LARP: A Lightweight Auto-Refreshing Pseudonym Protocol for V2X*

Zheng Yang[1], Tien Tuan Anh Dinh[2], Chao Yin[3], Yingying Yao[4], Dianshi Yang[2], Xiaolin Chang[4], and Jianying Zhou[2]

[1] Southwest University, Chongqing, China
[2] Singapore University of Technology and Design, Singapore
[3] Chongqing University of Technology, Chongqing, China
[4] Beijing Jiaotong University, Beijing, China

Abstract. Vehicle-to-everything (V2X) communication is the key enabler for emerging intelligent transportation systems. Applications built on top of V2X require both authentication and privacy protection for the vehicles. The common approach to meet both requirements is to use pseudonyms which are short-term identities. However, both industrial standards and state-of-the-art research are not designed for resource-constrained environments. In addition, they make a strong assumption about the security of the vehicle's on-board computation units. In this paper, we propose a lightweight auto-refreshing pseudonym protocol (LARP) for V2X. LARP supports efficient operations for resource-constrained devices, and provides security even when parts of the vehicle are compromised. We provide formal security proof showing that the protocol is secure. We conduct experiments on a Raspberry Pi 4. The results demonstrate that LARP is feasible and practical.

Keywords: V2X communication, Privacy, Digital signature, Chameleon hash function, Pseudonym

1 Introduction

Vehicles are becoming computers on wheels. They are being equipped with computation and networking units, which can communicate with other devices on the roads or other remote services. Such vehicle-to-everything (V2X) communication underlies intelligent transportation systems (ITS) to enhance traffic safety and efficiency. For example, one of the first ITS systems demonstrated 35% fewer accidents [26]. It is estimated to reduce over 1 million collisions per year or an equivalent of $25.6 billion dollars [27]. The V2X market is estimated to reach 25.72 billion by 2025 [19].

There are two challenges to the wide adoption of V2X. First, information broadcast by a vehicle must be authenticated. In particular, safety-critical applications such as traffic control must trust that messages containing the vehicle

* This is the full version of the paper presented at SACMAT 2022 [].

location, speed, acceleration, and other safety-related information come from the correct vehicles. These messages also provide a close-up view of vehicles in close proximity, enabling enhanced situational awareness [29]. Second, disclosing location information raises privacy concerns, as it subjects the vehicles to long-term tracking [32]. An attacker can collect and correlate messages from a target vehicle to track its owner's habits since there is a strong relationship between vehicles and the owners.

A common approach that addresses the two conflicting challenges above is pseudonyms [11]. A pseudonym is a short-term identity that can provide authentication. Also, by changing pseudonyms frequently, the vehicle can avoid long-term tracking and identity linkage. The industry standards [15, 31] are based on public key infrastructure (PKI) in which a number of CAs issue pseudonym certificates and use CRL to revoke them. To address the performance limitation of CRL (size of the revocation list, and latency), a recent work uses an approach that delays activation of the certificates [29].

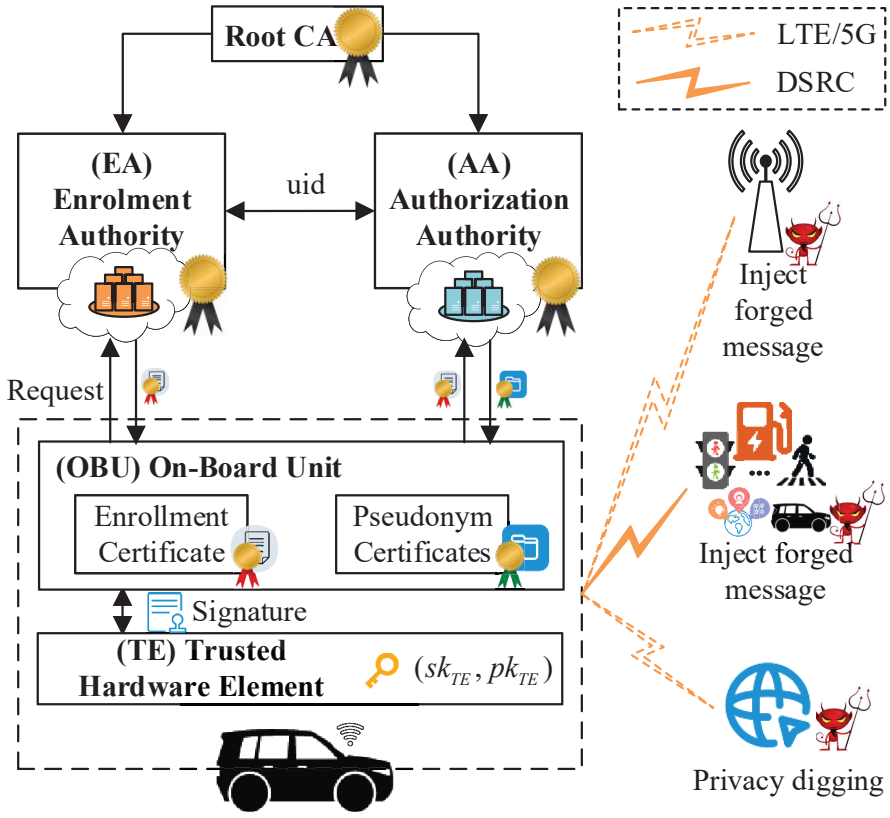

Fig. 1. ETSI Standard V2X PKI Architecture

To better illustrate the limitation of existing approaches for V2X pseudonyms, Figure 1 shows the high-level system model. There are two identity authorities: the enrolment authority (EA) and authorisation authority (AA). Both are certified by a root CA. The EA manages long-term identification and authentication of vehicles. The AA manages the pseudonym authorisation. The separation of EA and AA is to protect privacy, as both are assumed to be run by non-colluding parties. In this model, each vehicle consists of an on-board computation unit (OBU) and a trusted element (TE). The OBU performs most of the computation and communication tasks, while TE is trusted for storing keys and performing cryptographic operations. We notice that most of the recent commercial OBUs (e.g., [7,28]) are integrated with (contactless) smart card interface, so that they can provide personalized and versatile services (e.g., authentication and payments) for different drivers. Usually, the smart card is used as the TE, that is well known to be resource-constrained. To enroll, the vehicle first registers at EA for a unique identifier (uid) and a long-term certificate. It then periodically requests pseudonym certificates related to the uid. To authenticate a message, the TE signs it with the corresponding signing key of the pseudonym certificate.

One limitation of existing works is that they are based on standard signature schemes, for example, ECDSA [16], which is not well suited for resource-constrained devices such as the TEs. Furthermore, they assume that OBU is trusted. In particular, in IFAL [29], the AA sends the pseudonymous keys to the OBU which uses the keys to randomize signatures generated by the TE. In other words, OBU and TE jointly generate the signature. If the OBU is compromised (either via software compromise or physical attacks during servicing), the signatures can be linked back to the original certificate issued by EA, thereby violating vehicle privacy.

Our goal is to design and implement an efficient, secure pseudonym protocol that is suitable for resource-constrained TE and is secure against OBU compromise. To this end, we use the state-of-the-art signature scheme called LiS [35] to build a lightweight auto-refreshing pseudonym protocol called LARP. We choose LiS because it is efficient and supports unlimited data signing without being interrupted at the signer (unlike other lightweight signature schemes, e.g., [36,37]). In LARP, the AA and TE shares a secret seed. We generate pseudonyms by refreshing the random keys that are used to re-randomize the pseudonymous public key. As a result, the OBU in our scheme is not involved for the cryptographic operations, so the protocol is secure even if OBU is compromised. Another benefit of decoupling TE from the OBU is that the TE can be transferred to another OBU without any additional setup.

We have to solve two challenges when designing LARP. First, AA is semi-honest and tries to learn the signing key for the pseudonym. It is possible because LiS uses a universal hash function (UHF) that generates randomness based on a shared secret between AA and TE. To address this, we modify the computation at the AA to compute UHF on the exponent without affecting the signing operations. Second, communication between TE and AA is intermittent, that is, interrupted without warning. In this case, AA may keep updating the ran-

domness and become out of sync with the vehicles, which requires TE to run a synchronization protocol to keep the state consistent with the AA. This synchronization can be costly, which we improve by having a multi-layered structure of the randomness based on different time units, i.e., hour for the first layer, minutes for the second layer, and seconds for the third layer. Our evaluation shows that in the extremely worst case, the TE only needs to run UHF evaluations less than 24 times to sync up with the AA. We note that these two challenges are unique to LARP due to our strong threat model and lightweight signature schemes.

The contributions of this paper are summarized as follows:

- We construct a lightweight auto-refreshing pseudonym protocol named LARP for V2X communications. LARP achieves the auto-refreshing capability via a synchronization protocol. LARP is lightweight and significantly reduces the computation overhead of signature generation and verification by using chameleon hash functions. In addition, LARP supports fast sychronization between the vehicle and the AA by using a three-layered universal hash structure.
- We give a formal security proof that LARP satisfies the standard security and privacy requirements for V2X, under malicious OBU.
- We implement LARP on a Raspberry Pi 4 and evaluate the performance. The results show that it is practical for V2X. For instance, the signing latency in LARP is at least 10× better than that of a state-of-the-art protocol.

The remainder of the paper is organized as follows. Section 2 introduces the preliminaries. Section 3 describes the threat model and the security requirements. Section 4 provides the formal definitions. Section 5 details the design of LARP. Section 6 gives the security proofs. Section 7 discuses the performance of LARP. Section 8 presents related works, and we conclude in Section 9.

2 Preliminaries

This section describes the cryptographic primitives used in LARP.

General Notations. We let κ the security parameter, and $[n] = \{1, \ldots, n\} \subset \mathbb{N}$ be a set of integers ranging from 1 and n. We use $a \xleftarrow{\$} A$ to denote the action of sampling sampling a uniformly random element from a set A. Let $\|$ be an operation to concatenate two strings, $|\cdot|$ be an operation to get the bit-length of a variable, and $\#$ be an operation to get the number of elements in a set.

Table 1 lists the notations used in LARP.

Universal Hash Function. A universal hash function (UH) family [21]: $\mathcal{K} \times \mathsf{A} \to \mathsf{B}$ is a family of hash functions with a low number of collisions even in expectation, where $\mathcal{K}$, A and B denote key, message and output space of UH respectively, and they are determined by the security parameter κ. To instantiate UHF, let q be a large prime number. And we set $\mathcal{K} = \mathsf{A} = \mathsf{B} = \mathsf{Z}_q^*$. The instance of UHF by multiply modular scheme in [4] is as following.

