DoS-resilient Cooperative Beacon Verification for Vehicular Communication Systems

Hongyu Jin*, Panos Papadimitratos

Networked Systems Security Group, KTH Royal Institute of Technology, Stockholm, Sweden

Abstract

Authenticated safety beacons in Vehicular Communication (VC) systems ensure awareness among neighboring vehicles. However, the verification of beacon signatures introduces significant processing overhead for resource-constrained vehicular On-Board Units (OBUs). Even worse in dense neighborhood or when a clogging Denial of Service (DoS) attack is mounted. The OBU would fail to verify for all received (authentic or fictitious) beacons. This could significantly delay the verifications of authentic beacons or even affect the awareness of neighboring vehicle status. In this paper, we propose an efficient cooperative beacon verification scheme leveraging efficient symmetric key based authentication on top of pseudonymous authentication (based on traditional public key cryptography), providing efficient discovery of authentic beacons among a pool of received authentic and fictitious beacons, and can significantly decrease waiting times of beacons in queue before their validations. We show with simulation results that our scheme can guarantee low waiting times for received beacons even in high neighbor density situations and under DoS attacks, under which a traditional scheme would not be workable.

Keywords: Security, Privacy, Pseudonymous authentication, Efficiency

1. Introduction

Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication in Vehicular Communication (VC) systems entail high-rate transmissions: typically, for safety beacons, vehicles/On-Board Units (OBUs) transmit at a rate of 10 Hz, i.e., 10 beacons per second. In spite of approaches that adapt the beacon rate, the challenge is clear: as VC systems get progressively widely deployed, each vehicle will have to process safety beacons (along with other traffic) from several tens of vehicles within its OBU range, e.g., 300 messages per second for 30 neighboring vehicles. The provision of security and privacy protection aggravates the situation, adding communication overhead (digital signatures and short-term certificates attached, thus longer messages), as well as computation overhead (signature verifications mostly, and signature calculations).

A number of improvements (e.g., [1, 2, 3]) are compatible with the standardized pseudonymous authentication approach. For example, the Pseudonymous Certificates (PCs), termed *pseudonyms*, can be attached to beacons periodically or based on the sender's context [3], instead of attaching on each and every beacon. At the receiver side, a PC needs to be validated only once and cached [2]. For newly received beacons attached with cached PCs, only the signatures on beacons need to be verified. These approaches provide significant improvements and show how one can dimension processing power [1, 2], but they are conservative: they assume each node verifies signatures on all received beacons. Indeed, this is the straightforward approach. An alternative, adaptive, reactive approach has been considered in [4], but only for multi-hop messages.

In [5], vehicles share beacon verification results, so that the vehicles can benefit from verification efforts of neighboring vehicles. However, the work only considered an environment in which only benign vehicles broadcast authentic beacons. In fact, dynamic vehicular mobility and thus topology (connectivity and physical neighborhood) creates a simple yet very effective attack vector, a clogging DoS attack: an attacker, even external, could generate large volumes of fictitious beacons, purportedly from not previously encountered vehicles/OBUs. This would essentially prevent timely reception and validation of legitimate beacons. Even more so if there are multiple such offending adversarial transmitters across an area of the VC system - in which case, one could even classify this as a distributed attack, a Distributed Denial of Service (DDoS) attack. Discovering valid pseudonyms is challenging, because each node receives a pool of valid and non-valid pseudonyms, while the non-valid pseudonyms could be the majority. This attack is cheap, because the attacker only needs to generate random bytes as signatures and attaches them to the broadcasted masqueraded beacons, while all its neighbors would be affected. Although the shared verification results can expedite the validation process [5], the beacon queue can be saturated by an extremely high fictitious beacon rate, inevitably resulting in significantly high waiting time for authentic beacons (the majority of computation resources

*Corresponding author

Email addresses: hongyuj@kth.se (Hongyu Jin), papadim@kth.se (Panos Papadimitratos)

has to be used to verify fictitious signatures). Moreover, a successful PC verification does not guarantee the timely verifications of following beacons from the same sender, because adversaries can simply attach valid PCs to the fictitious beacons.

This is exactly the problem we address in this paper. We extend the design of cooperative beacon verification scheme proposed in [5] by taking into consideration the defense against DoS attacks and leveraging efficient symmetric key based authentication. Here, we use Timed Efficient Stream Loss-Tolerant Authentication (TESLA) [6] on top of (a traditional public key cryptography based) pseudonymous authentication. Vehicles cooperatively discover beacons signed under new authentic PCs and cache the PCs. The beacons attached with cached PCs can be validated based on (1) signature verifications, (2) shared verification results, or (3) TESLA Message Authentication Codes (MACs). Our simulation results show our scheme can significantly decrease waiting time for authentic beacons in high neighbor density situations and under DDoS attacks, where a traditional scheme would not be workable.

In the rest of the paper, we provide background information on VC security and privacy and related works (Sec. 2), we then explain the system and adversary models, and security requirements (Sec. 3). We present in detail our scheme (Sec. 4), and provide a security analysis (Sec. 5), followed by evaluation based on simulation results (Sec. 6), before concluding remarks (Sec. 7).

2. Background and related works

A basic functionality for VC systems is safety beacons, used to inform vehicle status to surrounding/neighboring vehicles. This can in turn improve traffic efficiency and safety by virtue of the awareness of the transportation environment. Authentication and integrity of safety beacons are strictly required. Traditional public key cryptography could provide those, yet the use of a long-term key pair and certificate would undermine user privacy: safety beacons could be trivially used to continuously track the sender vehicles. Pseudonymous authentication [2, 7, 8, 9] provides both security and privacy for safety beacons. A PC is a short-term certificate issued by the Vehicular Public-Key Infrastructure (VPKI) without any information on the long-term identity of the vehicle, thus making messages signed under different PCs unlinkable. However, this anonymity/pseudonymity is conditional: when misbehavior is detected, the relevant PCs (corresponding to the misbehaving OBU/vehicle) can be revealed through a resolution protocol and then revoked [2, 7, 8, 9].

Signature verifications remain expensive for resource-constrained OBUs. Thus, optimizations for decreasing communication and computation overhead [1, 2] have been proposed, but they do not change the fact that the signatures on each and every received beacons should be verified. Cooperative beacon verification [5, 10] can help validating beacons based on shared verification results, but it is not resilient to DoS attacks, because a large portion of computation resources would still be used to verify fictitious beacons. PCs can be validated based on a pre-downloaded bloom filter [11], but PCs for newly joining vehicles (commonly so in VC systems) after the generation (and downloading) of the bloom filter would not be included, thus signatures on those PCs need to be verified. This could still leave a gap for adversarial nodes to inject fictitious PCs to the network.

TESLA-based authentication for VC systems [12, 13] provides a cheaper way for validating successive beacons by a given sender, after a successful signature verification. However, it is still critical to validate new PCs and the corresponding beacons in high node density environment or under DoS attacks in a timely manner. Prediction of the next beacon content (e.g., vehicle location) can help validating beacons based on the prediction result included in a previous beacon [14, 15]. However, such an approach cannot tolerate packet losses: once a beacon is lost, the following beacon can only be verified based on its signature. Integrating with TESLA-based authentication can address packet losses in the prediction-based approaches [15]. However, in a network with high packet loss ratio (in dense neighborhood or under DoS attacks), the majority of beacons need to be validated based on TESLA MACs, while the prediction based authentication would be barely used.

Recall that vehicles need to change their PCs periodically and frequently. An adversary can target this feature and flood with fictitious beacons, which can significantly delay verifications (and affect the caching) of new PCs. This problem is not addressed by any of the works discussed above. Our scheme addresses this problem by providing a cooperative approach for efficiently discovering authentic PCs when a large amount of new (authentic and fictitious) PCs are received, and can provide timely beacon validation with shared verification results and TESLA.

Puzzle-based approaches [16, 17] have been proposed to defend against DoS attacks for mutual authentication in V2I and V2V communication, but they are not suitable for authenticating frequent connectionless safety beacons. Efficient pseudonym validation schemes based on alternative non-classical cryptographic primitives (e.g., ID-based cryptography) [18, 19, 20] rely on connection to Roadside Unit (RSU) and need to communicate with RSUs for updating pseudonyms of newly joined or revoked nodes, and could fail to validate pseudonyms when connection to RSUs is lost. An efficient Certificate Revocation List (CRL) release approach [21] is proposed to defend against DoS attacks on CRL checking, however, this is orthogonal to our scheme and the two can co-exist.

