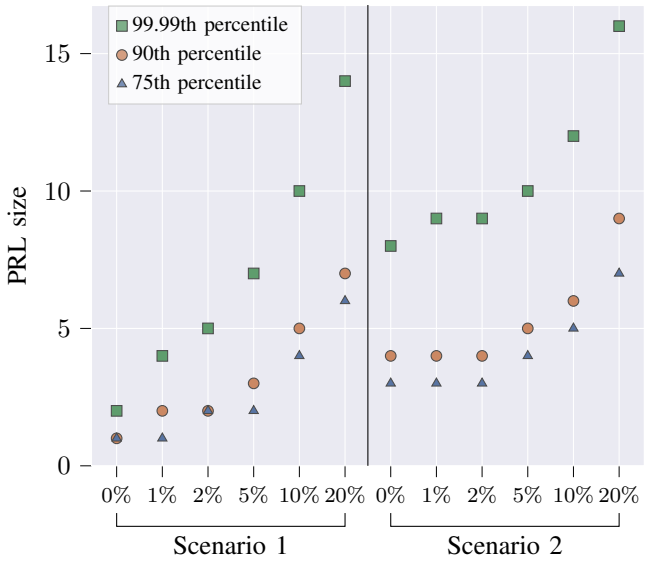

Fig. 6: Markov model percentiles for maximum PRL sizes for $T_{prl} = 30s$, $n = 800$, and different shares of attackers in each baseline revocation scenario.

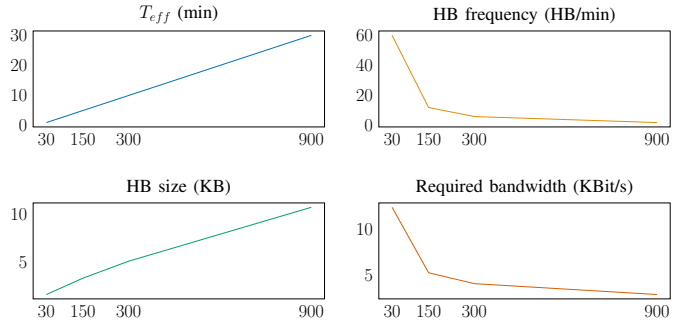

Fig. 7: Evaluation of the impact of T_{v} on the V2X system. The X axes show the value of T_{v} in seconds.

remain loosely synchronized with the infrastructure during their operation. This means that, to avoid temporary service interruption, TCs need to process *at least* one HB every T_{v} (cf. Sect. V). Thus, the frequency of HBs is inversely proportional to the value of T_{v} with a certain coefficient N, which accounts for network latencies and missing packets. The top-right plot of Fig. 7 shows the frequency with coefficient $N = 30$, i.e., a new HB is generated every $T_{\mathsf{v}}/30$ seconds. With larger T_{v}, the number of HBs per minute drastically decreases, resulting in fewer resources used.

HB size. According to Eq. (2), the size of a HB is linearly dependent on the size of the PRL, plus a constant value that represents the sizes of timestamp t_{hb} and digital signature $sig_{\mathsf{RA}}()$. Additionally, as discussed in Sect. VII-B, the size of the PRL is directly impacted by the value of T_{v}. That is, the higher T_{v} is chosen, the longer each pseudonym needs to stay in the PRL, according to Eq. (11). This means that, on average, HBs are bigger when T_{v} increases. The bottom-left plot of Fig. 7 shows the HB size when varying T_{v}, considering the 99.99 percentile of the PRL size in relation to T_{v} on the worst case scenario, i.e., Scenario 2 with 800 pseudonyms and 20% of attackers in the network (cf. Fig. 6). We also assume that a pseudonym identifier is 64 bytes, t_{hb} is 8 bytes, and we overestimate the size of $sig_{\mathsf{RA}}()$ to 512 bytes, which is higher than most RSA and elliptic curve signatures [36].

Network bandwidth. Lastly, it is important to evaluate the network bandwidth required for transmitting HBs. This can be computed by multiplying the size of a HB by the frequency with which HBs are distributed. Results show that the minimum bandwidth needed to transmit HBs is rather small, i.e., around 12 KBit/s in the worst-case scenario and with large N.

Conclusions. The choice of T_{v} should be a good compromise between security and usability. The smaller T_{v} is, the shorter a revocation takes, but it also increases the resources used and the risk of de-synchronization in honest vehicles. Besides, a small T_{v} is feasible when the V2X network is reliable and low-latency, e.g., in urban areas, but may not be always practical, e.g., in rural areas. On the other hand, a large T_{v} results in a more resilient and efficient V2X system at the cost of a slower revocation. However, compared with passive revocation (cf. Sect. VIII-A), even T_{eff} values of 30 minutes are a significant improvement on the revocation time, coming at negligible cost, as shown in Fig. 7.

VIII. Discussion

Our design advances the state of the art by providing several advantages compared to related work (Sect. III), which we analyze in Sect. VIII-A. We discuss compatibility with V2X protocols such as ETSI, SCMS and DAA in Sect. VIII-B, and we explain a few corner cases of our design in Sect. VIII-C. Finally, we examine privacy implications in Sect. VIII-D.

A. Qualitative Comparison with Related Work

This section performs a qualitative analysis of network overhead, verification overhead and revocation time of our approach compared with related work.

Network overhead. As discussed in Sect. VII-C, the bandwidth required to transmit HBs is as low as a few kilobits per second in most cases, even considering targeted attacks that cause a high number of revocations. This is a significant improvement over active revocation schemes that leverage CRLs, since these lists are ever growing and thus their size increases over time. We even outperform schemes that combine active and passive revocation to reduce the CRL size: For example, [41] suggests periods in the CRL of 3-4 weeks for each pseudonym. Compared to our evaluation (cf. Sect. VII-C), this is still three to four orders of magnitude higher.

Furthermore, unlike passive revocation schemes that require frequent pseudonym changes [25], our design consumes minimal resources at runtime. Since HBs can be broadcast within an edge area, the number of messages transmitted by the infrastructure for revocation only depends on the number of edge areas and not on the number of vehicles. This makes the system extremely scalable, as the RA only needs to distribute relatively small-sized HBs at low frequency (cf. Sect. VII-C).

Verification overhead. Active revocation schemes incur some latency during the verification of V2V messages to check the revocation status of pseudonyms. Even though some improvements exist to either reduce the CRL size or optimize the

verification time, such schemes still cause a non-negligible overhead (Sect. III). Our design, instead, does not require any additional verification overhead except for processing HBs, which are received more seldomly than V2V messages (cf. Sect. VII-C). Thus, the overhead for the TC should be comparable to passive revocation schemes. In Sect. V-B we propose to use the PRL to perform active revocation, which is considered an *optional* feature that can be enabled only when the PRL is small enough, which we demonstrated to be true in most cases (Sect. VII-B).