Table 1. The Description of Notations

Notation	Description
$\mathsf{sk}_X, \mathsf{pk}_X$	The entity X's long-term private key and public key respectively.
uid	The unique index mark for an entity.
$\mathsf{pid}_{\mathsf{uid}}^j$	The j-th pseudonym identity of the entity whose unique index mark is uid.
$\mathsf{ppk}_{\mathsf{uid}}^j$	The j-th pseudonym public key of the entity whose unique index mark is uid.
$\mathsf{psk}_{\mathsf{uid}}^j$	The j-th pseudonym secret key of the entity whose unique index mark is uid.
$\mathsf{pvk}_{\mathsf{uid}}^j$	The j-th pseudonym verification key of the entity whose unique index mark is uid.
M	A secret parameter.
$\mathsf{hk1}, \mathsf{hk2}, \mathsf{hk3}$	The universal hash function keys and each of them composes of two sub-keys that are $\mathsf{hk1} = (\mathsf{hk1}_0, \mathsf{hk1}_1)$, $\mathsf{hk2} = (\mathsf{hk2}_0, \mathsf{hk1}_1)$ and $\mathsf{hk3} = (\mathsf{hk3}_0, \mathsf{hk1}_1)$.
r_1	The constantly refreshing secret parameter of first UHF layer.
r_2	The constantly refreshing secret parameter of second UHF layer.
r_3	The constantly refreshing secret parameter of third UHF layer.
k_j	A constantly refreshing secret key.
H	A cryptographic hash function.
sig	A signature.
$\mathsf{T}_{\mathsf{start}}, \mathsf{T}_{\mathsf{end}}$	The start and end time of AA publishing pseudonym verification key, respectively.
$\mathsf{T}_{\mathsf{pvk}}$	The validity period of a pseudonym verification key.
$\mathsf{T}_{\mathsf{gap}}$	The interval of vehicle running signature generation in LARP.
T_t	The life-span of the LARP scheme.
T_s	The runtime of $\mathsf{Sign}_{\mathsf{V2X}}$

The UHF key $\mathsf{hk} = (\mathsf{hk}_0, \mathsf{hk}_1)$ composes of two group elements hk_0 and hk_1, which are randomly chosen from Z_q^*. Given a message m, the hash value x is generated through $\mathsf{x} = \mathsf{UHF}(\mathsf{hk}, m) = \mathsf{hk}_0 \cdot m + \mathsf{hk}_1 \pmod q$.

Definition 1. *We say that a set of hash functions* UH *is universal hash function family if:*

1. *We uniformly choose a hash function* $\mathsf{UHF} \in \mathsf{UH}$ *by sampling a random hash key* $\mathsf{hk} \xleftarrow{\$} \mathcal{K}$*, and*
2. $\forall (x, y) \in \mathsf{A}$*, we have the probability such that* $\Pr[\mathsf{UHF}(\mathsf{hk}, x) = \mathsf{UHF}(\mathsf{hk}, y)] \leq \frac{1}{\#\mathsf{A}}$.

Chameleon Hash Function. A chameleon hash function (CHF) consists of three algorithms.

- $\mathsf{CHKGen}(1^\kappa)$: Let p and q be two large prime numbers such that $p = u \cdot q + 1$ where u is a small integer. This algorithm samples random group generator g of order q in Z_p^* and a random secret key $\mathsf{sk_{CH}} \leftarrow \mathsf{Z}_q^*$, and computes the public key $\mathsf{pk_{CH}} = g^{\mathsf{sk_{CH}}} \pmod p$.
- $\mathsf{CHF}(\mathsf{pk_{CH}}, m, r)$: This algorithm takes the public key $\mathsf{pk_{CH}}$, a message $m \in \mathsf{Z}_q^*$, and a randomness $r \in \mathsf{Z}_q^*$ as input, and outputs a value $y = g^r \cdot (\mathsf{pk_{CH}})^m \pmod p$. Comparing with the chameleon hash function in [18], only the positions of m and r are exchanged, and this change is only conceptual.
- $\mathsf{CHColl}(\mathsf{sk_{CH}}, r', \mathsf{M}, m)$: This algorithm takes the secret key $\mathsf{sk_{CH}}$, and $(r', \mathsf{M}, m) \in \mathsf{Z}_q^*$ as input, outputs $r = \mathsf{M} \cdot \mathsf{sk_{CH}} + r' - m \cdot \mathsf{sk_{CH}}$ such that $\mathsf{CHF}(\mathsf{pk_{CH}}, m, r) = \mathsf{CHF}(\mathsf{pk_{CH}}, \mathsf{M}, r')$.

Given an adversary $\mathcal{A}$ and a chameleon function CH, the collision resistance of CH is defined by the security game $\mathsf{Game}^{\mathsf{CR}}_{\mathcal{A},\mathsf{CH}}(\kappa)$ as follows.

$\mathsf{Game}^{\mathsf{CR}}_{\mathcal{A},\mathsf{CH}}(\kappa)$:
  $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{CHKGen}(1^\kappa)$;
  $(m, r, m', r') \leftarrow \mathcal{A}(\kappa, \mathsf{pk})$;
return 1 if $m' \neq m$ and $\mathsf{CHF}(m', r') = \mathsf{CHF}(m, r)$,
  and 0 otherwise

Definition 2. *Given* $\mathsf{pk_{CH}}, m, r$*, the probability of an adversary* $\mathcal{A}$ *in breaking the security of chameleon hash function under the security parameter* κ *is defined as* $\mathsf{Adv}^{\mathsf{CR}}_{\mathcal{A},\mathsf{CH}}(\kappa) = \Pr[\mathsf{Game}^{\mathsf{CR}}_{\mathcal{A},\mathsf{CH}}(\kappa) = 1]$*. The chameleon hash function is secure when* $\mathsf{Adv}^{\mathsf{CR}}_{\mathcal{A},\mathsf{CH}}(\kappa)$ *is negligible in* κ*.*

Bloom Filter. Bloom Filter (BF) [3] is a probabilistic data structure that provides space efficient storage of a set and that can efficiently test whether an element is a member of the set. The algorithms of a bloom filter are defined as follows.

- $\mathsf{Init}(\mathsf{N}, \epsilon)$: The initialization algorithm takes a set size N to initiate the BF of bit length $1.44\epsilon\mathsf{N}$ with a false positive rate (FPR) of $2^{-\epsilon}$.
- $\mathsf{Insert}(m)$: This algorithm inserts the element m into BF.
- $\mathsf{Check}(m)$: The check algorithm returns 1 if the element m is in BF, and 0 otherwise.

Here we need to the Bloom filter to provide the adversarial resilience property [20] with steady representation. We define a security game (adapted from [20]) $\mathsf{Game}^{\mathsf{AR}}_{\mathcal{A},\mathsf{BF}}(\kappa, T, N, \epsilon)$ that is played between an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and a challenger based on a Bloom filter BF scheme and the parameters (κ, T, N, ϵ), where T is the time of the Bloom filter being used. Note that the parameter T restrict the running time of adversaries in the game. Let $\mathsf{CurrentTime}()$ be a function to get the current system time of the game.

$\mathsf{Game}^{\mathsf{AR}}_{\mathcal{A},\mathsf{BF}}(T, N, \epsilon)$:
  $\mathsf{BF.Init}(N, \epsilon)$;
  $T_s := \mathsf{CurrentTime}()$;
  $IS, st \leftarrow \mathcal{A}_1(T, T_s, N, \epsilon)$;
  $\mathsf{BF.Insert}(m_i)$ for $\forall\ m_i \in IS$
  $m^* \leftarrow \mathcal{A}_2(st, \mathsf{BF})$;
return 1 $m^* \notin IS$ and $\mathsf{CurrentTime}() - T_s \leq T$,
  and 0 otherwise

Definition 3. *The advantage of a PPT adversary $\mathcal{A}$ in breaking the security of a Bloom filter* BF *under the security parameter κ is denoted by* $\mathsf{Adv}^{\mathsf{AR}}_{\mathcal{A},\mathsf{BF}}(\kappa, T, N, \epsilon) := \Pr[\mathsf{Game}^{\mathsf{AR}}_{\mathcal{A},\mathsf{BF}}(\kappa, T, N, \epsilon) = 1]$. *We say* BF *is secure if no PPT adversary has non-negligible advantage* $\mathsf{Adv}^{\mathsf{AR}}_{\mathcal{A},\mathsf{BF}}(\kappa, T, N, \epsilon)$.

3 Overview

This section first presents the system and threat model of LARP. It then discusses the high-level security goals and challenges in realizing these goals.

System model. Our system consists of the vehicle, the identity authorities, and the service providers, as illustrated in Figure 1. The vehicle consists of a trusted hardware element (TE) and an on-board computation unit (OBU). All communication between TE and other entities goes through the OBU. The enrolment authority (EA) and authorisation authority (AA) are certified by a root CA. The EA manages long-term identification and authentication of vehicles. The AA manages the pseudonym authorisation. EA and AA are separated entities, and are assumed to be run by non-colluding parties [14]. The service providers verify the authentication of messages sent from the vehicle.

A pseudonym protocol in V2X consists of three steps.

1. TE registers with the EA, which returns a secret key and unique uid. EA shares the TE's uid with AA.
2. During initialization with AA, the TE sends its uid, secret parameters and an enrollment certificate signed by EA.
3. TE then periodically sends requests to AA for a verification key. The vehicle signs its messages with the key, which can then be checked by the verifier.

Threat model. TE is trusted to protect the device's private keys and to perform cryptographic operations. TE also has an internal clock that is loosely synchronized with those of other TEs. We assume no side-channel attacks and operations within the TE are secure. The communication between TE and the authorities is secure.

The OBU is malicious. It controls the communication between TE and other entities, and therefore can intercept, inject, and tamper with the communication. We do not consider denial of service attacks, as they can be detected by the user.

We note that state-of-the-art solutions, namely IFAL, assume honest OBU, since their protocols rely on both TE and OBU to generate signatures. When OBU is compromised or just out of function, it may simply broadcast TE's signatures and its public key, allowing long-term tracking of the vehicle.

EA and AA are non-colluding authorities. Both can see all messages in the network. EA is honest-but-curious and wants to link messages belonging to the same vehicle. AA is also honest-but-curious. It follows the protocol, but it wants to impersonate the vehicle by learning the signing keys. But EA and AA do not collude with each other. The verifiers, including RSU, pedestrians, signal lights, Internet services, etc., are malicious. They provide the correct safety services to the vehicle but want to track and impersonate vehicles based on the messages.

Goals and Challenges. We mainly have the following security goals for LARP:

- ***Authentication***. Messages generated by the vehicle are authenticated, that is, the attacker cannot forge messages from a non-compromised vehicle (whose signing key is not exposed to the attackers).
- ***Pseudonymity***. A message generated by the vehicle does not reveal its long-term identity.

We note that related works also support *accountability* property, which allows a party to reveal the long-term identity of misbehaved vehicles [29]. However, this property is only possible when EA and AA share more information than uid. We consider this as a strong assumption and outside of our threat model.

To fit in the V2X environment, LARP is motivated to handle two challenges.

- ***Poor and unreliable network***. V2X communication is not reliable, due to interference, vehicle going out of range, or simply not running. As a consequence, the shared state between the vehicle and authorities may go out of sync. This precludes interactive protocols in which the vehicle communicates with the authorities for every message.
- ***Resource constraint***. The protocol must run on vehicle's embedded units with low CPU, memory, communication, and storage capability.