3. System model and security requirements

3.1. System and adversary model

In our system, each vehicle is equipped with a set of PCs obtained from a VPKI [9]. We assume a Sybil-resilient

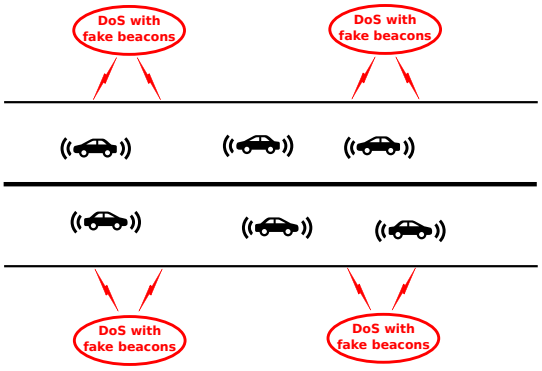

Figure 1: Adversaries launching DDoS attacks.

pseudonym lifetime policy as the one proposed in [9]: PCs have non-overlapping lifetimes and all pseudonym lifetimes are aligned. We mandate each and every beacon of a vehicle is properly signed with the private key corresponding to the currently valid PC.

We consider an overall landscape of adversarial models, and security and privacy requirements [22] for our scheme design, while, in this paper, we are primarily concerned with adversaries that seek to exhaust computation resources of OBUs. As shown in Fig. 1, adversaries can flood the VC systems with fake beacons at high rate (e.g., on the road side with powerful devices) to prevent vehicles from validating other vehicles' beacons in a timely manner. Due to the broadcast nature of V2V communication, even a single adversary could create devastating effect as fictitious beacons would affect all vehicles within communication range. Even more so, an adversary could mount DoS attacks at multiple points with several devices, resulting in a DDoS attack. This can affect the awareness of neighboring vehicles and a number of VC applications. The fictitious beacons will be proven invalid when the signatures are verified. However, in a traditional scheme (in the absence of DoS-resilient and efficient verification mechanism), this could significantly delay verifications of authentic beacons, because long waiting times are needed when the queues are saturated with fictitious beacons. More specifically, vehicles cannot distinguish authentic beacons among the received (authentic and fictitious) beacons and verify them first in order to gain awareness of surrounding vehicles.

3.2. Security and privacy requirements

Our scheme is designed to address the above mentioned adversary model taking into consideration the following security and privacy requirements:

Authentication and integrity - Node messages should allow their receivers to corroborate the legitimacy of their senders and verify they were not modified or replayed. We do not require strict identification of the sender, but require the validation that the sender is a legitimate participant of the VC systems.

Non-repudiation and accountability - Any node can be tied to its actions, and, if need arises, be held accountable and possibly have its long-term identity revealed and have itself evicted from the system.

Anonymity/Pseudonymity and unlinkability - A vehicle's beacons should be only linkable over a protocol selectable period, τ. Anonymity/Pseudonymity should be conditional, allowing the system to identify a misbehaving node and evict it.

DoS-resilience - Nodes should be resilient to adversarial nodes and their flooded fictitious beacons. Even under DoS attacks, increase in beacon validation delays should be moderate and nodes should be able to gain awareness of neighboring vehicles in a timely manner.

Table 1: Notation

N	*Neighbor size*
PC	*Pseudonymous certificate*
$\{msg\}_{\sigma_{PC}}$	*Signed message with PC attached*
α	*No. of verification results in a beacon*
γ	*Beacon frequency*
τ	*PC lifetime*
$H()/H$	*Hash function/Hash value*
$MAC_K(msg)$	$H(K\|\|msg)$
L_H	*Hash/MAC size*
L_{TESLA}	*Length of TESLA key chain*
t_{now}	*Fresh timestamp (current time)*
t_{next}	*Next own beacon point after* t_{now}
Pr_{loss}	*Probability of packet loss*

4. Our scheme

4.1. Overview

Our scheme extends the traditional V2V message verification, leveraging cooperating vehicles (referred as *nodes* in the rest of the paper) to defend against DoS attacks and reduce validation delays. The basic idea is to augment each (safety) beacon with brief identifiers of previously validated beacons, and attach with TESLA key and MAC. The identifiers indicate the corresponding beacons have been verified by the sender. This is exactly where nodes can benefit from each other: accepting a beacon can help verifying the (received and queued) beacons the identifiers in this beacon point to. The TESLA keys and MACs can expedite message validation, thus remaining resilient to extreme network situations that vehicles receive more beacons than they could handle/verify. Moreover, under a DoS attack, these identifiers point nodes to the potentially valid beacons attached with non-cached PCs. These beacons are assigned higher priority in the queue so that nodes can obtain faster the sought awareness of surrounding benign nodes. Table 1 summarizes notation used in this paper.

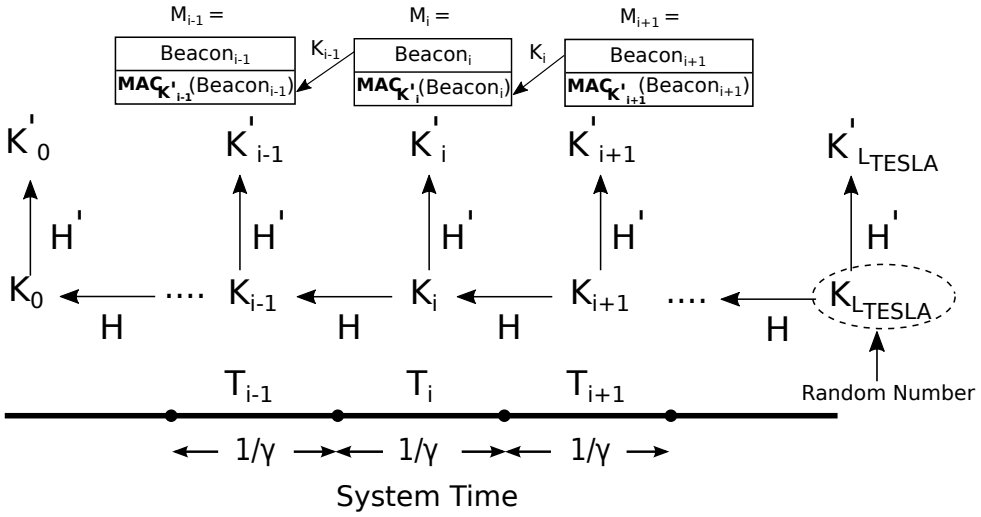

Figure 2: TESLA key chain generation and usage.

4.2. TESLA key chain generation

Each node broadcasts safety beacons at a rate γ. In this paper, we assume $\gamma = 10\ Hz$, a typical value in the literature [23, 15, 14]. Thus, a PC can be used for authenticating at most $L_{TESLA} = \tau \cdot \gamma$ beacons. For each pseudonym, a node generates a TESLA key chain [6] with a length of L_{TESLA}, as shown in Fig. 2. $H()$ and $H'()$ are two different hash functions. K_i is the key that will be disclosed with the beacon broadcasted during time slot, T_{i+1}, and $K'_i = H'(K_i)$ is the key used to calculate TESLA MAC for the beacon during T_i. An independent TESLA key chain is generated for each PC, in order to ensure unlinkability among the beacons broadcasted (and authenticated) under different PCs. For example, consider $\tau = 5\ min$, then a node generates a hash chain for each pseudonym, with $L_{TESLA} = 3,000$. In our scheme, each beacon is signed by the sender with the private key corresponding to the currently valid PC and also attached with a TESLA MAC calculated based on the corresponding TESLA key. Each TESLA key is used to authenticate a beacon broadcasted during its corresponding time slot. We assume clocks of nodes are loosely synchronized and system time is split in a same manner for each node. Each time slot (T_i) is 0.1 *sec* and we mandate each node should broadcast only one beacon within a time slot. The exact time points each node broadcasts beacon in a time slot could vary, but we assume beacons from a single node is periodical (i.e., broadcasted every 0.1 *sec*). The corresponding TESLA key, K_i (and K'_i), should be only used to authenticate a beacon broadcasted within the corresponding time slot T_i. For example, K'_i should be used to generate MAC for $Beacon_i$ and K'_{i+2} should be used to generate MAC for $Beacon_{i+2}$. In case a beacon was skipped or not successfully broadcasted during, e.g., T_{i+1}, then K_{i+1} (and K'_{i+1}) should be skipped. This helps receivers to authenticate TESLA MACs with correct TESLA keys based on the beacon reception times.