Revocation time. Compared to active revocation, our design provides comparable results in the average revocation time, especially when leveraging the extensions discussed in Sect. V-B and evaluated in Sect. VII-A. However, active revocation does not have an upper bound on the effective revocation time, as some vehicles may receive CRL updates with unpredictable delay. Besides, our scheme outperforms passive revocation since a rotation of pseudonyms by the infrastructure every few minutes is infeasible [25], while schemes relying on pre-issued certificates with activation codes have a huge gap between revocation and actual removal of malicious actors from the network [43], [41].

B. Compatibility with Existing Protocols

ETSI ITS. In line with our system model (cf. Sect. IV-B), vehicles following *ETSI TS 102 941* [20] obtain long-term *enrollment credentials* from an Enrollment Authority, after successful authentication of their canonical identity. These credentials are then used to obtain pseudonymous *authorization tickets* from an Authorization Authority. Timestamps are used for replay protection [19]. ETSI [20] further proposes passive revocation of pseudonyms by centrally revoking long-term credentials (cf. Sect. II). Our scheme is applicable here and can largely improve on revocation time and network overhead. This, however, requires TCs in all vehicles.

SCMS. Although similar to the ETSI architecture, the SCMS scheme employs active revocation (cf. Sect. II). Our scheme can entirely replace CRLs in SCMS to reduce the network utilization and nullify the verification overhead that vehicles incur when processing V2V messages, which may be high due to the use of linkage values in the CRL [6]. Analogously to ETSI, the design of SCMS would need to include TCs.

DAA. On the academic side, DAA-based protocols have been proposed as an approach for pseudonym generation with strong privacy properties (cf. Sect. II). Vehicles use a JOIN protocol to enroll, authenticating themselves via attestation to receive a long-term credential. To obtain pseudonym certificates, some protocols require communication with an Authorization Authority [25]; others allow TCs to derive pseudonyms autonomously via an CREATE protocol [46], [12]. While many DAA protocols do not explicitly discuss replay protection, we argue that Eq. (1) is still a requirement of the V2X network, and thus provided. Under this assumption, even DAA protocols are compatible with our revocation scheme.

DAA protocols that already propose self-revocation can benefit from our design by having stronger guarantees on revocation time. However, protocols where TCs derive pseudonyms directly [46], [12] might not satisfy Req. 6 because the infrastructure cannot map pseudonyms to long-term identities, and therefore revoked vehicles might be able to re-enroll the network. While this has clear benefits in terms of privacy, it raises security concerns (cf. Sect. VIII-D).

Applicability beyond V2X. Our revocation scheme can be generalized to revoke *any* type of credential. Hence, it may be applied to other use cases, e.g., in scenarios where 1) real-time communication requirements hamper the use of classic revocation checks, 2) users may generate pseudonymous credentials for enhanced privacy w.r.t. peers or the infrastructure, or 3) revocation needs to be fast and reliable to avoid serious damage by bad actors. V2X networks check all boxes but we see benefits for, e.g., smart cities, direct mobile-to-mobile communication, and other peer-to-peer applications.

C. Security Considerations

Vehicles powered off. Vehicles that are temporarily powered off cannot process any HBs, meaning that potential revocations may be missed without the TC noticing, and without being able to execute automatic revocation. Thus, it is important to persist the value of the last t_{hb} processed by a TC, such that vehicles would remain synchronized when powered off for a short time. For longer offline periods, instead, vehicles would need to re-synchronize with the infrastructure and, simultaneously, check their revocation status.

Self-identification of pseudonyms. The underlying assumption of our revocation scheme is that, to perform self-revocation, a TC is able to identify its own pseudonym certificates within the PRL. However, it may not be always possible to do so when the TC occasionally deletes some pseudonyms from memory, e.g., due to pseudonym rotation. A trivial solution to this problem would be to never delete any pseudonyms, though this may not be possible in resource-constrained environments (e.g., TPMs). Previous work [21], [47] discusses a more efficient solution that leverages cryptographic tokens, allowing TCs to safely delete their pseudonyms.

Denial-of-Service Attacks. An attacker may cause vehicles to get de-synchronized via network attacks, preventing them from processing HBs. This could, e.g., be achieved by jamming all wireless network activity close to a victim. While not being practical at scale, jamming attacks would likely disconnect the victim from the network regardless of the credential management mechanisms employed, and may thus require re-enrollment or other connection management steps. Therefore, such attacks likely do not cause additional harm when our approach is used.

D. Privacy Discussion

Privacy requirements in V2X include that adversaries cannot link a pseudonymous signature to a canonical identity or other pseudonyms, or observe the use of specific network resources [46], [20], [48]. Aside from adversarial interactions, data minimization towards the infrastructure is also a design goal of V2X systems to minimize the overall attack surface [6].

Our design does not compromise the privacy provisions of the underlying certificate management steps – enrollment, pseudonym generation, etc. – because HBs only contain public information and an adversary receiving PRLs only sees a list of pseudonyms (similarly to traditional CRLs) but learns no

further information about specific vehicles, nor can they link pseudonyms to each other. Learning when specific pseudonyms are considered compromised by the infrastructure is unavoidable, and breaking the underlying cryptographic primitives is outside of our attacker model.

However, Req. 6 and technical standards in Europe [20] and the US [6] highlight that revoked vehicles must no be able to re-enroll. This compromises data minimization in the V2X infrastructure since it necessitates mapping a pseudonym back to the canonical identity or long-term credentials of the vehicle and then deny-listing that identity. There are DAA-based approaches that preserve unlinkability of pseudonyms towards the infrastructure, as TCs generate pseudonyms autonomously [46], [12]. However, this in turn raises security concerns since re-enrollment of revoked vehicles becomes feasible. Possible technical solutions to this conundrum depend on the V2X protocol and its credential management scheme, as well as the capabilities of the TC. Therefore they are out of scope of this work.

IX. Summary & Conclusions

Revocation of credentials and pseudonymous identities is a critical aspect of security and trust management in V2X systems, necessary to prevent malicious actors from manipulating V2X communication for personal gain or to harm people and infrastructure. In this paper, we present and evaluate a self-revocation scheme that addresses scalability and security issues of existing approaches, and we formally prove that revocation in our scheme cannot be bypassed and completes with a configurable upper time bound. Furthermore, our approach is compatible with recent standardization efforts in the ETSI framework in Europe and the security credential management system in the US, and also with privacy-preserving pseudonym schemes that rely on Direct Anonymous Attestation. Compared to other solutions, our design requires a very limited communication bandwidth and we argue that the distributed nature of our scheme reduces computational resource utilization, specifically at the infrastructural layer. We show that our approach scales well to large numbers of revocations and allows for revocation times in the order of seconds, outperforming even short-lived pseudonyms. We conclude that approaches that leverage trusted computing technologies, such as ours, come with improved scalability and performance characteristics that warrant inclusion in future standardization efforts in application scenarios that deal with large numbers of participants and critical low-latency communications, beyond V2X and ITS. Future work needs to investigate privacy-preserving solutions to block malicious actors from re-joining ITS networks after revocation, without incurring availability issues for well-behaving actors when losing connectivity.