We address the first challenge by designing an efficient synchronization protocol allowing the vehicle and authorities to sync up their shared secret quickly. The protocol is based on a multi-layer structure. Each layer corresponds to a different time granularity (for instance, second, minute, hours, etc.). We address the second challenge by building upon existing lightweight signature schemes. In particular, we modify LiS [35] to use the three-layered structure, and to use an auto-refreshing pseudonym key to randomize the signing keys.

4 V2X Formal Model

V2X Scheme. Here we review the notation of a V2X scheme that is adapted from [29] (with the identical security properties). A V2X scheme Π consists of the following efficient algorithms:

- $\mathsf{CreatePKI}(1^\kappa)$: This is a public parameter generation algorithm which takes as input the security parameter 1^κ, and outputs the private and public key pair $(\mathsf{sk}, \mathsf{pk})$ of an authority (i.e., CA, EA, and AA). The public parameters (PP) include the public keys of the root CA, EA, and AA. The private parameters (SP) are the corresponding private keys.
- $\mathsf{CreateVehicle}(c)$: This is a key generation algorithm for a vehicle, which takes as input a vehicle reference c, and outputs a unique identifier (uid) in the V2X system, and the corresponding the secret key and public key pair $(\mathsf{sk}_{\mathrm{TE}}, \mathsf{pk}_{\mathrm{TE}})$ for TE. We assume that each vehicle is uniquely identified by a unique reference c_i for enrollment (See more discussion about vehicle reference in [29]).
- $\mathsf{CreatePSK}(\mathsf{sk}_{\mathrm{TE}}, j)$: This is a pseudonym secret key generation algorithm for a TE, which takes as input the secret key of a TE and the index j, and outputs the j-th pseudonym secret key $\mathsf{psk}^j_{\mathrm{TE}}$ and the corresponding pseudonym identity $\mathsf{pid}^j_{\mathsf{uid}}$. We assume that each pseudonym secret/verification key pair is only valid for time $\mathsf{T}_{\mathsf{pvk}}$ for privacy requirement. That is, the life-span T_t of the V2X scheme should be divided into multiple time slots $\{T_i\}_{i\in[\mathsf{T}_t/\mathsf{T}_{\mathsf{pvk}}]}$.
- $\mathsf{CreatePPK}(\mathsf{pk}_{\mathrm{TE}}, j, \mathsf{aux})$: The pseudonym verification key generation algorithm takes as input the public key of a TE $\mathsf{pk}_{\mathrm{TE}}$, the index j, and some auxiliary input aux (e..g, cached state of AA), and outputs the j-th pseudonym verification key $\mathsf{pvk}^j_{\mathsf{uid}}$ and the corresponding pseudonym identity $\mathsf{pid}^j_{\mathsf{uid}}$.
- $\mathsf{Sign}_{\mathsf{V2X}}(\mathsf{psk}_{\mathsf{uid}}, m)$: This signing algorithm takes as input the pseudonym signing key $\mathsf{psk}_{\mathsf{uid}}$ and a message m, and outputs a signature sig for m.
- $\mathsf{Verify}_{\mathsf{V2X}}(\mathsf{sig}, \mathsf{pvk}_{\mathsf{uid}}, m)$: This verification algorithm takes input the pseudonym verification key $\mathsf{pvk}_{\mathsf{uid}}$, the message m and the signature sig, and outputs 1 if sig is a valid signature from the verification key pvk and 0 otherwise.

The V2X scheme Π would also run the following protocols based on the above algorithms:

- $\mathsf{EnrolVehicle}$ protocol is an interactive protocol between a vehicle and EA that is executed to register a vehicle. This protocol may run the algorithm $\mathsf{CreateVehicle}$ to enroll a vehicle and generate uid and the secret and public key pair of the vehicle.
- $\mathsf{AuthoriseVehicle}$ protocol is an interactive protocol between a vehicle and AA that is used to generate the pseudonym public keys for vehicles. This protocol may run the algorithms $\mathsf{CreatePPK}$ and $\mathsf{CreatePSK}$ to jointly generate the pseudonym secret and verification key pair.

V2X Security. This subsection provides the security formalization of a V2X scheme. To formalize the fundamental security property of V2X scheme, we define the authentication game $\mathsf{Game}^{\mathsf{Auth}}_{\mathcal{A},\Pi}$ in the following.

In the authentication game, $\mathcal{A}$ will interact with the V2X scheme Π involving N_v vehicle , and at most N_p pseudonym pubic key for each vehicle. In the game, the adversary can have access to two oracles $\mathcal{O}_{\mathsf{EV}}()$ and $\mathcal{O}_{\mathsf{AV}}()$, which allows to the (passive) adversary to execute the V2X protocols $\mathsf{EnrolVehicle}$

and AuthoriseVehicle respectively to get the protocol transcripts (which may include the pseudonym public key related information). We shall use a list PSK to record all pseudonym secret keys of vehicles generated during AuthoriseVehicle oracle queries. Note that a psk can be retrieved in terms of pid. Moreover, we let $\mathcal{O}_{\mathsf{S}}(\mathsf{PSK}, \mathsf{pid}, m)$ be a signing oracle which first finds out the corresponding pseudonym secret key $\mathsf{psk} \in \mathsf{PSK}$ by pid, and then runs $\mathsf{Sign}_{\mathsf{V2X}}(\mathsf{psk}, m)$ to generate a signature sig for the input message m. In the meantime, we use a list SL to record the tuple $(\mathsf{sig}, m, \mathsf{pvk}^j)$ that corresponds to each signing oracle query. In addition, we assume that the adversary can ask at most $l = l(\kappa)$ signing oracle queries, where l is defined by a function $l(\kappa)$ in κ.

The goal of the adversary in the game is to forge a signature of a pseudonym verification key. For simplicity, we let $\mathcal{O}_{\mathsf{V2X}}$ be a set of oracles $\{\mathcal{O}_{\mathsf{EV}}(), \mathcal{O}_{\mathsf{AV}}(), \mathcal{O}_{\mathsf{S}}(\mathsf{PSK}, \cdot, \cdot)\}$. We will say that a pseudonym verification key pvk is valid if it is generated by oracle $\mathcal{O}_{\mathsf{AV}}()$.

$\mathsf{Game}^{\mathsf{Auth}}_{\mathcal{A},\Pi}(\kappa)$:
  $(PP, SP) \leftarrow \mathsf{CreatePKI}(1^{\kappa})$;
  $(\mathsf{sig}^*, m^*, \mathsf{pvk}^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{V2X}}}(PP)$;
return 1 if $(\mathsf{sig}^*, m^*, \mathsf{pvk}^*) \notin \mathsf{SL}$ and pvk^* is valid and
  $\mathsf{Verify}(\mathsf{sig}^*, \mathsf{pvk}^*, m^*) = 1$, and 0 otherwise

Definition 4 (V2X Security). *The advantage of an efficient adversary $\mathcal{A}$ in breaking the security of a V2X scheme Π under the security parameter κ is defined as* $\mathsf{Adv}^{\mathsf{Auth}}_{\mathcal{A},\Pi}(\kappa) = \Pr[\mathsf{Game}^{\mathsf{Auth}}_{\mathcal{A},\Pi}(\kappa) = 1]$. *The V2X scheme Π is secure, which means that no efficient adversary has non-negligible advantage* $\mathsf{Adv}^{\mathsf{Auth}}_{\mathcal{A},\Pi}(\kappa)$.

V2X Privacy. We first discuss about the achievable privacy of a V2X scheme and the limitations. The functional requirement that vehicles frequently broadcast highly unique location and trajectory data to receivers limited only by communication distance [6] cannot be protected through cryptography. The completely unlinkable V2X signatures using a distinct pseudonym for each V2X message are still vulnerable to attacks that utilize the connection between vehicle speed and location at different time [2, 13].

Unlike the above, the contents of the broadcast V2X messages and their digital signatures are considered respectively in the V2X scheme. This allows the privacy leakage of the V2X system in a way that does not depend on human behavior or vendor-specific implementations to be quantified.

We define the V2X privacy from the perspective of pseudonymity and separate it from the functional content of collaborative awareness messages. The captured realistic adversaries only have a partial overview of the entire environment, which will lead to that the adversaries will experience uncertain periods during which the target vehicle does not appear to broadcast its position or trajectory. If there is enough noise in other vehicle forms, the adversary will not be sure whether the target vehicle can be re-identified.

V2X Pseudonymity. It is defined as whether a pseudonym can be linked to the canonical identity of a vehicle. The contents of the V2X messages are not allowed to provide adversaries with a non-negligible advantage on pseudonymity breach. Each vehicle corresponds to a unique vehicle reference. Vehicles broadcast V2X messages using the methods and the pseudonym scheme specified in the underlying V2X scheme. In the following, we define an pseudonymity game $\mathsf{Game}^{\mathsf{Pseud}}_{\mathcal{A},\mathsf{V2X}}$ that is played with an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$.

In the game, there are two phases. Firstly, the game simulator creates the system parameters and secret and public keys of the vehicles, and then provides the all public information to the adversary $\mathcal{A}_0$. After a polynomial number of queries given by $\mathcal{O}_{\mathsf{V2X}}$, $\mathcal{A}_0$ must output two target vehicle references c_0^* and c_1^*, which must have never been input to the $\mathsf{EnrolVehicle}$ oracle. Then, the simulator randomly samples a random bit b, and create the public and secret key pair for c_b^* by running $\mathsf{CreateVehicle}(PP, SP, c_b^*)$ and returns $\mathsf{pk}^*_{\mathrm{TE}_b^*}$ as the challenge, where TE_b^* is referenced by c_b^*. Both vehicle references c_0^* and c_1^* would become invalid, i.e., they will never be handled by $\mathcal{O}_{\mathsf{EV}}$ oracle. After time $\mathsf{T}_{\mathsf{pvk}}$ (i.e., the pseudonym verification key of c_b^* starts working), the adversary $\mathcal{A}_1$ keeps asking oracles in $\mathcal{O}_{\mathsf{V2X}}$ at the second phase based on $\mathsf{pk}_{\mathrm{TE}_b^*}$. Eventually, $\mathcal{A}_1$ outputs a guess bit b'. The adversary wins the game if $b' = b$.

$\mathsf{Game}^{\mathsf{Pseud}}_{\mathcal{A},\mathsf{V2X}}(\kappa)$:
$(PP, SP) \leftarrow \mathsf{CreatePKI}(1^\kappa)$;
$b \xleftarrow{\$} \{0,1\}$;
$(st, c_0^*, c_1^*) \leftarrow \mathcal{A}_0^{\mathcal{O}_{\mathsf{V2X}}}(PP)$;
$(\mathsf{sk}^*_{\mathrm{TE}_b^*}, \mathsf{pk}^*_{\mathrm{TE}_b^*}) \leftarrow \mathsf{CreateVehicle}(c_b^*)$;
invalidate (c_0^*, c_1^*) and wait for time $\mathsf{T}_{\mathsf{pvk}}$;
$b' \leftarrow \mathcal{A}_1^{\mathcal{O}_{\mathsf{V2X}}}(PP, st, c_b^*, \mathsf{pk}^*_{\mathrm{TE}_b^*})$;
return 1 if $b' = b$, and 0 otherwise

Definition 5 (V2X Pseudonymity). *The advantage of an efficient adversary* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ *in breaking the privacy of a V2X scheme* $\mathsf{V2X}$ *under the security parameter* κ *is defined as* $\mathsf{Adv}^{\mathsf{Pseud}}_{\mathcal{A},\mathsf{V2X}}(\kappa) = |\Pr[\mathsf{Game}^{\mathsf{Pseud}}_{\mathcal{A},\mathsf{V2X}}(\kappa) = 1] - 1/2|$. *The V2X scheme* $\mathsf{V2X}$ *is pseudonymity conscious, which means that no efficient adversary has a non-negligible advantage* $\mathsf{Adv}^{\mathsf{Pseud}}_{\mathcal{A},\mathsf{V2X}}(\kappa)$.