A node needs at most 864,000 TESLA keys per day (i.e., 24 h) with $\gamma = 10\ Hz$. Thus, the maximum storage needed is $864,000 * L_H$. For example, with $L_H = 80\ bit$ (such a hash size is sufficient for safety beacons, due to their ephemeral nature), the storage size is 8.64 MB. A node could store a TESLA key every, e.g., 10 slots and calculate the TESLA keys in between on-the-fly when they are needed. For example, K_i and K_{i+10} can be stored, and K_{i+1} ... K_{i+9} can be easily calculated based on K_{i+10}; this significantly decreases the storage overhead for TESLA keys. For an off-the-shelf vehicular OBU, 864,000 hash computations can be completed within a few seconds and a storage overhead of 8.64 MB (864 KB if keys at every 10 slots are stored) per day is acceptable.

4.3. Beacon queue maintenance

The format of a signed beacon in our scheme is changed into:

$$Beacon_i = \{Status, t_i, K_{i-1}, H_1, ..., H_\alpha\}_{\sigma_{PC}} \quad (1)$$

and the format of a beacon message (i.e., a signed beacon attached with a TESLA MAC) is:

$$M_i = \{Beacon_i, MAC_{K'_i}(Beacon_i)\} \quad (2)$$

Status consists of vehicle status information, including location, velocity, direction, etc. t_i is the timestamp of the beacon message, which should be within the time slot T_i. K_{i-1} is the TESLA key for the previous beacon (or time slot). $H_1, ..., H_\alpha$ are the hashes of latest (based on the times of reception) verified beacons: the beacons that the signatures were verified, not cooperatively verified or verified based on TESLA keys and MACs.

Algorithm 1 shows the beacon reception process. Consider a received beacon message, M_i. Attached PC of

Algorithm 1 Beacon reception

Received a beacon message M_i
$M_i = \{Beacon_i = \{Status, t_i, K_{i-1}, H_1...H_\alpha\}_{\sigma_{PC}}, MAC_i\}$
if $PC \in CACHED\_PC$ **then**
  if No beacon with K_{i-1} was received, **and** K_{i-1} is the correct key corresponds to time slot of t_i **then**
    Insert $\{M_i, H(M_i)\}$ to the head of $Queue_1$.
    Find beacon message, M', attached with PC from $Queue_1$.
    if M' is found **then**
      Remove M' from $Queue_1$.
      Input M' to Algorithm 4 for TESLA MAC validation.
    end if
  else
    Drop M_i.
  end if
else
  Insert $\{M_i, H(M_i)\}$ to the head of $Queue_1$.
end if

Algorithm 2 Queue element selection

if $Queue_2$ is not empty **then**
  Choose first element, E, from $Queue_2$.
  if Multiple elements exist in $Queue_2$ with same PC of E **then**
    Choose the element, E, with the latest timestamp.
  end if
else if $Queue_1$ is not empty **then**
  k element(s) exist(s) in $Queue_1$ with beacon timestamps satisfy $t_{beacon} + 1/\gamma > t_{next}$.
  if $k > 0$ **then**
    Uniform randomly selects an element, E, from k first elements of $Queue_1$.
  else if $k = 0$ **then**
    Choose first element, E, from $Queue_1$.
  end if
end if
Input E to Algorithm 3.

M_i is checked first. If it was *verified and cached* (simplified as *cached* in the rest of the paper), then the receiver checks whether the attached TESLA key is correct. This is done by validating it against the TESLA key in a previous beacon from the same sender[1]. For example, if a beacon, M_{i-2}, received during the time slot, T_{i-2}, was verified based on signature and includes a TESLA key, K_{i-3}. Then, the correct TESLA key disclosed in T_i should be $H^2(K_{i-3})$, i.e., $H(H(K_{i-3}))$. If multiple beacons with the correct TESLA key, K_{i-1}, were received, only the first of them is kept, due to the nature of single hop communication (see Sec. 5 for further analysis).

If M_i satisfies the above requirement or the attached PC is non-cached, then the receiver generates a queue element and insert it to the head of $Queue_1$:

$$\{M_i, H(M_i)\} \tag{3}$$

[1]We refer a pseudonymous identity as a sender. Thus, messages attached with the PC are considered sent from the same sender, while beacons sent under two different PCs (from the same node) can be considered sent from two different senders in this context.

Each node maintains two queues. $Queue_1$ stores newly received beacon messages and $Queue_2$ stores potentially valid beacon messages with non-cached PCs attached to them. Elements in $Queue_2$ are extracted from $Queue_1$ during the execution of Algorithms 3 and 4 (explained below). $Queue_2$ is given higher priority in order to ensure timely awareness of newly encountered nodes when computation resource is scarce; the beacons signed under the cached PCs can be validated through the "cheaper" MACs.

The selection of the beacons to verify from the queues is done according to Algorithm 2. When $Queue_2$ is not empty, the node pops an element from the head. Consider the sender of this element (i.e., beacon) is V. If multiple elements sent from V exist in $Queue_2$, the element with the most recent timestamp is chosen: validating this beacon can further validate earlier beacons based on MACs.

If $Queue_2$ is empty, an element from $Queue_1$ is selected. Unlike element selection from $Queue_2$, the element here is randomly chosen among those that satisfy the condition: $t_{beacon} + 1/\gamma > t_{next}$, where t_{beacon} is the timestamp of each queued beacon and t_{next} is the time point for the next own beacon broadcast. Recall that in Algorithm 1, the new elements are inserted at the head

Algorithm 3 Cooperative verification

$E = \{M, H(M)\}$
$M = \{Beacon = \{Status, t_i, K_{i-1}, H_1...H_\alpha\}_\sigma, MAC_i\}$
if The signature of $Beacon$ is valid **then**
  Accept $Beacon$.
  if PC of $Beacon$ is new **then**
    Add PC to $CACHED\_PC$.
    Extract beacon(s) signed under the same PC from $Queue_2$ and $Queue_1$ except the latest one;
    store the extracted beacon(s) to $BEACON\_SET$ and remove $BEACON\_SET$ from $Queue_1$.
    for Each M_i in $BEACON\_SET$ **do**
      Input M_i to Algorithm 4 for TESLA MAC validation.
    end for
  end if
  for Each H_i in $H_1..H_\alpha$ of $Beacon$ **do**
    if An element, E', with the message hash, H_i, is found in $Queue_1$ **then**
      $E' = \{M' = \{Beacon', MAC'\}, H(M')\}$
      if PC' of $Beacon'$ is in $CACHED\_PC$ **then**
        Accept $Beacon'$.
      else
        Remove E' from $Queue_1$ and insert to the end of $Queue_2$.
      end if
    end if
  end for
end if

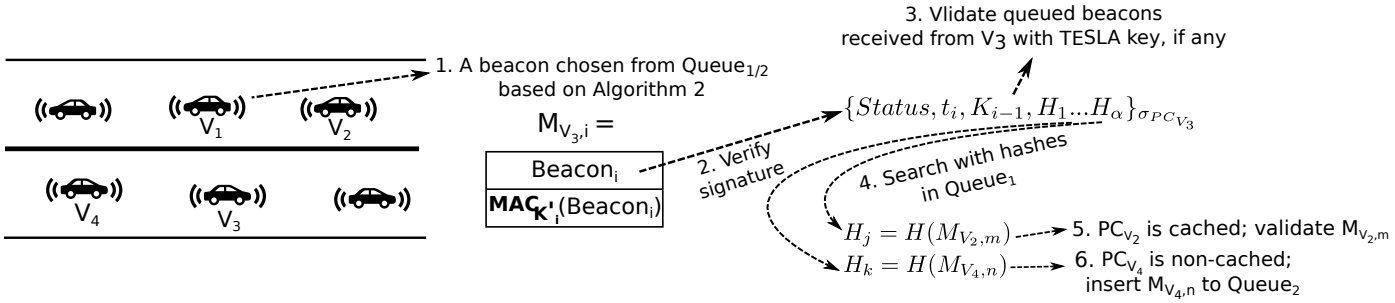

Figure 3: An example of cooperative verification steps (Algorithm 3).

of $Queue_1$ and, here, an element is chosen again from the head of $Queue_1$. The following three reasons led to such a Last Come First Served (LCFS) and conditional randomized selection process:

- Verifications of fresh beacons benefit other nodes when the results are shared through own beacons. For example, in a traditional First-Come First-Served (FCFS) scheme, consider a node (V_1) that chooses a beacon, $M_{V_2,i-1}$, from $Queue_1$ for signature verification, while the latest beacon from V_2, $M_{V_2,i}$, was lost. Then, sharing this result is not helpful for the receiver, V_3, which received both $M_{V_2,i-1}$ and $M_{V_2,i}$, because $M_{V_2,i-1}$ was already validated based on the TESLA key disclosed with $M_{V_2,i}$.