Acknowledgments

This research is partially funded by the Research Fund KU Leuven, by the Flemish Research Programme Cybersecurity, and by the CyberExcellence programme of the Walloon Region, Belgium. This research has received funding under EU H2020 MSCA-ITN action 5GhOSTS, grant agreement no. 814035. This research is partially funded by a grant of the Research Foundation — Flanders (FWO), under grant number 11E5120N.

References

[1] K. J. Ahmed and M. J. Lee, "Secure LTE-Based V2X Service," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3724–3732, 2018.

[2] F. Alder, G. Scopelliti, J. Van Bulck, and J. T. Mühlberg, "About Time: On the Challenges of Temporal Guarantees in Untrusted Environments," in *Proceedings of the 6th Workshop on System Software for Trusted Execution*, ser. SysTEX '23. Association for Computing Machinery, 2023, p. 27–33.

[3] A. Angelogianni, I. Krontiris, and T. Giannetsos, "Comparative Evaluation of PKI and DAA-based Architectures for V2X Communication Security," in *2023 IEEE Vehicular Networking Conference (VNC)*, 2023, pp. 199–206.

[4] F. M. Anwar and M. Srivastava, "Applications and Challenges in Securing Time," in *12th USENIX Workshop on Cyber Security Experimentation and Test (CSET 19)*. Santa Clara, CA: USENIX Association, Aug. 2019.

[5] H. Birkholz, T. Fossati, W. Pan, and C. Bormann, "Epoch Markers," Internet Engineering Task Force (IETF), Active Internet-Draft, 03 2023.

[6] B. Brecht, D. Therriault, A. Weimerskirch, W. Whyte, V. Kumar, T. Hehn, and R. Goudy, "A Security Credential Management System for V2X Communications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 12, pp. 3850–3871, 2018.

[7] E. Brickell, J. Camenisch, and L. Chen, "Direct Anonymous Attestation," in *Proceedings of the 11th ACM Conference on Computer and Communications Security*, ser. CCS '04. Association for Computing Machinery, 2004, p. 132–145.

[8] C. Canella, J. Van Bulck, M. Schwarz, M. Lipp, B. Von Berg, P. Ortner, F. Piessens, D. Evtyushkin, and D. Gruss, "A systematic evaluation of transient execution attacks and defenses," in *28th USENIX Security Symposium (USENIX Security 19)*, 2019, pp. 249–266.

[9] Z. Chen, G. Vasilakis, K. Murdock, E. Dean, D. Oswald, and F. D. Garcia, "VoltPillager: Hardware-based fault injection attacks against Intel SGX Enclaves using the SVID voltage scaling interface," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 699–716.

[10] Cloud Native Computing Foundation, "Kubernetes," https://kubernetes.io, 2023, accessed 2023-11-16.

[11] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile," Internet Engineering Task Force (IETF), RFC 5280, 2008.

[12] N. Desmoulins, A. Diop, Y. Rafflé, J. Traoré, and J. Gratesac, "Practical anonymous attestation-based pseudonym schemes for vehicular networks," in *2019 IEEE Vehicular Networking Conference (VNC)*. IEEE, 2019, pp. 1–8.

[13] 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.

[14] J. R. Douceur, "The sybil attack," in *International workshop on peer-to-peer systems*. Springer, 2002, pp. 251–260.

[15] ETSI, "LTE; 5G; Security aspect for LTE support of Vehicle-to-Everything (V2X) services," European Telecommunications Standard Institute (ETSI), Technical Specification (TS) TS 133 185, 07 2017, version 14.0.0.

[16] ——, "Intelligent Transport Systems (ITS); Security; Pre-standardization study on pseudonym change management," European Telecommunications Standard Institute (ETSI), Technical Report (TR) TR 103 415, 04 2018, version 1.1.1.

[17] ——, "5G; Service requirements for enhanced V2X scenarios," European Telecommunications Standard Institute (ETSI), Technical Specification (TS) TS 122 186, 11 2020, version 16.2.0.

[18] ——, "Intelligent Transport Systems (ITS); Security; Pre-standardization study on Misbehaviour Detection," European Telecommunications Standard Institute (ETSI), Technical Report (TR) TR 103 460, 10 2020, version 2.1.1.

[19] ——, "Intelligent Transport Systems (ITS); Security, ITS communications security architecture and security management," European Telecommunications Standard Institute (ETSI), Technical Specification (TS) TS 102 940, 07 2021, version 2.1.1.

[20] ——, "Intelligent Transport Systems (ITS); Security; Trust and Privacy Management," European Telecommunications Standard Institute (ETSI), Technical Specification (TS) TS 102 941, 11 2022, version 2.2.1.

[21] D. Förster, H. Löhr, J. Zibuschka, and F. Kargl, "REWIRE – Revocation Without Resolution: A Privacy-Friendly Revocation Mechanism for Vehicular Ad-Hoc Networks," in *Trust and Trustworthy Computing*. Cham: Springer International Publishing, 2015, pp. 193–208.

[22] GlobalPlatform, "Trust & security in automotive systems," *White Paper*, Oct. 2023. [Online]. Available: https://globalplatform.org/resource-publication/trust-security-in-automotive-systems/

[23] G. Guette and O. Heen, "A TPM-based architecture for improved security and anonymity in vehicular ad hoc networks," in *2009 IEEE Vehicular Networking Conference (VNC)*, 2009, pp. 1–7.

[24] H. Hermanns, "Markov Chains," in *Interactive Markov Chains*. Springer, 2002, pp. 35–55.

[25] C. Hicks and F. D. Garcia, "A Vehicular DAA Scheme for Unlinkable ECDSA Pseudonyms in V2X," in *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, 2020, pp. 460–473.

[26] IEEE, "Information technology – Trusted platform module library – Part 1: Architecture," Institute of Electrical and Electronics Engineers (IEEE), Standard Std 11889–1:2015, 01 2015.

[27] Infineon, "OPTIGA TPM - Trusted Platform Module," https://www.infineon.com/cms/en/product/security-smart-card-solutions/optiga-embedded-security-solutions/optiga-tpm/, 2023, accessed 2023-11-16.