5 LARP

In this section, we introduce a lightweight auto-refreshing pseudonym scheme LARP for V2X communication. We stress that LARP conforms to ETSI ITS standards as the system architecture remains unchanged.

5.1 Overview

There are three important system parameters. An instance of LARP has a lifetime specified by $\mathsf{T}_{\mathsf{start}}$ and $\mathsf{T}_{\mathsf{end}}$. We use $\mathsf{T}_{\mathsf{pvk}}$ to denote the validity period of a verification key (or *epoch*). The ETSI standard recommends setting $\mathsf{T}_{\mathsf{pvk}}$ to five minutes. $\mathsf{T}_{\mathsf{gap}}$ is the period between *runs*. For example, if the vehicle stops at time t_i and starts again at time t_j, then $\mathsf{T}_{\mathsf{gap}} = (t_j - t_i)$. Fig. 2 illustrates how LARP is divided into consecutive $\mathsf{T}_{\mathsf{pvk}}$ intervals. The vehicle may be active (i.e., running) over many intervals. However, when $\mathsf{T}_{\mathsf{gap}} > 0$, the vehicle runs the key generation algorithm to generate the latest verification key.

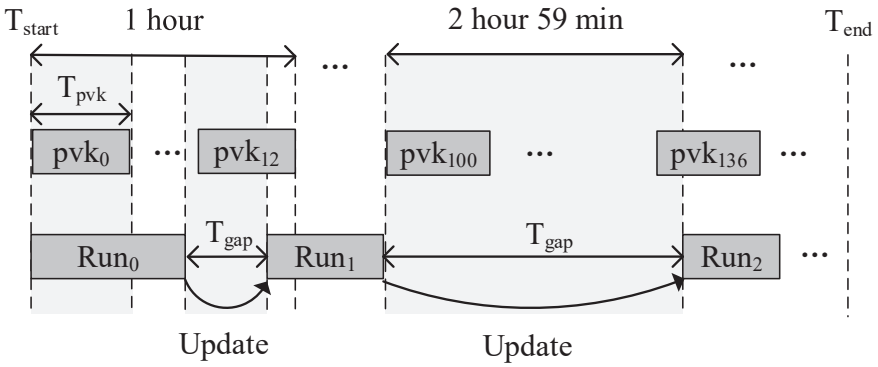

Fig. 2. Overview of LARP Usage Time.

LARP adapts the lightweight signature scheme LiS_2 [35] to construct the LARP scheme. A signature of LARP is generated entirely by the TE without input from the OBU. In LARP, the unique uid is bound to TE instead of OBU in IFAL, so it can fit in a more flexible real-world scenario in which many distinct drivers share the vehicle (e.g., a family). Since TE (e.g., smart cards) can be replaced, we need the OBU to serve as a card reader to authenticate a TE. To generate the pseudonym verification keys, we let TE and AA share a symmetric key k_0 from which they can independently create more pseudonym verification keys.

LiS_2 uses pre-computation to generate verification points. A verification point is the output of the chameleon hash function over a dummy message M and a random value r'_i, i.e., $\mathsf{CHF}(\mathsf{pk}_{\mathsf{CH}}, M, r'_i)$. More specifically, a UHF function is used as a pseudo-random sequence generator to update the random values, that is, $r'_i := \mathsf{UHF}(\mathsf{hk}, r'_{i-1}) = \mathsf{hk}_0 \cdot r'_{i-1} + \mathsf{hk}_1 \bmod q$, where q is a big prime number and $\mathsf{hk} = (\mathsf{hk}_0, \mathsf{hk}_1)$ is a hash key of UHF. These randomnesses should be kept privately by a fully trusted key generation center (KGC) in LiS_2. Otherwise, an attacker can extract the signing key from the randomnesses. However, in the V2X setting, the AA might be curious about signing keys of TE, so it cannot have access to these values. Our scheme addresses this by letting AA compute

the UHF on the exponent only, i.e., $g^{\mathsf{hk}_0 \cdot r'_{i-1} + \mathsf{hk}_1}$ where g is a group generator. Note that the AA can only have $g^{r'_{i-1}}$ and cannot know hk_0 in plaintext (for the above security reason). So we further split the first sub-key of UHF hk_0, into two secret shares, i.e., $\mathsf{hk}_0 \cdot \alpha$ and $1/\alpha$ (such that the hk_0 can be reconstructed from them), and distribute them to the EA and AA, respectively, where α is a random value. That is, with these shares, EA and AA can jointly compute the secret exponentiation without knowing hk_0.

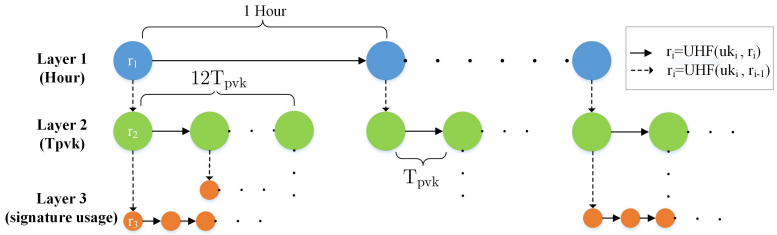

Fig. 3. Three-layered UHF Tree Diagram.

Unlike LiS_2, which assumes uninterrupted communication (for continuous message authentication), LARP needs to handle the unpredictable stopping of a vehicle (e.g., at lunchtime) as shown in Figure 2. A stop causes the TE's randomness r_i to be out of date. To catch up, the TE could update the randomness by computing the value for every epoch, as in LiS_2. This approach is inefficient. For example, a one-hour stop may result in thousands of UHF computations (or more than 30 seconds on Raspberry Pi 4 platform). Instead, we propose a three-layered UHF approach that quickly skips over expired values. These layers are divided in terms of hours (first layer), pseudonym verification key valid time $\mathsf{T}_{\mathsf{pvk}}$ (second layer), and the signature generation (third layer). Each layer uses a different UHF key. The initial input of the second or the third layer is from the upper layer. Suppose the vehicle stops for 1 hour and 50 minutes. To sync up the randomness, the TE needs to compute one UHF (i.e., determined by 1 hour) at the first layer, then use the output to compute $50/\mathsf{T}_{\mathsf{pvk}}$ UHFs at the second layer and compute one final UHF at the third layer. We show a high-level overview of the three-layered UHF approach in Fig. 3. This approach results in a significant reduction in UHF computation.

5.2 Algorithms of LARP

In this section, we describe the algorithms and protocols of LARP. Our construction leverages on a hash function $\mathsf{H} : \{0,1\}^* \rightarrow \mathsf{Z}_q^*$, a universal hash function family, and a chameleon hash function. In the following, we ignore CreatePKI algorithm which is implemented with standard digital signature schemes (e.g., ECDSA). We assume the output of this algorithm at the EA and the AA are $(\mathsf{sk}_{\mathrm{EA}}, \mathsf{pk}_{\mathrm{EA}})$ and $(\mathsf{sk}_{\mathrm{AA}}, \mathsf{pk}_{\mathrm{AA}})$, respectively.

We describe the signing procedure of EA and AA without mentioning the specific signature schemes.

$\mathsf{CreateVehicle}(c)$. This algorithm is run by the vehicle during registration with the EA, and involves the following steps.

1. Generate a key pair by running the key generation function of chameleon hash function $(\mathsf{sk_{CH}}, \mathsf{pk_{CH}}) \leftarrow \mathsf{CHKGen}(1^\kappa)$.
2. Sample three random keys $(\mathsf{hk1}, \mathsf{hk2}, \mathsf{hk3}) \xleftarrow{\$} \mathcal{K}$ for the the universal hash function family UHF. Each key corresponds to a layer in the multi-layer UHF structure.
3. Sample a random message $\mathsf{M} \xleftarrow{\$} \mathsf{Z}_q^*$.
4. Sample an initial random value $r_1 \xleftarrow{\$} \mathsf{Z}_q^*$ of the first UHF layer, a pseudonym seed $\mathsf{k}_0 \xleftarrow{\$} \mathsf{Z}_q^*$, and a random value $\alpha \xleftarrow{\$} \mathsf{Z}_q^*$ for blinding the hash key of UHF.
5. Generate the second layer and the third layer randomnesses as $r_2 = \mathsf{UHF}(\mathsf{hk1}, r_1)$ and $r_3 = \mathsf{UHF}(\mathsf{hk2}, r_2)$.
6. Compute secret/public key pair $\mathsf{sk_{TE}} := (\mathsf{sk_{CH}}, \mathsf{hk1}, \mathsf{hk2}, \mathsf{hk3}, r_1, r_2, r_3, \mathsf{M} \cdot \mathsf{sk_{CH}}, \mathsf{k}_0, \alpha)$ and $\mathsf{pk_{TE}} := (\mathsf{pk_{CH}}, (\mathsf{pk_{CH}})^{\mathsf{M}}, \mathsf{hk1}_0, \mathsf{hk2}_0, g^{\mathsf{hk1}_1}, g^{\mathsf{hk2}_1}, g^{\mathsf{hk3}_1}, g^{r_1}, g^{r_2}, g^{r_3})$ for TE.
7. Keep the index of pseudonym epoch j as a mutable state, which can be determined by the current time.
8. Share k_0 and $\mathsf{uk}^* = \alpha \cdot \mathsf{hk3}_0 \pmod q$ with AA over a secure channel (e.g., Transport Layer Security (TLS) protocol [5, 23]).