- Randomized selection among fresh beacons guarantees that the nodes would not choose the same latest received beacon to verify, thus decreases duplicated signature verifications performed by the cooperating nodes.

- The time condition ($t_{beacon} + 1/\gamma > t_{next}$) guarantees that, before the dissemination of next own beacon, the beacons, corresponding to piggybacked verification results, have not been validated based on TESLA MACs by receivers, otherwise, the shared verification results could be less useful.

If no beacon message satisfies the time condition, then the first queue element is popped from the head of $Queue_1$.

4.4. Cooperative beacon verification

Algorithm 3 shows our cooperative verification scheme, taking E chosen from Algorithm 2 as the input. The signature(s) of (the attached non-cached PC and) M is(are) verified first. If PC is non-cached, then it is cached and the beacons from the same sender can be validated based on TESLA MACs (except the most recent one, for which the corresponding TESLA key has not been disclosed/received). The hashes $H_1, ..., H_\alpha$ in M are used to validate further the beacons in $Queue_1$. The node searches, with each H_i, in $Queue_1$ for a matching $H(M')$. If a matching queue

Algorithm 4 TESLA MAC Validation

```
1: M_i = {Beacon_i = {Status, t_i, K_{i-1}, H_1...H_α}_σ, MAC_i}
2: if MAC_{K'_i}(Beacon_i) = MAC_i then
3:     Accept Beacon_i.
4:     for Each H_i in H_1..H_α of Beacon do
5:         if An element, E, with the message hash, H_i, is found in Queue_1 then
6:             if PC of Beacon is not in CACHED_PC then
7:                 Remove E from Queue_1 and insert to the tail of Queue_2.
8:             end if
9:         end if
10:    end for
11: else
12:    Drop M.
13: end if
```

element is found, the node checks whether the attached PC' is cached. If yes, then M' is accepted. Otherwise, the element is inserted to the tail of $Queue_2$. For the latter case, M' is not simply accepted because this validation is correlated with the validations of extra beacon messages carrying the same PC' based on TESLA MACs, thus the security risk is high (see Sec. 5 for further analysis). Fig. 3 shows an example of the cooperative verification scheme: V_1 chooses a beacon from its queue for signature verification. After the signature verification is passed, K_{i-1} is used to validate queued beacons sent from V_3. A piggybacked hash value, H_j is found to match a beacon attached with PC_{V_2}. As a result, $M_{V_2,m}$ is accepted, because PC_{V_2} is already cached. Moreover, another hash value, H_k, is found to match a beacon, $M_{V_4,n}$, while PC_{V_4} is non-cached. Therefore, $M_{V_4,n}$ is inserted to $Queue_2$ for a signature verification later.

4.5. TESLA-based validation

Each TESLA-based validation (in Algorithms 1 and 3) is processed according to Algorithm 4. If the TESLA MAC is validated, then each H_i in the validated beacon is used to search matching beacon message from $Queue_1$. If there is any matching beacon attached with non-cached PC, then it is inserted to the tail of $Queue_2$. The hashes are only used to search for potentially valid beacon messages attached with non-cached PCs, but not used for validating beacons, because TESLA-based authentication does not provide non-repudiation. Therefore, the TESLA-validated beacons should not be used to further validate more beacons (see Sec. 5 for further analysis).

5. Privacy and security Analysis

In this section, we provide privacy and security analysis for our scheme. We emphasize that we are not concerned here with the validity of the message content, e.g., the correctness of a location or an alert about emergency braking; those are orthogonal and can be addressed by relevant consistency checking [24, 25] and data-centric security [26] schemes. Here, we are concerned with incorrectly signed (with arbitrary content) messages, and the attempt to saturating benign vehicles with (fake) signature verifications while affecting validations of authentic safety beacons.

5.1. Privacy analysis

We note that privacy is not weakened by our scheme. The shared verification results in a beacon do not link transmissions of any other node, beyond what one can infer from the geographical information included in the beacons themselves. An independent TESLA key chain is generated for each PC, thus messages authenticated under different PCs cannot be linked. Beacons correlated (i.e., linkable) based on the TESLA keys from a same TESLA key chain are also signed under the same PC, so that they can be already trivially linked based on the PC (at most for a period, τ) even without TESLA keys, as is the case for a traditional scheme.

5.2. Security analysis

Beacons in our scheme are validated based on one of three components: (1) signature, (2) MAC, and (3) shared verification result. We first analyze security properties for each validation approach. Then, we discuss non-repudiation provision, notably how to compensate for the lack of non-repudiation in TESLA. Finally, we conclude with the analysis of resilience to DoS attacks of our scheme.

Signature verification: The use of pseudonymous authentication, as per the standards under development [23, 9], guarantees non-repudiation and message integrity and authentication; as long as the receiving node performs the cryptographic validation itself (message signature and attached pseudonym validation). Inclusion of fresh timestamps in messages and timestamp checking prevent replay of messages.

TESLA-based validation: TESLA provides message integrity and authentication after hash chain anchors are verified with signatures [6]. Time synchronization (e.g., through GPS) guarantees that only the corresponding TESLA keys are used at any point in time (see Fig. 2). Our scheme prevents memory exhaustion attacks, because at most one beacon carrying the correct TESLA key needs

to be queued. This is due to the nature of single-hop communication. The first beacon with the correct TESLA key would be either (1) the (potentially) authentic one, or, (2) if the authentic one was lost, the first among the masqueraded beacons attached with the correct TESLA key (overheard by adversary after it was disclosed by its legitimate sender). However, the latter one will be proven invalid either through its signature verification or TESLA MAC validation. This is not more harmful than receiving a randomly created fake beacon, which is even easier to generate from the perspective of an adversary. We mandate that each beacon is signed by the sender. If the content of a beacon is suspicious, the receiver can choose to verify the signature on the beacon, thus non-repudiation can be achieved.

Cooperative verification: Shared verification results in beacon messages are used to either find potentially valid beacon messages attached with non-cached PCs or validate latest beacons attached with cached PCs. For a cached PC, there is always at most one beacon message in $Queue_1$ from the same sender. A shared verification result matching this beacon could be used to validate this beacon. In this case, non-repudiation, authentication and integrity are not directly achieved based on signature or TESLA MAC on the validated beacon, but the trust on this beacon is established leveraging another node. If the validated beacon is proven fake later, the node that shared this verification result is accountable for this misbehavior. A newer beacon from the same sender (as the validated beacon) can help further corroborate the correctness of cooperatively validated beacon (based on TESLA MAC). Moreover, with mobility prediction [4] and content validation [24, 25] approaches, a vehicle can choose to accept the cooperatively validated beacon only when the beacon content is within the prediction/validation error threshold. Again, when the content of any beacon is suspicious, signature can be verified to achieve immediate non-repudiation, authentication and integrity.

On the other hand, for a non-cached PC, there could exist several beacon messages from the same sender in $Queue_1$. A falsely accepted fake PC could result in accepting a series of false beacons. Therefore, the signature of a beacon message attached with non-cached PC should be verified (by inserting the beacon message to $Queue_2$) before the correlated beacons can be validated based on TESLA MACs.