[28] M. K. Jangid and Z. Lin, "Towards A TEE-based V2V Protocol For Connected And Autonomous Vehicles," in *Workshop on Automotive and Autonomous Vehicle Security (AutoSec)*, vol. 2022, 2022, p. 27.

[29] V. Kumar, J. Petit, and W. Whyte, "Binary Hash Tree Based Certificate Access Management for Connected Vehicles," in *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, ser. WiSec '17. Association for Computing Machinery, 2017, p. 145–155.

[30] B. Larsen, T. Giannetsos, I. Krontiris, and K. Goldman, "Direct Anonymous Attestation on the Road: Efficient and Privacy-Preserving Revocation in C-ITS," in *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, ser. WiSec '21. Association for Computing Machinery, 2021, p. 48–59.

[31] J. Liu, S. Zhang, W. Sun, and Y. Shi, "In-Vehicle Network Attacks and Countermeasures: Challenges and Future Directions," *IEEE Network*, vol. 31, no. 5, pp. 50–58, 2017.

[32] X. Lou, T. Zhang, J. Jiang, and Y. Zhang, "A survey of microarchitectural side-channel vulnerabilities, attacks, and defenses in cryptography," *ACM Computing Surveys (CSUR)*, vol. 54, no. 6, pp. 1–37, 2021.

[33] P. Maene, J. Götzfried, R. de Clercq, T. Müller, F. Freiling, and I. Verbauwhede, "Hardware-Based Trusted Computing Architectures for Isolation and Attestation," *IEEE Transactions on Computers*, no. 99, 2017.

[34] S. Meier, B. Schmidt, C. Cremers, and D. Basin, "The TAMARIN Prover for the Symbolic Analysis of Security Protocols," in *Computer Aided Verification*. Springer Berlin Heidelberg, 2013, pp. 696–701.

[35] K. Murdock, D. Oswald, F. D. Garcia, J. Van Bulck, D. Gruss, and F. Piessens, "Plundervolt: Software-based fault injection attacks against Intel SGX," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 1466–1482.

[36] NIST, "Digital Signature Standard (DSS)," National Institute of Standards and Technology (NIST), Technical Standard FIPS 186-5, 2023.

[37] NXP Semiconductors, "Automotive-Qualified Embedded Secure Element (SE)," https://www.nxp.com/products/security-and-authentication/secure-car-access/automotive-qualified-embedded-secure-element-se:NCJ38A, 2023, accessed 2023-11-16.

[38] M. Schneider, R. J. Masti, S. Shinde, S. Capkun, and R. Perez, "Sok: Hardware-supported trusted execution environments," *arXiv preprint arXiv:2205.12742*, 2022.

[39] G. Scopelliti, C. Baumann, F. Alder, E. Truyen, and J. T. Muehlberg, "Artifacts for Efficient and Timely Revocation of V2X Credentials," Nov. 2023. [Online]. Available: https://doi.org/10.5281/zenodo.10210425

[40] R. Sedar, C. Kalalas, F. Vázquez-Gallego, L. Alonso, and J. Alonso-Zarate, "A comprehensive survey of v2x cybersecurity mechanisms and future research paths," *IEEE Open Journal of the Communications Society*, vol. 4, pp. 325–391, 2023.

[41] M. A. Simplicio, E. L. Cominetti, H. Kupwade Patil, J. E. Ricardini, and M. V. M. Silva, "ACPC: Efficient revocation of pseudonym certificates using activation codes," *Ad Hoc Networks*, vol. 90, p. 101708, 2019.

[42] Trusted Computing Group, "TCG Vehicle Services Working Group," https://trustedcomputinggroup.org/work-groups/vehicle-services/, 2023, accessed 2023-11-16.

[43] E. Verheul, C. Hicks, and F. D. Garcia, "IFAL: Issue First Activate Later Certificates for V2X," in *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, 2019, pp. 279–293.

[44] A. A. Wagan, B. M. Mughal, and H. Hasbullah, "VANET Security Framework for Trusted Grouping Using TPM Hardware," in *2010 Second International Conference on Communication Software and Networks*, 2010, pp. 309–312.

[45] Q. Wang, D. Gao, and D. Chen, "Certificate Revocation Schemes in Vehicular Networks: A Survey," *IEEE Access*, vol. 8, pp. 26 223–26 234, 2020.

[46] J. Whitefield, L. Chen, T. Giannetsos, S. Schneider, and H. Treharne, "Privacy-enhanced capabilities for VANETs using direct anonymous attestation," in *2017 IEEE Vehicular Networking Conference (VNC)*. IEEE, 2017, pp. 123–130.

[47] 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.

[48] T. Yoshizawa, D. Singelée, J. T. Mühlberg, S. Delbruel, A. Taherkordi, D. Hughes, and B. Preneel, "A Survey of Security and Privacy Issues in V2X Communication Systems," *ACM Computing Surveys*, p. 3558052, Aug. 2022.

APPENDIX A
TAMARIN MODELS

This appendix extends Sect. VI, providing more details on how we translated our design to TAMARIN and how we captured the properties of our design. Here, we also discuss our variant where a trusted time source in the TC is available (cf. Sect. V-B).

A. Mapping our Design to TAMARIN

To map our design to Tamarin, we used only six main rules:

- `RA_generate_heartbeat`: This rule takes as input the current time and PRL and produces a new HB as output, signed with the RA's private key. The HB is also sent out to the network channel, and an action fact `HeartbeatGenerated(HB)` is produced. This rule can only be executed once per PRL and time step to reduce the number of states, since multiple execution with the same parameters would generate the same HB, which is not relevant in our model since TAMARIN already allows replays of network messages.
- `RA_issue_revocation`: Taking as input the current time, PRL, and a pseudonym's public key, this rule generates a new PRL by adding the pseudonym to the list and incrementing the sequence number by one. We added a restriction to revoke each pseudonym only once. After execution, the action fact `RevocationIssued(ps, t)` is generated.
- `TC_get_pseudonym`: This rule allows the TC to obtain a new pseudonym credential. How exactly this credential is obtained (e.g., via a communication to the infrastructure) is outside of our scope; What is important in our

```
// property (i)
lemma effective_revocation [heuristic=o "oracle.py"]:
" All msg ps t #i . MessageAccepted(msg, ps, t)@i ==>
    Ex tv #j . SystemInitialized(tv)@j & j<i
    & not (
        Ex ps2 t_rev #k . RevocationIssued(ps2, t_rev)@k
        & k<i
        & GreaterThan(t, t_rev + tv + tv)
    )"

// property (ii)
lemma no_operations_after_timeout [heuristic=o "oracle.py"]:
"All t #i . NewOperation(t)@i ==>
    Ex tv #j . SystemInitialized(tv)@j & j<i
    & not (
        Ex ps t_rev #k . RevocationIssued(ps, t_rev)@k
        & k<i
        & GreaterThan(t, t_rev + tv)
    )"
```

Fig. 8: Properties of our design with a trusted time source in TC translated to TAMARIN lemmas.

model is that the TC can obtain and use an arbitrary number of pseudonyms. The only restriction of this rule is to ensure that only non-revoked TCs can obtain new pseudonyms (cf. Req. 2).