$\mathsf{CreatePSK}(\mathsf{psk}'_{\mathsf{uid}}, j)$. We change the input secret key here for efficiency concerns. It is easy to see that $\mathsf{psk}'_{\mathsf{uid}}$ can be computed from $\mathsf{sk_{TE}}$. The TE of the vehicle runs this algorithm to generate pseudonym secret key for the j-th epoch, $\mathsf{psk}^j_{\mathsf{uid}}$, based on the stored, most recently used pseudonym secret key $\mathsf{psk}'_{\mathsf{uid}}$ and the time T_l when it is last used. This algorithm involves the following steps:

1. Get a vehicle stop time $\mathsf{T_{gap}}$ based on current time T_c as $\mathsf{T_{gap}} = \mathsf{T}_c - \mathsf{T}_l$. Note that the main difference between two pseudonym secret keys are the random values r_1, r_2 and r_3. So we mainly update the pseudonym key and randomness in $\mathsf{psk}'_{\mathsf{uid}}$ to yield the up-to-date pseudonym secret key $\mathsf{psk}^j_{\mathsf{uid}}$.
2. Call **Algorithm 1** to synchronize the parameters with AA based on the most recently used pseudonym secret key $\mathsf{psk}'_{\mathsf{uid}}$ and the vehicle stop time $\mathsf{T_{gap}}$, and generate the updated pseudonym secret key $\mathsf{psk}^j_{\mathsf{uid}}$ and pseudonym identity $\mathsf{pid}^j_{\mathsf{uid}}$. The algorithm first gets the hours in $\mathsf{T_{gap}}$ and the number of pseudonym epochs in the remainder of $\mathsf{T_{gap}}$ that is less than an hour, which are denoted as $\mathsf{n}_1 = \frac{\mathsf{T_{gap}}}{1\mathsf{hr}}$ and $\mathsf{n}_2 = \frac{\mathsf{T_{gap}} - \mathsf{n}_1 \cdot 1\mathsf{hr}}{\mathsf{T_{pvk}}}$, respectively. We let T_s be the maximum time cost for computing a signature at TE. Then, we can further obtain $\mathsf{n}_3 := \frac{\mathsf{T_{gap}} - \mathsf{n}_1 \cdot 1\mathsf{hr} - \mathsf{n}_2 \cdot \mathsf{T_{pvk}}}{\mathsf{T}_s}$ that is the number of UHF evaluations in the third layer of the UHF tree. All the time values in the paper will be implicitly converted to the same unit in different calculations unless the calculations are specifically defined. The algorithm will execute n_i times UHF to skip the randomness used in the i-th layer. Totally, there are $\mathsf{n}_1 + \mathsf{n}_2 + \mathsf{n}_3$ UHF operations executed to update the random values r_1, r_2 and r_3 in the pseudonym secret key $\mathsf{psk}^j_{\mathsf{uid}}$. To update the pseudonym identity $\mathsf{pid}^j_{\mathsf{uid}}$ for

the j-th epoch, it firstly generates the j-th seed $\mathsf{k}_j = \mathsf{H}(\mathsf{k}_0||\mathsf{uid}||j)$ and then compute $\mathsf{pid}^j_{\mathsf{uid}} = \mathsf{H}(\mathsf{k}_j||\mathsf{uid})$.

Algorithm 1 $\mathsf{Update}(\mathsf{psk}'_{\mathsf{uid}}, \mathsf{T}_{\mathsf{gap}}, j)$

$\mathsf{n}_1 = \frac{\mathsf{T}_{\mathsf{gap}}}{1\mathsf{hr}}$ \\ Number of UHF evaluations in the first layer
$\mathsf{n}_2 = (\mathsf{T}_{\mathsf{gap}} - \mathsf{n}_1 \cdot 1\mathsf{hr})/\mathsf{T}_{\mathsf{pvk}}$ \\ Number of UHF evaluations in the second layer
$\mathsf{n}_3 := \frac{\mathsf{T}_{\mathsf{gap}} - \mathsf{n}_1 \cdot 1\mathsf{hr} - \mathsf{n}_2 \cdot \mathsf{T}_{\mathsf{pvk}}}{\mathsf{T}_s}$ \\ Number of UHF evaluations in the third layer
for $\mathsf{x} = 1$ to n_1 **do**
  $r_1 = \mathsf{hk1}_0 \cdot r_1 + \mathsf{hk1}_1$ \\ Update first UHF layer
end for
if $\mathsf{n}_1 \neq 0$ **then**
  $r_2 = \mathsf{hk2}_0 \cdot r_1 + \mathsf{hk2}_1$ \\ Switch to a new branch
end if
for $\mathsf{x} = 1$ to n_2 **do**
  $r_2 = \mathsf{hk2}_0 \cdot r_2 + \mathsf{hk2}_1$ \\ Update second UHF layer
end for
if $\mathsf{n}_1 \neq 0$ **or** $\mathsf{n}_2 \neq 0$ **then**
  $r_3 = \mathsf{hk3}_0 \cdot r_2 + \mathsf{hk3}_1$ \\ Switch to a new branch
end if
for $\mathsf{x} = 0$ to n_3 **do**
  $r_3 = \mathsf{hk3}_0 \cdot r_3 + \mathsf{hk3}_1$ \\ Update the third UHF layer
end for
$\mathsf{k}_j = \mathsf{H}(\mathsf{k}_0||\mathsf{uid}||j)$;
$\mathsf{pid}^j_{\mathsf{uid}} = \mathsf{H}(\mathsf{k}_j||\mathsf{uid})$ \\ Update the pseudonym identity
$\mathsf{psk}^j_{\mathsf{uid}} = (\mathsf{k}_j, r_1, r_2, r_3, \mathsf{sk}_{\mathrm{TE}})$
return $\mathsf{psk}^j_{\mathsf{uid}}, \mathsf{pid}^j_{\mathsf{uid}}$

$\mathsf{CreatePPK}(\mathsf{pk}^{\mathrm{AA}}_{\mathrm{TE}}, j, \mathsf{aux})$: This algorithm is run by AA. Let $\mathsf{pk}^{\mathrm{AA}}_{\mathrm{TE}}$ be the public key of TE kept by AA, which encompasses the updated public randomness $g^{r_1}, g^{r_2}, g^{r_3}$, and the parameters $(\mathsf{pk}_{\mathsf{CH}})^{\mathsf{M}}$, g^{hk3_1} and a mutable j. Here we specifically define the auxiliary value $\mathsf{aux} = (\mathsf{k}_j, \mathsf{uid}, \mathsf{uk}^*)$. The algorithm has the following steps.

1. Compute the pseudonym identity and pseudonym public key for the vehicle through $\mathsf{pid}^j_{\mathsf{uid}} = \mathsf{H}(\mathsf{k}_j||\mathsf{uid})$ and $\mathsf{ppk}^j_{\mathsf{uid}} = (\mathsf{pk}_{\mathsf{CH}})^{\mathsf{k}_j}$.
2. Initialize a new Bloom filter instance BF_j by $\mathsf{BF}_j.\mathsf{Init}(l_s, \epsilon)$, where $l_s = \lceil \frac{\mathsf{T}_{\mathsf{pvk}}}{\mathsf{T}_s} \rceil$;
3. For $i \in [l_s]$, AA does the following:
   - Send g^{r_3} to EA who will compute a response $\mathsf{U} = g^{r_3 \cdot \mathsf{uk}}$ for AA, where $\mathsf{uk} = 1/\alpha \pmod q$ is a secret share (which is used to remove the secret α from uk^* on the exponent of a value) obtained from the corresponding vehicle in the $\mathsf{EnrolVehicle}$ protocol (described later);
   - Refresh g^{r_3} as $g^{r_3} = (\mathsf{U})^{\mathsf{uk}^*} \cdot g^{\mathsf{hk3}_1}$;
   - Compute a verification point $\mathsf{vp}_i = (\mathsf{pk}_{\mathsf{CH}})^{\mathsf{M}} \cdot g^{r_3}$, and insert vp_i into the j-th Bloom filter instance as $\mathsf{BF}_j.\mathsf{Insert}(\mathsf{vp}_i)$.

4. Return the pseudonym identity $\mathsf{pid}^j_{\mathsf{uid}}$ and the pseudonym verification key $\mathsf{pvk}^j_{\mathsf{uid}} = (\mathsf{ppk}^j_{\mathsf{uid}}, \mathsf{BF}_j)$ with its signature which can be downloaded from AA by the public.

$\mathsf{Sign}_{\mathsf{V2X}}(\mathsf{psk}'_{\mathsf{uid}}, m)$: This algorithm is run by the vehicle (the TE) to generate signatures for V2X messages. To sign a message m, it does the following steps:

1. Run $\mathsf{CreatePSK}(\mathsf{psk}'_{\mathsf{uid}}, j)$ to get the pseudonym secret key $\mathsf{psk}^j_{\mathsf{uid}}$ based on the most recently used pseudonym secret key $\mathsf{psk}'_{\mathsf{uid}}$.
2. Execute **Algorithm 2** to generate the signature sig.

Algorithm 2 $\mathsf{Sign}(\mathsf{psk}^j_{\mathsf{uid}}, m)$

$\mathsf{R} \xleftarrow{\$} \mathsf{Z}^*_q$
$\mathsf{h} = \mathsf{H}(m||\mathsf{R})$
$\mathsf{s} = \mathsf{M} \cdot \mathsf{sk}_{\mathsf{CH}} + r_3 - \mathsf{k}_j \cdot \mathsf{h} \cdot \mathsf{sk}_{\mathsf{CH}}$ \\ Signature generation
$\mathsf{sig} = (\mathsf{s}, \mathsf{R})$
return sig

$\mathsf{Verify}_{\mathsf{V2X}}(\mathsf{sig}, \mathsf{pvk}^j_{\mathsf{uid}}, m)$: To verify a $\mathsf{sig} = (\mathsf{s}, \mathsf{R})$ for a message m, the verifier does the following steps:

1. Compute a hash $\mathsf{h} = \mathsf{H}(m||\mathsf{R})$.
2. Verify the point $\mathsf{vr} = (\mathsf{ppk}^j_{\mathsf{uid}})^{\mathsf{h}} \cdot g^{\mathsf{s}}$, where $\mathsf{ppk}^j_{\mathsf{uid}} \in \mathsf{pvk}^j_{\mathsf{uid}}$.
3. Return $\mathsf{BF}_j.\mathsf{Check}(\mathsf{vr})$ for $\mathsf{BF}_j \in \mathsf{pvk}^j_{\mathsf{uid}}$.

Protocols. The end-to-end protocols of LARP are described below.

EnrolVehicle: A vehicle interacts with the EA as follows:

1. The vehicle c first runs $\mathsf{CreateVehicle}(c)$ algorithm to create its secret and public key pair $(\mathsf{sk}_{\mathrm{TE}}, \mathsf{pk}_{\mathrm{TE}})$ for its TE.
2. The vehicle sends an authRequest, which contains $\mathsf{pk}_{\mathrm{TE}}$, $\mathsf{uk} = 1/\alpha \pmod q$, and its canonical identity reference c to EA through a secure channel.
3. Upon receiving authRequest, EA waits for the out-of-band document to assert the vehicle registrant. Then, EA generates a unique uid as a shared pseudonymous reference between the EA and AA.
4. EA signs a enrollFile which contains uid and $\mathsf{pk}_{\mathrm{TE}}$, and returns the signed enrollFile to TE.
5. Finally, EA maintains a database storing the tuples in the form $(\mathsf{uid}, c, \mathsf{pk}_{\mathrm{TE}}, \mathsf{uk})$ regarding a vehicle.