Non-repudiation: We mandate each beacon be signed by the sender, so that signature verification can ensure non-repudiation. However, TESLA-based validation and cooperative verification cannot guarantee that the validated/accepted beacons are properly signed. Although accountability for cooperatively verified beacons can be traced to senders of verification results; TESLA-based validation does not provide non-repudiation, as validations are merely based on symmetric TESLA keys.

Consider an adversary fabricates beacons with disclosed TESLA keys and claim the beacons were received at correct time slots (with correct TESLA keys and fake signatures). The original generator of those TESLA keys cannot deny the fabricated beacons were not sent by him-/her-self. On the other hand, with this in mind, a legitimate sender can also send beacons with correct TESLA keys and fake signatures at correct time slots. A receiver would accept the beacons with TESLA-based validations, and realize the signatures were invalid later. However, as TESLA-based validations do not provide non-repudiation, the sender cannot be held accountable, exactly because the beacons presented by the victim might have been fabricated (by the victim the same way the adversary did).

With such a vulnerability, TESLA-based validations should be carefully used. This is the case in our scheme: TESLA-based validations are only used for validating successive beacons attached with cached PCs. Redundancy of frequent beacons and predictability of vehicle status within a short period [27] can minimize the vulnerability incurred by the lack of non-repudiation: a beacon that deviates too much from a previous signature-validated beacon can be considered suspicious and the signature of this beacon can be verified as a backup approach. Moreover, the piggybacked hash values in TESLA-validated beacons are only used for discovering potentially valid beacons from new senders, which are verified based on signatures later.

Thwarting clogging (D)DoS: With TESLA-based validation, once a TESLA key for a PC is stored after a successful signature verification, then the rest of the beacons form the same sender can be validated based on TESLA MACs. However, TESLA-based validation could be helpful only after a successful signature verification for a beacon message attached with a non-cached PC. Discovery of authentic beacons attached with non-cached PCs still remains an issue. This is problematic especially when vehicles reach a time point for pseudonym change. Flooded fictitious beacons can affect the verifications of new PCs, thus the awareness of surrounding vehicles.

With our cooperative verification scheme, vehicles can efficiently fetch potentially valid beacons attached with non-cached PCs based on shared verification results and verify them first in order to gain awareness of newly encountered vehicles with short delays. Once a beacon from a sender vehicle has been successfully verified, the successive beacon messages from the same sender can be validated based on TESLA MACs or shared verification results.

6. Performance evaluation

In this section, we evaluate our scheme with simulations. We show that our scheme achieves low delays in high vehicle density scenario, and even under DDoS attacks, which would not have been possible for the standard approach (signature verification of all received beacon messages, referred as *baseline* scheme in the rest of the paper).

Table 2: System Parameters (***Bold*** for Default Setting)

N	20, 30, 40, 50, ***60***, 70, 80
γ	***10*** Hz
T_{vrfc}	***4*** ms
Pr_{loss}	0.1, ***0.2***, 0.3, 0.4, 0.5, 0.6, 0.8
α	1, 2, 3, ***4***, 5
N_{adv}	***4*** (Static), ***10*** (Mobile)
γ_{adv}	***250*** Hz (No packet loss)

6.1. Simulation settings

We use OMNeT++ [28] to simulate our scheme and analyze the system performance. We consider two mobility scenarios: a static scenario and a mobile scenario. For the mobile scenario, Veins [29] is used for the connection between SUMO [30] and OMNeT++. Table 2 shows the system parameters and values used in the simulation. We assume a communication range of 200 m with a packet loss ratio of Pr_{loss}, i.e., a node can successfully receive a broadcasted beacon within the communication range with a probability, $1 - Pr_{loss}$. We assume a signature verification delay of T_{vrfc} for both PC verification and message verification. For simplicity, we assume the beacon validations based on hash computations (i.e., cooperative validations or TESLA-based validations) incur zero delay (in reality, they introduce a tiny delay, which can be in the order of μs). For each simulation setting, we perform 5 randomly seeded experiments of 1 min and the results are averaged over these 5 runs. The bold values in Table 2 are the default ones used in our simulation. For example, when N is the parameter we examine (Fig. 5b), then the rest of the parameters have the default values: $Pr_{loss} = 0.2$, $T_{vrfc} = 4$ ms, $\alpha = 4$ and $\gamma = 10$ Hz. $N_{adv} = 4$ and $\gamma_{adv} = 250$ Hz are only set for adversarial network scenarios. The default values of T_{vrfc} and γ are typical values based on the literature (e.g., [2, 23]). Note that we assume all beacons from adversaries are received by benign nodes (within the communication range without any packet loss) in order to simulate an extreme situation. For example, when $Pr_{loss} = 0.8$, the equivalent real beacon rate from each adversarial node would be 1250 Hz. This could be non-realistic for a single adversary, but we can consider one adversarial node in our evaluation as an aggregate of multiple adversarial nodes in reality.

6.2. Static mobility scenario

Fig. 4 shows the simulated area for the static scenario: we place the node we evaluate in the center of a 200 m disc area, and place N nodes randomly within the disc. Moreover, another $3 * N$ nodes (thus, a same node density as in the inner disc) are placed at the outer disc (i.e., gray area). We consider this setup to avoid evaluating the scheme in a setting that is overly optimistic and favorable for our scheme. As we assume an effective communication range of 200 m, the node we evaluate receives beacons from

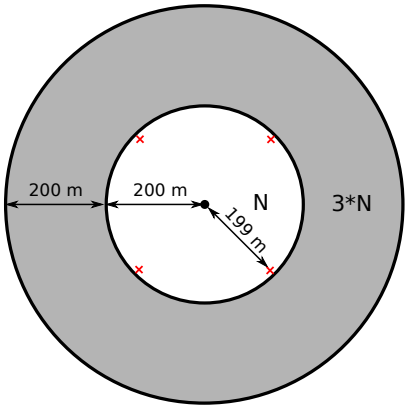

Figure 4: Static scenario: Simulated area with the evaluated node at the center with a neighbor density of N. X marks indicate placement of adversarial nodes.

senders within the inner disc. However, the node in the inner disc will also receive messages from the outer disc (gray area). We add this 200 m extra area to emulate a more realistic scenario, in which the evaluated node could also receive verification results that it is not interested in. Otherwise, if simulated only with the inner disc, the scenario would be very optimistic: all the verification results piggybacked by the received beacons will be useful (i.e., corresponds to beacons within the communication range).

Benign network: We start from the evaluation under benign network conditions, without adversarial nodes launching a DDoS attack. All nodes start beaconing at same time (at the beginning of each simulation run). This essentially simulates the situation that vehicles reach a time point for PC change, which can be considered as the most challenging situation in a benign network.

Fig. 5 shows average waiting time and Fig. 6 shows ratio of validated beacons based on different validation types. *Waiting time* of a beacon is defined as the waited time in queue (from the reception time point) before its validation/verification. Fig. 5a shows the average waiting time for the baseline scheme as a function of N. For $N = 20$ and $N = 30$, we use baseline scheme without TESLA or cooperative verification (i.e., $\alpha = 0$). When $N \geq 40$, the queue would not be sustainable: the average message arrival rate would be $N * \gamma * (1 - Pr_{loss}) \geq 320$ Hz, while only at most 250 (i.e., $\frac{1\ s}{T_{vrfc}}$) signature verifications can be performed per second. Therefore, we use TESLA [12, 13] without cooperative verification for $N \geq 40$. We see when $N \geq 40$, the average waiting time is around 0.1 s, much higher than that for $N \leq 30$. This is due to a significant ratio of beacon messages need to be validated based on TESLA MACs, as shown in Fig. 6a. Fig. 5b shows the average waiting time with our cooperative verification scheme under default settings (see Table 2). The average waiting time is significantly decreased thanks to shared verification results. For example, for the default settings ($N = 60$), the average waiting time is decreased from

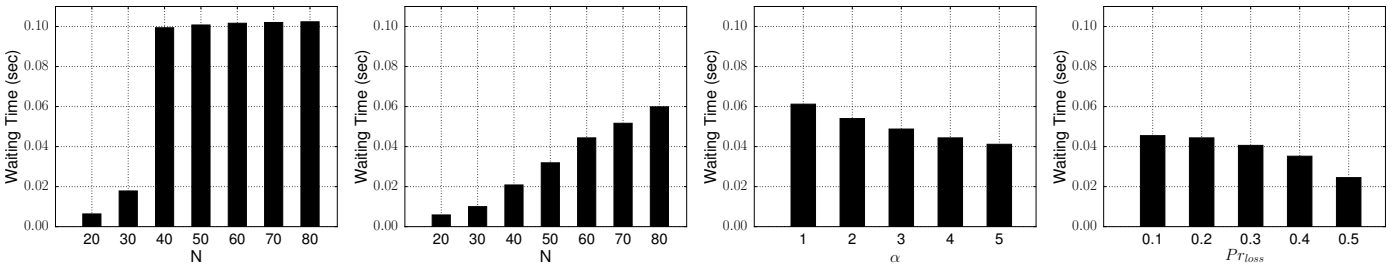

Figure 5: Benign and Static: Average waiting time as a function of N with (a) baseline scheme ($\alpha = 0$ and TESLA for $N \geq 40$) and (b) cooperative beacon verification, and as a function of (c) α and (d) Pr_{loss} under default settings.