- `TC_process_heartbeat`: The TC processes a HB taken as input from the channel. Restrictions ensure that the received HB has a valid signature and timestamp. In addition, the TC has to check whether to perform self-revocation or not. To do so, we divided this rule in two mutually-exclusive rules that produce different results: if any of the TC pseudonyms is in the PRL, the action fact `Revoked(t)` is produced, otherwise it is not. Either way, a `HeartbeatProcessed(HB,t)` action fact are produced, along with a new `!Time(t_hb)` persistent fact to synchronize the TC time.
- `TC_sign_message`: The TC generates a new V2V message signed with a pseudonym's private key. It takes the latest time as input, while restrictions ensure that the TC has not been previously revoked (i.e., the `Revoked(t)` action fact does not exist). The rule produces a signed timestamped message and sends it to the out channel. A `Signed(msg,ps)` action fact is produced as well.
- `process_message`: This rule models a generic third entity that receives a V2V message from network and verifies it. If the signature is valid and the message is fresh, a `MessageAccepted(msg,ps,t)` action fact is generated.

B. Trusted Time Source in TCs

The design that models a trusted time source in TC comes with a major difference, i.e., that TC and RA use the same `!Time` facts. That is, this means that their clock is perfectly synchronized, and the TC does not need to process HBs to advance its internal time.

Table II provides the full list of lemmas used in this variant, most of which are analogous to the main model. The main difference is that we added three *reusable lemmas*, i.e., lemmas that TAMARIN can leverage to prove other lemmas. This was helpful since we experienced that the two proof lemmas were harder to verify efficiently than in our main model. By defining additional lemmas marked as `reuse`, we could instead build efficient proofs, allowing TAMARIN to terminate in only a few minutes.

Fig. 8 shows the proof properties of this model. Two are the main differences compared to the main model: First, the `effective_revocation` lemma can now directly prove our first property to calculate the exact value for T_{eff}. Indeed, we now assume that all TCs are perfectly synchronized with the RA time, hence the revocation time is now absolute and comes at $t_{\mathsf{rev}} + 2T_{\mathsf{v}}$. Second, thanks to internal timeouts in TC, we can have a stronger proof for the second property, saying that not only does the TC not process HBs generated after $t_{\mathsf{rev}} + T_{\mathsf{v}}$, but also that the TC cannot perform *any* operations after that time because the automatic revocation timeout would be triggered. To prove this, we defined an additional rule called `TC_do_operation`, which models a generic operation performed by the TC via the `NewOperation(t)` action fact. This is then used in `no_operations_after_timeout` to prove that there cannot be `NewOperation(t)` action facts with a value for t bigger than $t_{\mathsf{rev}} + T_{\mathsf{v}}$.

APPENDIX B
CALCULATING EXPECTED SIZE OF THE PRL

This appendix covers the basics of the utilized probabilities to then explain the Markov matrix. We also describe the calculation of the stationary distribution as a means to calculate the expected PRL size. Finally, we discuss the computation of the probabilities and expected values used in Sect. VII-B.

A. Utilized Probabilities

Recall from Sect. VII-B that we model the processes of gaining and losing pseudonyms from the PRL as two individual probabilities, then we combine them in the next step into a Markov chain. The two probabilities shown in Eq. (13) and Eq. (14) are denoted as G_{il} and L_{il} respectively and can be seen as probability matrices that at location il have the probability for being in state i and gaining or losing l pseudonyms from the list. The core assumption for these two probabilities is that both adding and removing pseudonyms from the revocation list can be seen as a binomial distribution that only depends on: 1) the current size of the list, i.e., how many certificates there are left to be revoked or removed from the list, 2) the probability for each pseudonym to be revoked at any given time, and 3) the time that a pseudonym stays in the list. As such, this model assumes that pseudonyms are renewed as soon as they are evicted from the PRL, i.e., the list can never hold more than the total number of n pseudonyms, which is a realistic assumption for large n as it is unlikely that so many pseudonyms in the system would be revoked in a short time during regular operation. Furthermore, in this way we model the process of losing pseudonyms from the list as a probabilistic process that has the expected value at T_{prl}. We, however, can not model a precise eviction of the pseudonym from the PRL after T_{prl} time steps. In our view this is an acceptable tradeoff as on average and over the lifetime of the PRL, the modeled behavior comes very close to the actual behavior of evicting pseudonyms from the PRL after exactly T_{prl} time steps.

Lemma	Type	Oracle	Steps	Description
sign_possible	S		6	Ensures that the TC can sign V2V messages
generate_hb_possible	S		5	Ensures that the RA can generate and distribute HBs
issue_revocation_possible	S		5	Ensures that the RA can revoke pseudonyms, adding them to the PRL
processing_hb_possible	S		9	Ensures that the TC can receive and process a HB
revocation_possible	S		13	Ensures that the TC can self-revoke when receiving a HB
process_message_possible	S		7	Ensures that a third entity can process V2V messages signed by the TC
exists_par_tv_2	S		5	Used to verify that T_{v} is arbitrary
exists_par_tv_4	S		7	Used to verify that T_{v} is arbitrary
no_signing_after_timeout	F		205	Ensures that the TC cannot make signatures after performing automatic revocation
no_signing_after_revocation	F		2	Ensures that the TC cannot make signatures after performing self-revocation
all_heartbeats_processed_within_tolerance	F	✔	260	Ensures that outdated HB are correctly discarded by the TC
all_messages_accepted_signed_exists	R	✔	62	Ensures that, for each V2V message verified by a third entity, the same message was previously signed by the TC
all_messages_accepted_within_tolerance	R	✔	282	Ensures that outdated V2V messages are correctly discarded by the third entity
no_messages_accepted_after_revocation	R	✔	74	Ensures that, if revocation is mandated at time t_{rev}, all TC messages accepted by a third entity cannot contain timestamp greater than $t_{\mathsf{rev}} + T_{\mathsf{v}}$
effective_revocation	P	✔	4	Verifies property (i) in Sect. V-A
no_operations_after_timeout	P	✔	76	Verifies property (ii) in Sect. V-A

TABLE II: Full list of lemmas proven with TAMARIN on our model with trusted time. *Type* is either *S* (sanity), *F* (functional), *R* (reuse) or *P* (proof). *Oracle* indicates whether an oracle was needed, while *Steps* the number of steps required by TAMARIN to prove the lemma.