AuthoriseVehicle: A vehicle may run this protocol with AA to book pseudonym certificates as follows:

1. A vehicle submits its $\mathsf{enrollFile}$ (obtained from $\mathsf{EnrolVehicle}$ protocol) and uk^* to AA to request pseudonym certificates from the start time $\mathsf{T_{start}}$ and end time $\mathsf{T_{end}}$ for requesting pvk. Note that $\mathsf{T_{start}}$ and $\mathsf{T_{end}}$ are fixed (e.g., $\mathsf{T_{start}}$= 00:00:00 and $\mathsf{T_{end}}$ = 23:59:59) for privacy concerns.
2. AA computes $\mathsf{n}_1 = \frac{\mathsf{T}_t}{\mathsf{1hr}}$ and $\mathsf{n}_2 = \frac{\mathsf{1hr}}{\mathsf{T_{pvk}}}$, where $\mathsf{T}_t = T_{\mathsf{end}} - \mathsf{T_{start}}$.
3. For $j \in [\frac{\mathsf{T}_t}{\mathsf{T_{pvk}}}]$, AA does the following:
   - If $(j \mod \mathsf{n}_2) = 0$ (meaning the next hour starts), then update the first layer's randomness $\mathsf{n}_1 = \frac{\mathsf{T_{end}} - \mathsf{T_{start}}}{\mathsf{1hr}}$ and the second layer's randomness $g^{r_2} = (g^{r_1})^{\mathsf{hk2}_0} \cdot g^{\mathsf{hk2}_1}$; Otherwise, update the second layer's randomness $g^{r_2} = (g^{r_2})^{\mathsf{hk2}_0} \cdot g^{\mathsf{hk2}_1}$;
   - Send g^{r_2} to EA and receive back $\mathsf{U} = g^{r_2 \cdot \mathsf{uk}}$;
   - Compute $g^{r_3} = (\mathsf{U})^{\mathsf{uk}^*} \cdot g^{\mathsf{hk3}_1}$ which is the first randomness of the corresponding branch in the third layer;
   - Compute the j-th seed $\mathsf{k}_j = \mathsf{H}(\mathsf{k}_0 || \mathsf{uid} || j)$;
   - Call $(\mathsf{pid}^j_{\mathsf{uid}}, \mathsf{pvk}^j_{\mathsf{uid}}) = \mathsf{CreatePPK}(\mathsf{pk}_{\mathrm{TE}}, j, \mathsf{aux})$ to get the j-th pseudonym identity and pseudonym verification key, where $\mathsf{aux} = (\mathsf{k}_j, \mathsf{uid}, \mathsf{U}, \mathsf{uk}^*)$.
4. Eventually, AA can obtain a set of pseudonym identities and pseudonym verification keys $\{\mathsf{pid}^i_{\mathsf{uid}}, \mathsf{pvk}^i_{\mathsf{uid}}\}_{i \in [\mathsf{T}_t / \mathsf{T_{pvk}}]}$.

Comparison with LiS2**.** Only the algorithms **Algorithm 2** and $\mathsf{Verify_{V2X}}$ in LARP are identical to the corresponding components of LiS2. Furthermore, in LiS_2, the public key replenishment is outsourced to a fully trusted key generation center (KGC). However, the drawback of LiS_2 is that the KGC needs to keep some critical states (e.g., randomness r') that can enable it to extract the signing key of a client. That is, the KGC can subtract two signatures (as shown by Step 3 of **Algorithm 2**) to extract the signing key with the knowledge of the randomness and message. Hence, we have to prevent the semi-honest AA from obtaining the signing key in the construction of LARP. Our idea for achieving this is to let AA have a secret share of the UHF Key and compute the UHF on the exponent. So the AA in LARP does not know any randomness in plaintext.

Conditional Tracing and Revocation. Optionally, a V2X scheme may need to either punish or expel malicious vehicles. e.g., which intentionally injected fake data using its signing key. Once such vehicle is identified, the corresponding pseudonym certificates of the malicious vehicle should be revoked.

Specifically, we assume there is a trustworthy auditor (e.g., a government authority) which can get information from both AA and EA. The auditor can send the dishonest message-signature tuple (m, R, s) to AA which can first compute $\mathsf{h} = \mathsf{H}(m || \mathsf{R})$ and check the specific BF whether $\mathsf{x} = \mathsf{CHF}(\mathsf{ppk}^j_{\mathsf{uid}}, \mathsf{h}, \mathsf{s}) = (\mathsf{ppk}^j_{\mathsf{uid}})^{\mathsf{h}} \cdot g^{\mathsf{s}}$ is in it. Then the auditor signs and publishes a message to notify of invalid the $\mathsf{pvk}^j_{\mathsf{uid}}$ and ask AA to stop generating verification key for it. In addition, the auditor sends the uid to EA to notify the misbehavior of uid. EA considers the re-enrolment of the uid's vehicle depending upon the reasons for deactivation and existing regional vehicle registration laws.

6 Security Analysis

In this section, we present the security results of LARP. LARP provides the security properties defined in Section 4. Let T_s be the runtime of $\mathsf{Sign}_{\mathsf{V2X}}$. And we can have a bound of the query number of signing oracle as $l \leq \mathsf{T}_t/\mathsf{T}_s$. Let l_r be the bit-length of Z_q.

We show the security results of LARP via the following theorems. Here we only give the high-level ideas of the proofs. We present the full proofs of the these theorems in Appendix A and B, respectively.

Theorem 1. *Suppose that the chameleon hash function and the Bloom filter are secure, and the hash function* H *is modeled as a random oracle. Then* LARP *is a secure V2X scheme with advantage* $\mathsf{Adv}^{\mathsf{Auth}}_{\mathcal{A},\mathsf{LARP}}(\kappa) \leq \mathsf{N}_v \cdot (\mathsf{Adv}^{\mathsf{CR}}_{\mathcal{A},\mathsf{CH}}(\kappa) + l^2/2^{l_r} + \mathsf{Adv}^{\mathsf{AR}}_{\mathcal{A},\mathsf{BF}}(\kappa, \mathsf{T}_t, N, \epsilon)$*, where* $l \leq \mathsf{T}_t/\mathsf{T}_s$.

The proof of this theorem is given by a sequence of games following [25]. The first game is the real V2X security experiment. We gradually change the game until the adversary's advantage in the final game is zero. The challenger first guesses (in advance) which vehicle will be successfully attacked by the adversary. Since there are N_v vehicles, the probability that the challenger guesses correctly is lower-bounded by $1/\mathsf{N}_v$. Next, the challenger replaces each output of the UHF of the guessed victim vehicle with uniform random values. Due to the security of UHF, the adversary cannot distinguish this modification. Then, we change the game by letting the challenger randomly generate a signature for each message of the victim vehicle instead of running the collision calculation algorithm CHColl. So that we can reduce the security to that of the chameleon hash function. Furthermore, we further reduce the security to the false positive error of BF. Finally, the challenger can reject any signatures which are generated by herself, so the adversary has zero advantage in the last game.

Theorem 2. *If the hash function* H *is modeled as random oracle, the proposed* LARP *is a pseudonymity conscious V2X scheme with advantage* $\mathsf{Adv}^{\mathsf{Pseud}}_{\mathcal{A},\mathsf{LARP}}(\kappa) \leq \frac{\mathsf{N}_v \cdot q_{\mathsf{H}}}{2^{l_r}}$*, where* q_{H} *is the number of oracle queries of* H*, and* l_r *is the bit-length of the output of* H.

In the proof, we mainly show that the adversary only has a negligible advantage to ask a random oracle query $k^*_j = \mathsf{H}(k^*_0||\mathsf{uid}||j)$ with a pseudonym key k^*_0 of a vehicle since the seed k_0 is randomly chosen and not corrupted. So the adversary cannot remove k^*_j from the corresponding pseudonym public key $\mathsf{ppk}^j_{\mathsf{uid}}$ to break the pseudonymity of the vehicle.

7 Performance Evaluation

In this section, we evaluate the computation and communication overhead of LARP. The results show that LARP is practical for V2X. Table 2 shows the cost

of a UHF operation with different parameters (for reference), the signature and verification overhead.

Setup. We implement LARP on a Raspberry Pi 4 with 8 GB RAM. The device simulates a TE onboard a vehicle. The verifier is implemented in a more powerful device, namely a desktop PC with Intel (R) Xeon (R) E3-1230 v2 3.30 GHz, running Window 10 operating system. Moreover, we use the cryptographic library Miracl [1] to realize the algorithms in LARP. The hash function used by LARP is instantiated with SHA-256.

Computation overhead of Algorithm 1. The epoch time $\mathsf{T}_{\mathsf{pvk}}$ is set to 5 minutes as recommended by ETSI standard.

Table 2. Runtimes of evaluation

Scheme	$\|q\| = 128bits$	$\|q\| = 192bits$	$\|q\| = 384bits$
A UHF	0.177 ms	0.227 ms	0.457 ms
LARP Signature	20.629 ms	21.719 ms	23.527 ms
LARP Verification	23.024 ms	24.943 ms	27.41 ms

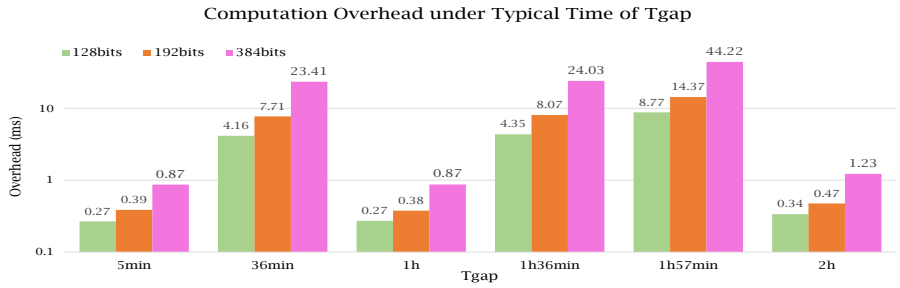

Fig. 4. Computation Overhead under Typical $\mathsf{T}_{\mathsf{gap}}$

We now examine the number of UHF computation under different $\mathsf{T}_{\mathsf{gap}}$ when the vehicle starts. If $\mathsf{T}_{\mathsf{gap}} < \mathsf{T}_{\mathsf{pvk}}$, the TE does not need to execute the UHF operation, and it can update from the start of T_{vk}. If $\mathsf{T}_{\mathsf{pvk}} \leq \mathsf{T}_{\mathsf{gap}} < 1\mathsf{hr}$, it needs to execute $\frac{\mathsf{T}_{\mathsf{gap}}}{\mathsf{T}_{\mathsf{pvk}}}$ UHF operations. If $1\mathsf{hr} \leq \mathsf{T}_{\mathsf{gap}}$, there will be $\frac{\mathsf{T}_{\mathsf{gap}}}{1\mathsf{hr}} + \frac{\mathsf{T}_{\mathsf{gap}} \mod 1hr}{\mathsf{T}_{\mathsf{pvk}}}$ UHF operations. For example, if $\mathsf{T}_{\mathsf{gap}} = 5$ min, or $\mathsf{T}_{\mathsf{gap}} = 1\mathsf{hr}$, only one UHF operation is needed. When $\mathsf{T}_{\mathsf{gap}} = 36$ min, 1hr36 min, or 1hr57 min, there are UHF opreations in the third layer extending the computation overhead.