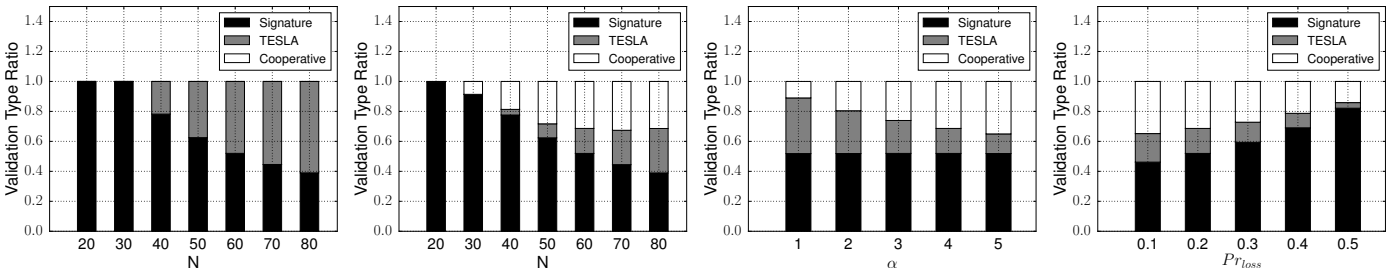

Figure 6: Benign and Static: Ratio of validated beacons based on different validation types as a function of N with (a) baseline scheme ($\alpha = 0$ and TESLA for $N \geq 40$) and (b) cooperative beacon verification, and as a function of (c) α and (d) Pr_{loss} under default settings.

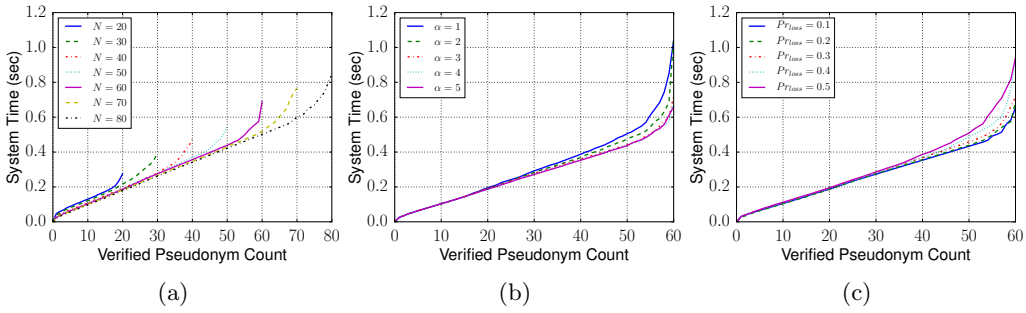

Figure 7: Benign and Static: Passed system time as a function of total verified pseudonyms (out of N) with different (a) N, (b) α and (c) Pr_{loss}.

around 0.1 s (baseline scheme with TESLA) to 0.045 s (our scheme). From Fig. 6b, we see that, when N is high, a significant ratio of beacons are validated cooperatively or based on TESLA MACs while a smaller ratio of beacons is validated based on signatures.

Fig. 5c shows the average waiting time as a function of α. The average waiting time decreases with larger α because more beacons can be validated based on shared verification results, as shown in Fig. 6c. However, the improvement becomes moderate when α is higher (e.g., from $\alpha = 4$ to $\alpha = 5$). Therefore, a reasonable α can be chosen based on different network scenarios to introduce minimum communication overhead (for the attached verification results). The average waiting time also decreases with higher Pr_{loss} (Fig. 5d) because less beacons can be received. As a result, a higher ratio of beacons can be validated based on signature verifications (Fig. 6d).

Fig. 7 shows the progression of non-cached PC verification under different settings. For example, with $N = 20$ in Fig. 7a, all PCs can be verified after around 0.25 s from the beginning of simulation. As described earlier, once a beacon message with non-cached PC is verified, then the rest of beacon messages attached with the same PC can be validated based on TESLA MACs or shared verification results. From Fig. 7, we see all the PCs are verified within 1 s under different settings, which indicates the node can gain awareness of all their neighbors within 1 s. Once a beacon message from a newly encountered vehicle is verified, the vehicle can continuously keep track of the corresponding vehicle with cheap TESLA-based validations and cooperative beacon verifications. This is beneficial especially for a high neighbor density scenario or under a DDoS attack (see below).

Adversarial network: We further consider the ad-

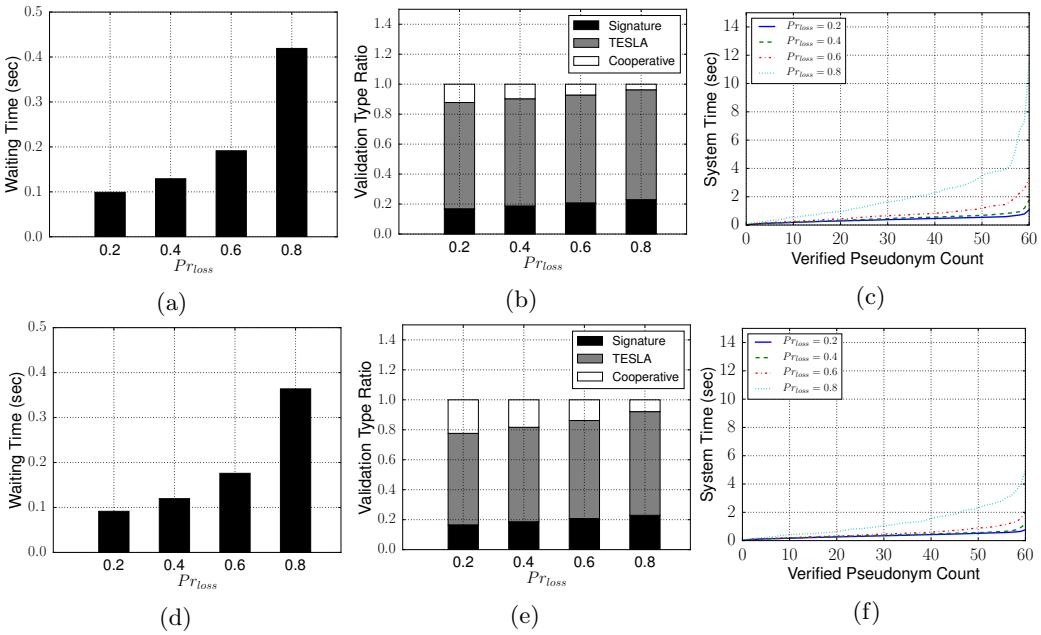

Figure 8: Adversarial and Static: Average waiting time as a function of Pr_{loss} with (a) $\alpha = 2$ and (d) $\alpha = 4$. Ratio of beacon validation types as a function of Pr_{loss} with (b) $\alpha = 2$ and (e) $\alpha = 4$.

versarial network scenarios, where benign nodes are under DDoS attacks. The four adversarial nodes start flooding with fictitious beacons (attached with false signatures and PCs) from the beginning of the simulation and the benign nodes start beaconing after 10 c. The simulation finishes after 1 min (thus, 70 s in total). Here, we also consider relatively high Pr_{loss} (e.g., 0.6 and 0.8), because it is normal to have more packet losses when the network is saturated by the fictitious beacons at a high rate.