B. Markov Model

We can now combine these two probabilities into a matrix that at location p_{ij} has the probability of moving from state i to state j. Starting with the simple example of $n = 3$ pseudonyms, multiple entries of this matrix are straightforward:

- Starting at any state i and stepping into state $j = 0$ requires to not revoke any new pseudonyms and to lose all existing pseudonyms from the list. This is the combination of the two probabilities $L_{ii}G_{i0}$. For example, going from state $i = 2$ to $j = 0$ requires to lose 2 pseudonyms when there are 2 in the PRL, and requires to gain no new pseudonyms when there are already 2, leading to the two probabilities $L_{22}G_{20}$
- Starting at state $i = 0$ and stepping into a state j implies that no pseudonyms can be lost from the PRL and j pseudonyms have to be added.
- Similarly, starting in state $i = 3$ and stepping into any state j implies that no pseudonyms can be gained, but $j - i$ pseudonyms are removed from the PRL.

The most interesting state transitions in this probability matrix are the transitions that consist of multiple combinations of events. One example is the transition from $i = 2$ to $j = 1$. Here, we first have the obvious possibility that we reach the state $j = 1$, i.e., one pseudonym in the PRL, by losing one pseudonym and gaining none. However, we also have the possibility to lose both pseudonyms in the PRL and gain one, leading to still one remaining pseudonym in the PRL. This is because the pseudonyms in the PRL and outside of the PRL are independent and pseudonyms can still be revoked while others are removed from the list. Thus, the state transition consists of the following parts: $p_{21} = L_{21}G_{20} + L_{22}G_{21}$.

The full state probability matrix for $n = 3$ pseudonyms can be seen below. Note that a matrix for n pseudonyms is a $n + 1 \times n + 1$ matrix to accommodate the state of the empty PRL.

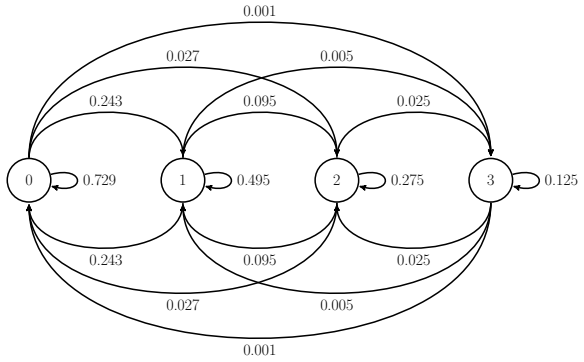

Fig. 9: A graph of the Markov chain with the exemplary parameters of $n = 3$, $p = 0.1$, and $T_{prl} = 2$.

$$P = \begin{bmatrix} L_{00}G_{00} & L_{00}G_{01} & L_{00}G_{02} & L_{00}G_{03} \\ L_{11}G_{10} & L_{10}G_{10}+L_{11}G_{11} & L_{10}G_{11}+L_{11}G_{12} & L_{10}G_{12} \\ L_{22}G_{20} & L_{21}G_{20}+L_{22}G_{21} & L_{20}G_{20}+L_{21}G_{21} & L_{20}G_{21} \\ L_{33}G_{30} & L_{32}G_{30} & L_{31}G_{30} & L_{30}G_{30} \end{bmatrix}$$

Such probability matrices are also called discrete time Markov chains [24]. Fig. 9 depicts the transition graph for the above Markov chain when we assume the parameters of $p = 0.1$ and $T_{prl} = 2$. Based on this first small example, we can approach a closed formula for an arbitrary number of pseudonyms (n). This closed formula is shown in Eq. (15). The core idea is that each entry depends on a combination of gaining and losing pseudonyms based on the maximum of $i - j$ and 0 and ranges up to i. Then, any state combination that is not possible will be 0 through the binomial coefficient. However, this range is necessary to catch the complicated combinations in the center of the matrix. Each combination is then based on losing l pseudonyms from the list and gaining $l - i + j$ pseudonyms, which basically iterates through the combinations with l as a stepping variable.

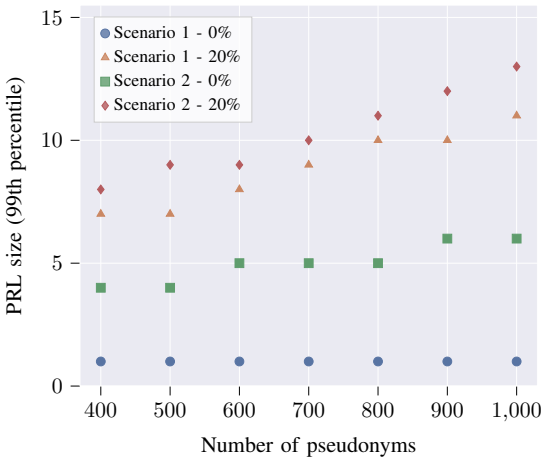

Fig. 10: 99th percentile PRL sizes for fixed $T_{prl} = 30$ and varying number of pseudonyms (n) under four scenarios.

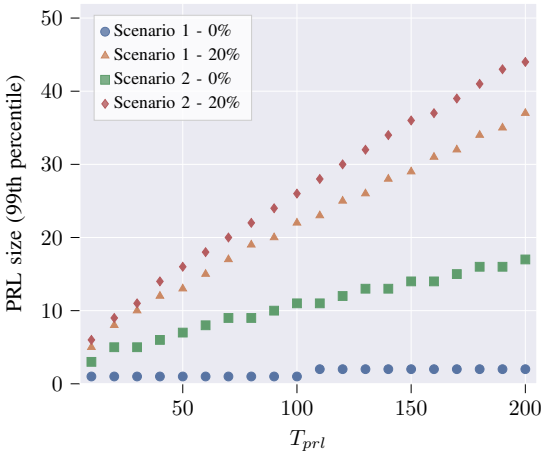

Fig. 11: 99th percentile PRL sizes for fixed $n = 800$ pseudonyms and varying T_{prl} under four scenarios.

C. Calculating the Expected PRL Size

In discrete time Markov chains, a probability state vector π can be multiplied with the Markov matrix P to gain the probabilities for π after a single transition in the model [24] as follows: $\pi' = \pi \cdot P$. This, however, is only sufficient to gain the state probabilities after specific amounts of steps. In contrast to this approach, Markov chains may approach a so-called state equilibrium which is a stationary probability distribution that does not change even after taking a step in the Markov chain. The stationary distribution follows the equation $\pi = \pi \cdot P$ [24].