Fig. 4 shows the computation overhead with varying $\mathsf{T}_{\mathsf{gap}}$, and Table 3 shows the number of operations in three layers for each $\mathsf{T}_{\mathsf{gap}}$. It can be seen that the cost does not increase linearly with $\mathsf{T}_{\mathsf{gap}}$. In particular, the cost with $\mathsf{T}_{\mathsf{gap}} = 1\mathsf{hr}$ or $\mathsf{T}_{\mathsf{gap}} = 2\mathsf{hr}$ is less than that of $\mathsf{T}_{\mathsf{gap}} = 1$hr36 min or $\mathsf{T}_{\mathsf{gap}} = 36$ min. When

$\mathsf{T}_{\mathsf{gap}}$ is not divisible by $\mathsf{T}_{\mathsf{pvk}}$, the computation overhead is much higher because there are many UHF operations at the third layer. We note that in this case, the vehicle can choose $\mathsf{T}_{\mathsf{gap}}$ value as a higher value nearest to the one that is divisible by $\mathsf{T}_{\mathsf{pvk}}$, which significantly reduces the cost. To speed up the update process, the TE can also pre-compute and cache results of some intermediate UHF operations in advance during idle time.

Table 3. The number of operations

$\mathsf{T}_{\mathsf{gap}}$	n1	n2	n3
5 min	0	1	0
36 min	0	7	60
1hr	1	0	0
1hr36 min	1	7	60
1hr57 min	1	1	120
2hr	2	0	0

Signature overhead. In Table 2, we show the cost of signature generation (i.e., Algorithm 2) of LARP under three parameters $|q| = 128$ bits, $|q| = 192$ bits, $|q| = 384$ bits.

Verification overhead. In Table 2, we show the cost of signature verification. Although this is not the performance bottleneck in V2X, it can be seen that LARP supports efficient verification, about 23 ms for $|q| = 128$ bits.

Comparison with IFAL. In Table 4, we summerize the compareison between IFAL and LARP from the perspectives of security properties and performance of a vehicle. We let 'M-OBU' denote the security property on the resilience of malicious OBU. Let 'Add' and 'Mul' dente the addition and scalar multiplication in a group, respectively. Let 'EMul' denote the point multiplication in ECC, and 'H' be a hash operation.

Table 4. Comparison

	Security Properties			Performace	
	Auth	Pseudo	M-OBU	Signature	Verification
IFAL	√	√	×	2H+ 1EMul +1Add+4Mul	1H+2EMul
LARP	√	√	√	1H+2Add+2Mul	1H+2EMul+1BF

According to the benchmark results from [8], the cost of an EMul is roughly 300 times of the cost of a hash operation H. So the signing procedure of IFAL is much less efficient than LARP. For high-speed moving vehicles, it is always better to have a faster signing procedure. Moreover, LARP does not require the OBU to be honest.

In LARP, the main communication overhead consists of the cost of the enrollment of a vehicle, in which TE shares some values with EA and AA. After that, TE only needs to periodically send an activation request to AA. In contrast, in IFAL, a vehicle needs to communicate regularly with EA for activation codes, and with AA to obtain many pseudonym certificates. As a result, LARP has a lower communication overhead.

8 Related Work

In this section, we review the most recent standards and literature relating to security and privacy for vehicular networks.

Zarki *et al.* [38] first considered the security challenges of vehicular networks and proposed to utilize PKI-based digital signatures for authentication of vehicular messages. Shortly afterward, based on an extensive requirements analysis, the European SeVeCom project [17] harmonised many existing technologies to develop a complete security architecture for V2X. The design principles, security mechanisms, and reference architecture components of the SeVeCom project have been adopted by the leading V2X security architecture standards such as the ETSI standard [15] and the USDOT standard [31]. However, the digital signatures recommended in those standards are not lightweight, so they might not be suitable for some constrained devices used in a vehicle (e.g., smart card).

Beyond the standards, Petit *et al.* [22] investigated the broader literature on cryptographic schemes for vehicular communications and identified a lot of alternative technologies. Decentralising the roles of certificate generation and revocation to the vehicles is one of the techniques. Afterward, a revocation protocol without the need for pseudonym resolution (REWIRE) was designed by Förster *et al.* [10]. But Whitefield *et al.* pointed that the REWIRE protocol exits some security flaws and thus proposed an improved obscure token (O-Token) protocol, which can provide a powerful and verifiable guarantee for vehicle revocation without revealing the long-term identity of vehicles. To further enhance privacy against dishonest or colluding certificate authorities, Förster *et al.* [9] also introduced a scheme that can protect vehicles against colluding certificate authorities. Based on pseudonym certificates, Whitefield *et al.* [30] utilized direct anonymous attestation to propose a pseudonym certificate management scheme. What's more, Hicks *et al.* [12] also put forward a new V2X architecture and key management solution based on direct anonymous attestation (VDAA). Although VDAA adopted efficient standards-compliant ECDSA signatures on broadcast messages, the efficiency of signature generation and verification is still not friendly to the resource-constrained vehicles and devices.

In [34], Yant *et al.* proposed a new cryptographic primitive called group time-based one-time passwords (GTOTP), which can satisfy anonymity and traceability. Although GTOTP can be used for message authentication, it may suffer a long latency for verification (as shown in their application in the construction of privacy-preserving proof of location). It is an open question to use GTOTP to realize instant message authentication like a digital signature scheme.

Recently, the most influential pseudonym scheme is IFAL proposed by Verheul *et al.* [29]. Based on the ETSI standard PKI architecture, IFAL provides an improvement to the ETSI ITS security architecture and is ready to be integrated to the ETSI standard. Although IFAL avoids the need for certificate revocation by introducing pre-issued pseudonym certificates that are only usable upon receiving small activation codes (e.g., via SMS), there are still some weaknesses in IFAL. Firstly, IFAL is still not friendly to resource-constrained vehicles or other devices in terms of signature generation and verification. Secondly, the generation of a pseudonymous public key in IFAL relies on the authorisation authority. In addition, if a vehicle OBU is corrupted in IFAL, there is still the risk of leaking the privacy of the vehicle. Therefore, to avoid the above issues, we propose a lightweight auto-refreshing pseudonym (LARP) scheme for V2X in this paper.

9 Conclusion

In this paper, we introduced a lightweight auto-refreshing pseudonym scheme for V2X communications named LARP. LARP has very fast signing algorithm which is particularly suitable for embedded devices of the vehicles. We also showed the practicality of our scheme through implementing it on a Raspberry Pi 4. In addition, LARP realizes non-interactive pseudonym auto-refreshing and reduces risks of vehicle OBU being corrupted.

In the future, the readers are encouraged to further improve our scheme. Moreover, we will explore the introduction of blockchain technology to further reduce the reliance on trusted authorities and achieve more efficient use of road capacity. Since LARP is lightweight, it may have potential applications in Cyber-physical Systems (CPS). However, the implementation of cryptographic algorithms on a CPS device (e.g., programmable logic controller [33]) might be challenging.

We would like to thank the anonymous reviewers for their invaluable comments and suggestions. This work is supported by the National Natural Science Foundation of China (Grant No.61872051), and A*STAR under its RIE2020 Advanced Manufacturing and Engineering (AME) Industry Alignment Fund - Pre Positioning (IAF-PP) Award A19D6a0053. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of A*STAR. Zheng Yang is also supported by the Fundamental Research Funds for the Central Universities (Grant No. SWU-KR22003).

References

1. Miracl (2018), `https://github.com/miracl`
2. Bißmeyer, N., Mauthofer, S., Bayarou, K.M., Kargl, F.: Assessment of node trustworthiness in vanets using data plausibility checks with particle filters. In: 2012 IEEE Vehicular Networking Conference (VNC). vol. 2, pp. 78–85. IEEE (2012)

3. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Communications of the ACM (1970)
4. Carter, J.L., Wegman, M.N.: Universal classes of hash functions. Journal of computer and system sciences **18**(2) (1979)
5. Dierks, T., Rescorla, E.: The transport layer security (tls) protocol version 1.2. Tech. rep., Internet Engineering Task Force (IETF) (2008)
6. Eckhoff, D., Sommer, C.: Marrying safety with privacy: A holistic solution for location privacy in vanets. In: 2016 IEEE Vehicular Networking Conference (VNC). pp. 1–8. IEEE (2016)
7. EFKON: E-tag i-red on-board-unit 530 (2021)
8. Emura, K., Hayashi, T., Ishida, A.: Group signatures with time-bound keys revisited: A new model, an efficient construction, and its implementation. IEEE Trans. Dependable Secur. Comput. **17**(2), 292–305 (2020)
9. Förster, D., Kargl, F., Löhr, H.: Puca: A pseudonym scheme with strong privacy guarantees for vehicular ad-hoc networks. Ad Hoc Networks **37**(5) (2016)
10. Förster, D., Löhr, H., Zibuschka, J., Kargl, F.: Rewire–revocation without resolution: A privacy-friendly revocation mechanism for vehicular ad-hoc networks. In: International Conference on Trust and Trustworthy Computing. pp. 193–208. Springer, Springer (2015)
11. Giannetsos, T., Krontiris, I.: Securing v2x communications for the future: Can pki systems offer the answer? In: Proceedings of the 14th International Conference on Availability, Reliability and Security. pp. 1–8 (2019)
12. Hicks, C., Garcia, F.D.: A vehicular DAA scheme for unlinkable ECDSA pseudonyms in V2X. In: EuroS&P. pp. 460–473. IEEE (2020)
13. Huang, L., Matsuura, K., Yamane, H., Sezaki, K.: Enhancing wireless location privacy using silent period. In: IEEE Wireless Communications and Networking Conference, 2005. vol. 2, pp. 1187–1192. IEEE (2005)
14. Institute, E.T.S.: Etsi ts 102 731. intelligent transport systems (its); security; security services and architecture. Tech. Rep. (September 2010)
15. Institute, E.T.S.: Etsi ts 102 940. intelligent transportation systems (its); security; its communications security architecture and security management. Tech. Rep. (April 2018)
16. Johnson, D., Menezes, A., Vanstone, S.: The elliptic curve digital signature algorithm (ecdsa). International Journal of Information Security **1**(1) (2001)
17. Kargl, F., Papadimitratos, P., Buttyan, L., Müter, M., Schoch, E., Wiedersheim, B., Thong, T.V., Calandriello, G., Held, A., Kung, A.: Secure vehicular communication systems: Implementation, performance, and research challenges. IEEE Communications magazine **46**(11) (2008)
18. Krawczyk, H., Rabin, T.: Chameleon hashing and signatures (1998)
19. Liang, H., Wu, J., Mumtaz, S., Li, J., Lin, X., Wen, M.: Mbid: Micro-blockchain-based geographical dynamic intrusion detection for v2x. IEEE Communications Magazine **57**(10) (2019)
20. Naor, M., Yogev, E.: Bloom filters in adversarial environments. In: CRYPTO (2). Lecture Notes in Computer Science, vol. 9216, pp. 565–584. Springer (2015)
21. Nevelsteen, W., Preneel, B.: Software performance of universal hash functions. In: International Conference on the Theory and Applications of Cryptographic Techniques. Springer (1999)
22. Petit, J., Schaub, F., Feiri, M., Kargl, F.: Pseudonym schemes in vehicular networks: A survey. IEEE communications surveys & tutorials **17**(1) (April 2014)
23. Rescorla, E.: The transport layer security (tls) protocol version 1.3. Tech. rep., Internet Engineering Task Force (IETF) (2018)