Figs. 8a and 8d show the average waiting time as a function of Pr_{loss} when $\alpha = 2$ and $\alpha = 4$ respectively. We see with higher Pr_{loss}, the average waiting time increases, because the majority of beacons have to be validated based on TESLA MACs, as shown in Figs. 8b and 8e. A significant portion of CPU cycles are occupied for verifying fictitious beacons (in an attempt to find new authentic/valid PCs). For example, when $Pr_{loss} = 0.8$, the average waiting time is more than 0.4 s. However, this is still a significant improvement from the baseline scheme, with which the queue would not be even sustainable and waiting time would continuously increase as the time progresses. For example, from simulation results (not shown in the figures), we find that when $\alpha = 0$, only a few pseudonyms can be verified within 70 s, because the queue is filled with fictitious beacons. This cannot be solved even by setting a lifetime, e.g., 1 s, for each received beacon (a beacon is dropped if it stayed in the queue for more than 1 s), because the majority of authentic beacons are also dropped due to expiration. We see with higher α, the average waiting time slightly decreases (from Fig. 8a to Fig. 8d) thanks to more beacons can be validated based on shared verification results (Figs. 8b and 8e).

From Figs. 8c and 8f, we see that all PCs can be verified within 3 s when $Pr_{loss} \leq 0.6$. When $Pr_{loss} = 0.8$, more time is needed to gain awareness of all neighboring nodes: more than 10 s when $\alpha = 2$ and around 5 s when $\alpha = 4$. However, as mentioned earlier, this is still a significant improvement considering only a few PCs can be verified within 60 s after beaconing begins without cooperative verification.

6.3. Highway scenario

Fig. 9 shows the simulated area for the highway scenario. We consider a six-lane 1.5 km highway section, with vehicles entering from both ends. We set the vehicle arrival rate to make the average neighbor density roughly 60 ($N = 58.57$ on average exactly, measured from the simulation results). For this scenario, the first 1 min of each simulation run is used as a warm-up phase to make nodes spatially distributed across the simulated area. Nodes start beaconing from 60 s and the results between 60 s – 120 s is used. Waiting times are collected from nodes when passing the gray area (Fig. 9).

Benign network: In the highway scenario, nodes could continuously encounter new nodes during their trips. It is important that a node gains awareness of any new node entering its communication range on a timely manner. We first evaluate the average pseudonym validation delay, which reflects the speed of discovering new PCs (thus their owners). *Pseudonym validation delay* is defined as the passed time from a new node (PC) is encountered (in terms of received beacons) until at least one beacon from that node is verified. As described in Sec. 4, after a beacon is verified with signature, then the successive beacons from that node can be validated at least based on (relatively cheaper) TESLA MACs.

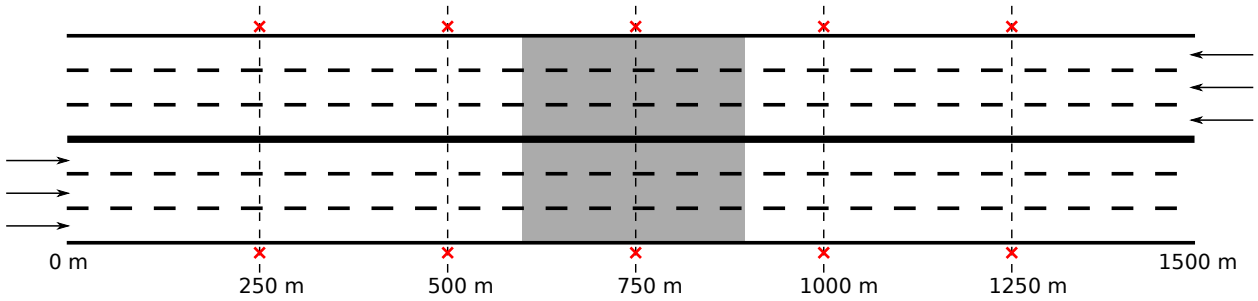

Figure 9: Highway scenario: A six-lane highway scenario with a neighbor density $N \approx 60$; results collected from nodes passing the gray area (central 300 m area). X marks indicate placement of adversarial nodes.

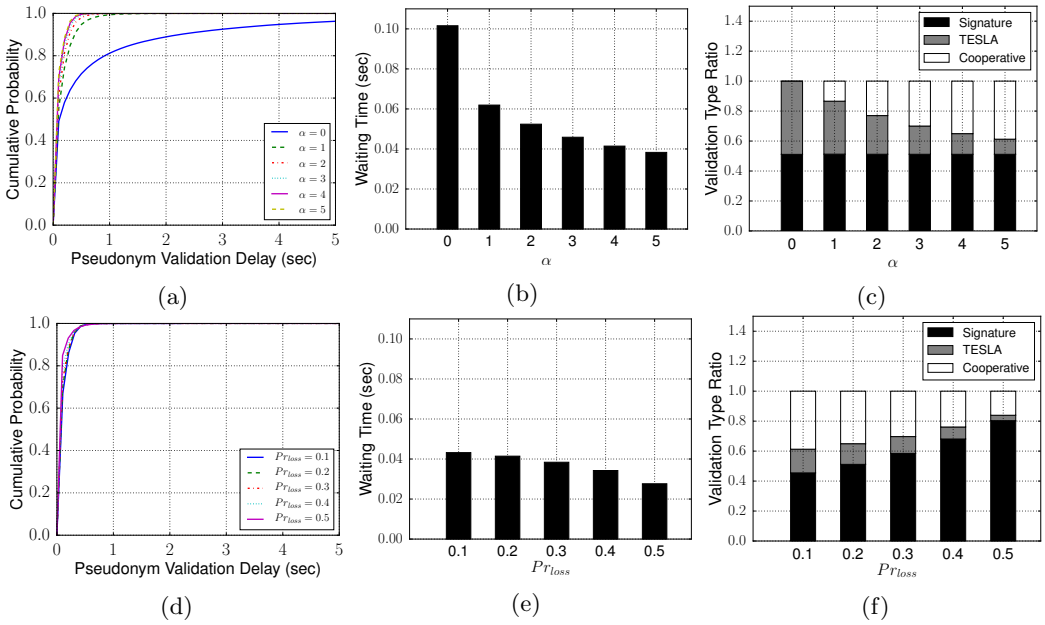

Figure 10: Benign and Highway: CDF of pseudonym validation delay as a function of (a) α and (d) Pr_{loss}. Average waiting time as a function of (b) α and (e) Pr_{loss}. Ratio of validated beacons based on different validation types as a function of (c) α and (f) Pr_{loss}.

We record pseudonym validation delays for each pair of sender/receiver satisfying the condition: the distance between two nodes had been less than 150 m. This essentially eliminates node pairs that only had been at the border of communication range of each other. From Figs. 10a and 10d, we see pseudonym validation delays are almost always less than 1 s with our scheme. This indicates that any pair of nodes that were closer than 150 m can discover each other within 1 s. However, with only TESLA-based validation ($\alpha = 0$), pseudonym validation delay significantly increases: only around 80 % of nodes can be discovered within 1 s.

Figs. 10b and 10e show average waiting time as a function of α and Pr_{loss} respectively, and Figs. 10c and 10f show ratios of beacon validation types as a function of α and Pr_{loss} respectively. We see the average waiting times and ratios of validation types are roughly same as those in the static scenario with $N = 60$ (see Fig. 5).

Adversarial network: We continue with evaluation under the adversarial network scenarios. Here, we first evaluate pseudonym validation ratio. Pseudonym validation ratio is defined as the ratio of validated PCs (i.e., discovered nodes) over total received legitimate PCs (i.e., encountered nodes) at the end of nodes' trips, reflecting the degree of awareness of neighboring nodes. Fig. 11 shows pseudonym validation ratio with $\alpha = 0, 2, 4$ respectively. Without the cooperative verification scheme (Fig. 11a when $\alpha = 0$), only slightly more than 50 % of PCs can be validated, because the majority of computation resource was dedicated for verifying fake signatures. Moreover, even those 50 % of PCs are validated with high delays (see Fig. 12a): only around 80 % (among the above-mentioned 50 %) of PCs can be validated within 5 s.