To gain this stationary distribution, we solve the following equation system:

$$\begin{aligned} &\pi = \pi P \\ &\leftrightarrow \pi - \pi P = 0 \\ &\leftrightarrow \pi(I - P) = 0 \\ &\leftrightarrow (I - P)^T \pi^T = 0 \end{aligned}$$

Lastly, since there may be several solutions to this linear equation, we add the specific constraint to this equation system that the sum of π is equal to 1. This is the case for each probability vector in Markov models [24], and follows the intuition that, from any state probability, the probability to reach any other state is 1. To add this constraint, we add a row of 1 to $(I - P)^T$ and append a 1 to the zero vector. Solving this linear equation yields a stationary distribution that is stable. Fig. 6 summarizes this stationary distribution by showing the list sizes that occur to 75%, 90%, and to 99.99%. The graph shows these percentiles under different scenarios, i.e., with different shares of attackers in the network.

In the following, we expand this evaluation with Figures 10 and 11. Fig. 10 depicts only the 99th percentile for a fixed T_{prl} and only focuses on the four extreme cases: For each of the two baseline scenarios, 0% and 20% of attackers in the network. The horizontal axis then depicts a growing number of pseudonyms (and, therefore, vehicles) in the network. This is to show that a larger number of vehicles does not exponentially grow the expected size of the PRL. Instead, the 99th percentile of the list size grows linearly with n. Similarly, Fig. 11 shows the same situation for a fixed number of pseudonyms at $n = 800$ but a varying window for T_{prl}. This second graph shows that the PRL size also grows linearly with a linear increase in the time that a pseudonym stays in the PRL.

D. Probabilities and Expected Revocations

Above, p is the probability of revocation in each time step. As time steps are at the granularity of seconds, and our system runs for a long time, it turns out that the per-step probabilities leading to realistic revocation rates are very small. Instead, the probabilities used in the PRL size evaluation were described in terms of a compound probability q that a revocation occurs at least once over a number of s steps. This probability can be computed via the geometric series

$$q(p, s) = \sum_{i=1}^{s} p \cdot (1 - p)^{i-1} = 1 - (1 - p)^s$$

which sums up the probabilities that the first revocation occurs the first time in step $i \in [1 : s]$. Note that this is equal to 1 minus the probability that in all s steps no revocation occurs. Solving for p we obtain

$$p(q, s) = 1 - \sqrt[s]{1 - q}\,.$$

We used this formula to compute the baseline per-step probabilities underlying our scenarios for honest vehicle and attacker separately. The final probabilities p for the Markov models were then obtained by averaging the probabilities weighted according to the share of attackers.

For estimating the expected numbers of revocations in our scenarios we needed to calculate the expected value for n pseudonyms over s steps. If each step was independent this would simply be $n \cdot s \cdot p$, however, as explained above, in our Markov model at most n pseudonyms can be revoked at a time. We compensated for this fact by taking into account the chance p_{prl} that a given pseudonym is on the PRL at any given time and computed the desired estimate E_{rev}:

$$E_{\text{rev}} = n \cdot s \cdot p \cdot (1 - p_{\text{prl}})$$

We set p_{prl} to the median PRL size, as obtained by the Markov model for each scenario, divided by n.

Software	Tested version
Docker	23.0.2
Docker Compose	2.17.2
kubectl	1.28.2
Minikube	1.31.2 (running Kubernetes 1.27.4)

TABLE III: List of all software dependencies required to run our artifacts, along with the version that we used in our setup.

APPENDIX C
ARTIFACT APPENDIX

This Appendix contains a complete description on the artifacts presented in our paper, along with detailed instructions for running the artifacts locally and reproducing our results.

A. Description & Requirements

This section provides all the information necessary to recreate the experimental setup to run our artifacts. All experiments can run on a commodity desktop machine. For experiment (E2), we provide a scaled-down configuration that can still provide meaningful results. See Sect. C-E for more details.

1) How to access: The artifacts are publicly available on Github[2]. While the `main` branch contains the latest version of the code, the `ndss-24-artifacts` tag contains the exact version of the code that was submitted for review in the artifact evaluation. The latter is also available on Zenodo [39].

2) Hardware dependencies: Our artifacts can be run on a commodity desktop machine with a x86-64 CPU. To ensure that all artifacts run correctly, it is recommended to use a machine with at least 8 cores and 16 GB of RAM.

3) Software dependencies: A recent Linux operating system is required, preferably one between Ubuntu $\geq$ 20.04 and Debian $\geq$ 10. Our artifacts have been tested on Ubuntu 22.04 LTS (Jammy). Additional dependencies are listed in Tab. III.

4) Benchmarks: None.

B. Artifact Installation & Configuration

We first require that our repository is downloaded to a local folder, e.g., via `git clone`. To install all required dependencies, we then provide an `install.sh` script that can be used by Ubuntu and Debian users to install up-to-date versions of the packages listed in Tab. III. Alternatively, such dependencies can be installed manually via official channels. Note that we require running the installation script and subsequent experiments using a non-root user with `sudo` privileges.

All artifacts leverage Docker containers for ease of use. However, except for experiment (E2), all experiments can be run locally. Additionally, most experiments have customizable parameters. In this Appendix, we provide instructions for running all experiments with Docker using a fixed set of parameters, but our repository contains extensive instructions for customizing each experiment.

[2]https://github.com/EricssonResearch/v2x-self-revocation

C. Experiment Workflow

Our artifacts contain three independent experiments. The first consists in the formal verification of our design using the Tamarin prover. The second experiment simulates a V2X scenario on Kubernetes, where our scheme is evaluated under different conditions. The third experiment performs a statistical evaluation on the size of PRL to assess the scalability of our approach.

The proposed workflow runs the three experiments sequentially. Our repository contains scripts and a Makefile that can be used to automate all experiments. For brevity, this Appendix only illustrates the `make` commands to run at each step, while extensive documentation is provided in our repository.

D. Major Claims

- (C1): Our scheme guarantees revocation within a deterministic upper bound T_{eff}, even in case of delayed or dropped revocation requests. This is proven by experiment (E1), whose results are reported in Sect. VI, Appendix A and Tabs. I and II.
- (C2): Our scheme is resilient to severe network malfunctions and delays that may occur in a real-world scenario. Even in these harsh conditions, revocation times are not only within the upper bound proven in C1, but also significantly shorter on average. This is proven by experiment (E2), whose results are reported in Sect. VII-A and Fig. 5. Additional experiments are described in our Github repository.
- (C3): Our scheme scales well with the number of vehicles and attackers in the network. Additionally, even when choosing a small T_{eff}, the required network and computational resources are still low. This is proven by experiment (E3), whose results are reported in Sect. VII-B, Appendix B and Figs. 6, 7, 10 and 11.