24. Schäge, S.: Tight proofs for signature schemes without random oracles. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer (2011)
25. Shoup, V.: Sequences of games: A tool for taming complexity in security proofs. IACR Cryptol. ePrint Arch. **2004** (2004)
26. Siemens, A.: Vehicle-to-x (v2x) communication technology. Mobility. Siemens. Com (2015)
27. Siemens, A.: Nhtsa 2020 report. Tech. Rep. (2019)
28. T-SYSTEM: satellic smart on-board unit (2021)
29. Verheul, E., Hicks, C., Garcia, F.D.: Ifal: Issue first activate later certificates for v2x. In: 2019 IEEE European Symposium on Security and Privacy (EuroS&P). pp. 279–293. IEEE, IEEE (2019)
30. Whitefield, J., Chen, L., Giannetsos, T., Schneider, S., Treharne, H.: Privacy-enhanced capabilities for vanets using direct anonymous attestation. In: IEEE Vehicular Networking Conference (VNC). pp. 123–130. IEEE, IEEE (2017)
31. Whyte, W., Weimerskirch, A., Kumar, V., Hehn, T.: A security credential management system for v2v communications. In: 2013 IEEE Vehicular Networking Conference. pp. 1–8. IEEE, IEEE (2013)
32. Yang, P., Deng, L., Yang, J., Yan, J.: Sasmf: Socially aware security message forwarding mechanism in vanets. Mobile Networks and Applications (2019)
33. Yang, Z., Bao, Z., Jin, C., Liu, Z., Zhou, J.: Plcrypto: A symmetric cryptographic library for programmable logic controllers. IACR Trans. Symmetric Cryptol. **2021**(3), 170–217 (2021)
34. Yang, Z., Jin, C., Ning, J., Li, Z., Dinh, A., Zhou, J.: Group time-based one-time passwords and its application to efficient privacy-preserving proof of location. In: ACSAC '21: Annual Computer Security Applications Conference, Virtual Event, USA, December 6 - 10, 2021. pp. 497–512. ACM (2021)
35. Yang, Z., Jin, C., Tian, Y., Lai, J., Zhou, J.: Lis: Lightweight signature schemes for continuous message authentication in cyber-physical systems. In: Proceedings of the 2020 ACM Asia Conference on Computer and Communications Security (2020)
36. Yao, A.C.C., Zhao, Y.: Online/offline signatures for low-power devices. IEEE Transactions on Information Forensics and Security **8**(2) (2013)
37. Yavuz, A.A., Ozmen, M.O.: Ultra lightweight multiple-time digital signature for the internet of things devices. IEEE Transactions on Services Computing (2019)
38. Zarki, M.E., Mehrotra, S., Tsudik, G., Venkatasubramanian, N.: Security issues in a future vehicular network. In: European Wireless. vol. 2 (2002)

A Proof of Theorem 1

We present the proofs in a sequence of games following [25]. Let E_i be an event that an adversary $\mathcal{A}$ wins in $\mathsf{Game}\ i$, i.e., $\mathcal{A}$ succeeds in forging a signature. The games are adapted from [35].

Game 0. This is the original security experiment. In Game_0, all oracle queries are answered honestly according to the specification of LARP. Thus, we have

$$\Pr[\mathsf{E}_0] = \mathsf{Adv}^{\mathsf{Auth}}_{\mathcal{A},\mathsf{LARP}}(\kappa).$$

Game 1. This game proceeds with exactly as the previous game, but the simulator would abort if fails to guess the *victim vehicle* that the adversary succeeds in forging its signature. Since there are N_v vehicles at all, the probability of a correct guess is at least $1/\mathsf{N}_v$. Thus, we have

$$\Pr[\mathsf{E}_0] = \mathsf{N}_v \cdot \Pr[\mathsf{E}_1].$$

Game 2. In this game, the simulator $\mathcal{C}$ proceeds exactly like the previous game, but each output of universal hash function UHF running by the TE of the guessed victim vehicle is replaced with a uniform randomness to generate the verification key instead of using the universal hash function UHF. Since the hash keys of UHF (of all three layers) are chosen uniformly at random, the output of UHF is distributed uniformly as well by the security of UHF. Hence, our change does not modify the distribution of the dummy random values. Therefore, we have that

$$\Pr[\mathsf{E}_1] = \Pr[\mathsf{E}_2].$$

Game 3. This game proceeds like before, but the challenger changes the game for the guessed victim vehicle uid^* as follows. The simulator first generate l dummy random messages and random values $\{(h_i, r'_i)\}_{i\in[l]}$. To map a message m^* chosen by the adversary to the pre-sampled h_i, we attach each message m_i^* with a random value R_i (by design). So the hash input string $m_i^*||R_i$ is unique unless a collision happens with a collision probability $l^2/2^{l_r}$. As a result, we can establish a unique connection between each string $m_i^*||R_i$ and h_i as $\mathsf{H}(m_i^*||R_i) = h_i$. Moreover, the verification key is then generated using the real signature and message pairs, i.e., $\{(\mathsf{psk}^j_{\mathsf{uid}^*}, h_i)\}_{i\in[l]}$, where $\mathsf{psk}^j_{\mathsf{uid}^*} = k_j \cdot \mathsf{sk}_{\mathsf{uid}^*}$ and $k_j = \mathsf{H}(k_0||uid||j)$. Note that $s_i := M \cdot sk_{\mathsf{uid}^*} + r'_i - h_i \cdot k_j \cdot sk_{\mathsf{uid}^*} \pmod q$. We can rewrite s_i as $s_i := r'_i + \tilde{h}_i \pmod q$ where $\tilde{h}_i = M \cdot sk_{\mathsf{uid}^*} - h_i \cdot k_j \cdot sk_{\mathsf{uid}^*}$. Since each h_i is unique, so is $\tilde{h}_i$. We claim that each signature value s_i is statistically close to a uniform random value with distance 0 due to the security results regarding combing function in [24, Lemma 1]. Thus, we have that

$$\Pr[\mathsf{E}_2] \leq \Pr[\mathsf{E}_3] + l^2/2^{l_r}.$$

The changes in this game enable us to reduce the security of LARP to that of the chameleon hash function in the next game.

Game 4. In this game, the simulator $\mathcal{C}$ proceeds exactly like the previous game but adds an abort rule. Namely, $\mathcal{C}$ aborts if the adversary submits a tuple $(m^*, \mathsf{sig}^*, \mathsf{pvk}^j_{\mathsf{uid}^*})$ which leads to a collision to one of those hash values recorded in BF of the guess victim vehicle, i.e., $\mathsf{CHF}(\mathsf{pvk}^j_{\mathsf{uid}^*}, \mathsf{H}(m^*||R^*), s*) = \mathsf{CHF}(\mathsf{pk}^*_{\mathrm{TE}}, M, r'_i)$ for some r'_i and $\mathsf{sig}^* = (s^*, R^*)$. If this event occurs with a non-negligible probability, then there must exist an efficient algorithm $\mathcal{F}$ which can break the security of the chameleon hash function by using $\mathcal{A}$.

Specifically, $\mathcal{F}$ could simulate the game for $\mathcal{A}$ while receiving a challenge public key pk^*_{CH} from the chameleon hash simulator. $\mathcal{F}$ can generate each pseudonym verification key as $\mathsf{ppk}^j_{\mathsf{uid}^*} := (pk^*_{\mathsf{CH}})^{k_j}$ using the knowledge of k_j. Then $\mathcal{F}$ uses

the corresponding $\mathsf{pvk}^j_{\mathsf{uid}^*}$ and the real signatures randomly chosen as in the previous game to generate the verification key rather than using dummy message/randomness pairs.

Due to the security of the chameleon hash function, we have that

$$\Pr[\mathsf{E}_3] \leq \Pr[\mathsf{E}_4] + \mathsf{Adv}^{\mathsf{CR}}_{\mathcal{A},\mathsf{CH}}(\kappa).$$

Game 5. In this game, $\mathcal{C}$ proceeds as before, but aborts if the adversary $\mathcal{A}$ submits a tuple $(m^*, \mathsf{sig}^*, \mathsf{pvk}^j_{\mathsf{uid}^*})$ such that $\mathsf{Check}(\mathsf{CHF}(\mathsf{pvk}^j_{\mathsf{uid}^*}, \mathsf{H}(m^*||R^*), s^*)) = 1$, and (m^*, sig^*) has not been queried by the adversary before in any signing oracle, i.e., $\mathcal{A}$ finds a false positive error of BF. By applying the false positive probability of BF, we have that

$$\Pr[\mathsf{E}_4] \leq \Pr[\mathsf{E}_5] + \mathsf{Adv}^{\mathsf{AR}}_{\mathcal{A},\mathsf{BF}}(\kappa, \mathsf{T}_t, N, \epsilon).$$

Put together with all probabilities in the above games, and we obtain the result of this theorem. □

B Proof of Theorem 2

The proof proceeds in the following sequence of games. Let E_i be an event that an adversary $\mathcal{A}$ wins in $\mathsf{Game}\ i$.

Game 0. This game is the original security experiment following original algorithms in LARP. In this game, all queries are answered honestly according to the specification of LARP. Thus, we have

$$\Pr[\mathsf{E}_0] = \mathsf{Adv}^{\mathsf{Pseud}}_{\mathcal{A},\mathsf{LARP}}(\kappa).$$

Game 1. This game proceeds exactly as the previous game, but the simulator aborts if the adversary ask a random oracle query with a pseudonym key k_0^* of a vehicle, i.e., $k_j^* = \mathsf{H}(k_0^*||\mathsf{uid}||j)$. Note that if the adversary knows the k_0^*, then it must be able to compute an inversion to the pseudonym verification key as pvk^{1/k_0^*} to get the public key of a vehicle. Since the H is a random oracle, the distribution of the output of H is uniformly distributed in Z_q. This implies that all pseudonym keys are uniform random as well. Since the challenged vehicles and the AA are not corrupted, the only way for the adversary to get a pseudonym key is to make a random guess. The probability of a correct guess on a pseudonym key is at most $1/2^{l_r}$. As there are N_v vehicles that the adversary can attack. The abort probability of simulator in this game is $\Pr[\text{abort}] = \frac{\mathsf{N}_v \cdot q_{\mathsf{H}}}{2^{l_r}}$. Thus we have

$$\Pr[\mathsf{E}_0] = \Pr[\mathsf{E}_1] + \Pr[\text{abort}] \leq \Pr[\mathsf{E}_1] + \frac{\mathsf{N}_v \cdot q_{\mathsf{H}}}{2^{l_r}}.$$

As the adversary cannot get the pseudonym keys in this game, an adversary can only win the game by randomly guessing the bit b. So we have the winning probability of this game $\Pr[\mathsf{E}_1] = 0$, which is the result of this theorem.