E. Evaluation

This section includes all the operational steps and experiments which must be performed to evaluate our artifacts and validate our results. In total, all experiments require between 30 and 60 human-minutes and around 8 compute-hours. We assume that the machine is configured correctly with required dependencies, as described in Sects. C-A and C-B. The same instructions, along with additional details, are provided in the top-level README file of our repository.

1) Experiment (E1) - Claim (C1): [Tamarin proofs] [2 human-minutes + 10 compute-minutes]: The experiment consists in using the open-source Tamarin prover tool (version 1.6.1[3]) to verify our revocation design. Tamarin receives as input our model specification along with the properties (*lemmas*) we want to verify, and attempts to verify such properties. Our artifacts include two separate models, one that illustrates the main design in Sect. V and another that illustrates a variation of our design where TC have a trusted time that is synchronized with the RA (Sect. V-B). See Sect. VI and Appendix A for more information.

[Preparation] In a new shell, go to the `proofs` folder.

[3]The recently-released version of Tamarin 1.8.0 is not able to verify our models efficiently. We are currently investigating this issue.

[Execution] Firstly, prove the main design, which is called `centralized-time`. The command to run, along with the expected output, is provided below.

```
# Prove the `centralized-time` model (5 minutes)
# Expected output:
# - Tamarin prints a "summary of summaries" at the end
# - In the summary, all lemmas are marked as "verified"
# - a `output_centralized.spthy` file under `./out`
make prove MODEL=centralized-time OUT_FILE=
     output_centralized.spthy
```

Secondly, prove the variation of our design with trusted time, which is called `distributed-time`. The command to run, along with the expected output, is provided below.

```
# Prove the `distributed-time` model (5 minutes)
# Expected output:
# - Tamarin prints a "summary of summaries" at the end
# - In the summary, all lemmas are marked as "verified"
# - a `output_distributed.spthy` file under `./out`
make prove MODEL=distributed-time OUT_FILE=
     output_distributed.spthy
```

[Results] Upon completion, Tamarin prints to standard output a summary where each lemma is marked either as *verified*, *falsified*, or *analysis incomplete*. In our case, the experiment can be considered successful if all lemmas are marked as *verified*. The output files *output_{centralized,distributed}.spthy*, contain the full proofs computed by Tamarin.

2) Experiment (E2) - Claim (C2): [Simulations] [20-30 human-minutes + 4.5-5 compute-hours]: The experiment consists in simulating a V2X edge area with a number of vehicles that interact with each other, some of them being malicious. The goal of this simulation is collecting data on revocation times for malicious vehicles, and analyzing their distribution compared to different scenarios and attacker levels. See Sect. VII-A for more information. The simulation runs on a Kubernetes cluster: Our experiments have been carried out on a large cluster with 8 nodes, spawning 400 vehicles and with a total simulation time of around 32 hours. For the artifact evaluation, however, we propose a *scaled-down* configuration that can run locally on a Minikube instance and spawns 50 vehicles, with a total simulation time of around 4 hours. We believe that this configuration is still acceptable since it only affects the quantity of gathered data (i.e., number of revocations and revocation times). The output plots will therefore be comparable to the figures in our paper, although less data will produce less outliers. Besides, given that there is a certain degree of randomicity (due to simulated network conditions), it is impossible to reproduce *exactly* the same results that are shown in the paper: Each simulation, therefore, will produce slightly different box plots.

[Preparation] In a new shell, go to the `simulation` folder. Run the commands below in sequence to set up a new Minikube cluster and build our application.

```
# Create a new Minikube instance (1-5 minutes)
# Expected output:
# - A success message from Minikube
# - kubectl works correctly: try running `kubectl get nodes`
make run_minikube

# Build application from source (3 minutes)
# Expected output: No error messages
make build_minikube
```

[Execution] To run all simulations at once, run the command below. Our scripts will autonomously manage each simulation and collect all data.

```
# Run all simulations (4.5-5 hours)
# Expected output:
# - a `simulations` folder created. `simulations/scenarios`
     contain one file for each run (16 in total)
# - (after 1 minute since start) the log file `simulations/
     out.log` does not show errors and is "SLEEPING"
# - (after 2-3 minutes since start) `kubectl -n v2x get pods
     ` shows all pods in "Running" state
# - (after 4.5-5 hours since start) the log file `
     simulations/out.log` shows "ALL DONE" as last message
# - (after 4.5-5 hours since start) the `simulations/results
     ` folder contains one file for each run (16 in total)
make run_simulations_background CONF=conf/ae.yaml
```

[Results] Assuming that all simulations have been successful, the plots can be generated by running `make plot_all`. The results can be then found in the *simulations/figs* folder. Scenario A1 corresponds to Fig. 5, while the other scenarios are described in our Github repository.

3) Experiment (E3) - Claim (C3): [Size of the PRL] [10-20 human-minutes + 2.5-3.5 compute-hours]: The experiment consists in creating a statistical model for the size of the PRL, where the process of adding and removing pseudonyms can be approximated to a Markov model. Then, based on this model, we plot the expected PRL sizes in terms of percentiles of all possible states, using different parameters such as the number of pseudonyms, the time each pseudonym needs to stay in the PRL (T_{prl}) and the share of attackers in the network. See Sect. VII-B and Appendix B for more information.

[Preparation] In a new shell, go to the `prl` folder.

[Execution] All the data and plots can be computed with one single command: `make all`. Alternatively, each plot can be computed separately according to the instructions below.

```
# probabilities and expected revocations (1-5 seconds)
# Expected output: results printed to standard output
make probabilities

# Fig. 15: transition graph (1-5 seconds)
# Expected output:
# - no errors printed to standard output
# - `tikz-graph.tex` plot under `./plots`
make tikz

# Fig 6: Series over different probabilities (15-20 min)
# Expected output:
# - no errors printed to standard output
# - plots `p-plot_n800_e30.{tex,png}` under `./plots`
make p-plot

# Fig. 16: Series over different number of pseudonyms (25-35
     min.)
# Expected output:
# - no errors printed to standard output
# - plots `n-plot_e30.{tex,png}` under `./plots`
make n-plot

# Fig. 17: Series over different T_prl (90-140 minutes)
# Expected output:
# - no errors printed to standard output
# - plots `t-plot_n800.{tex,png}` under `./plots`
make t-plot

# Fig. 7: Generate distribution for Tv (15-20 minutes)
# Expected output:
# - no errors printed to standard output
# - plots `tv-distribution.{tex,png}` under `./plots`
make tv-distribution
```

[Results] Assuming that all commands have succeeded, all plots can be found in the *plots* folder.