Figs. 11b and 11c show that, with our scheme, the nodes can effectively discover (in fact, almost all) valid PCs even under the DDoS attack. We see the pseudonym validation ratios are almost 100 %, but slightly less than 100 % with high Pr_{loss} values. This is because pseudonym validation delays are higher with higher Pr_{loss} values (see Figs. 12b and 12c): newly received PCs have not been validated at the end of node trips, while we can expect that these PCs could have been validated if the node trips

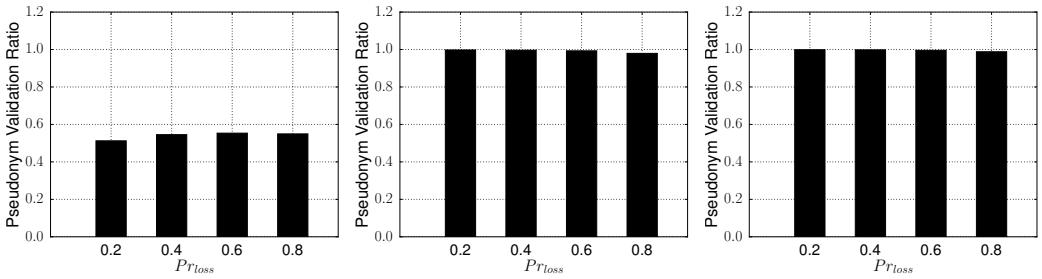

Figure 11: Adversarial and Highway: Pseudonym validation ratio as a function of Pr_{loss} with (a) $\alpha = 0$, (b) $\alpha = 2$ and (c) $\alpha = 4$.

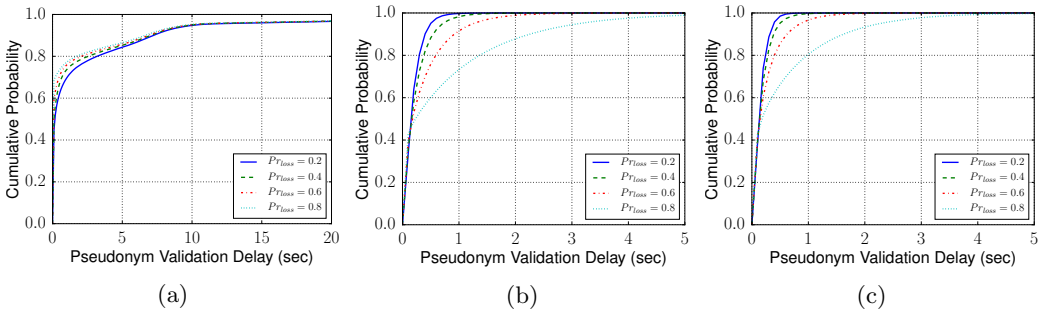

Figure 12: Adversarial and Highway: CDF of pseudonym validation delay as a function of Pr_{loss} with (a) $\alpha = 0$, (b) $\alpha = 2$ and (c) $\alpha = 4$.

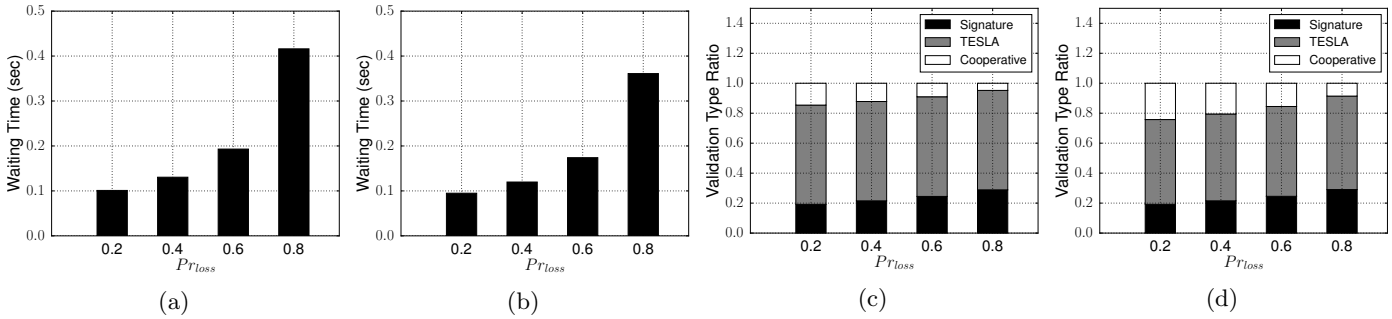

Figure 13: Adversarial and Highway: Average waiting time as a function of Pr_{loss} with (a) $\alpha = 2$ and (b) $\alpha = 4$. Ratio of validated beacons based on different validation types as a function of Pr_{loss} with (c) $\alpha = 2$ and (d) $\alpha = 4$.

continued further.

Fig. 13 shows average waiting times and ratios of validation types for the highway scenario under the DDoS attack. We see our scheme still provides reasonable waiting times (Figs. 13a and 13b) while more beacons have to be validated based on TESLA MACs and piggybacked verification results (Figs. 13c and 13d).

7. Discussions and conclusions

In this paper, we consider a beacon rate of 10 Hz. However, adaptive beacon rate schemes [31, 27, 32] adapt beaconing rates (generally lower than 10 Hz) based on the context (e.g., speed, vehicle density) for increased reliability and lower computation and communication overhead. Our scheme can be readily combined with adaptive beacon rate schemes. TESLA key chain generation and usage can be kept unchanged, but when beacon interval is larger than 0.1 s, the TESLA keys in between can be simply skipped. Skipped time slots are equivalent to lost beacons, thus transparent to the receiver (i.e., no change is needed for the beacon reception and validation processes), but the sender is taking the initiative.

We mandate each beacon to be signed by its sender so that the receiver can always choose to verify the signature if necessary, notably to achieve non-repudiation. Future work will address this explicitly, i.e., the conditions a receiver needs to verify signatures in order to minimize security risks incurred by internal adversaries (e.g., sharing false verification results or attaching fake signatures with correct TESLA keys and MACs).

We demonstrated how our cooperative beacon verification scheme could enable secure VC at network densities even double compared to those prior approaches could be workable for. Even under DoS attacks, vehicles could still maintain a low waiting time for each received beacon so

that the vehicles can gain awareness of neighboring vehicles within short time period. In addition, our scheme is orthogonal to all prior optimizations and could complement them.

References

[1] G. Calandriello, P. Papadimitratos, J.-P. Hubaux, A. Lioy, Efficient and robust pseudonymous authentication in vanet, in: ACM VANET, New York, USA, 2007.

[2] G. Calandriello, P. Papadimitratos, J.-P. Hubaux, A. Lioy, On the performance of secure vehicular communication systems, IEEE TDSC 8 (6) (2011) 898–912.

[3] M. Feiri, J. Petit, F. Kargl, Formal model of certificate omission schemes in vanet, in: IEEE VNC, Paderborn, Germany, 2014.

[4] N. Ristanovic, P. Papadimitratos, G. Theodorakopoulos, J.-P. Hubaux, J.-Y. L. Boudec, Adaptive message authentication for multi-hop networks, in: IEEE/IFIP WONS, Bardonecchia, Italy, 2011.

[5] H. Jin, P. Papadimitratos, Scaling VANET security through cooperative message verification, in: IEEE VNC, Kyoto, Japan, 2015.

[6] A. Perrig, R. Canetti, J. D. Tygar, D. Song, Efficient authentication and signing of multicast streams over lossy channels, in: IEEE Symposium on Security and Privacy, Berkeley, CA, 2000.

[7] S. Gisdakis, M. Laganà, T. Giannetsos, P. Papadimitratos, Serosa: Service oriented security architecture for vehicular communications, Boston, MA, USA, 2013.

[8] M. Khodaei, H. Jin, P. Papadimitratos, Towards deploying a scalable & robust vehicular identity and credential management infrastructure, in: IEEE VNC, Paderborn, Germany, 2014.

[9] M. Khodaei, H. Jin, P. Papadimitratos, Secmace: Scalable and robust identity and credential management infrastructure in vehicular communication systems, IEEE Transaction on Intelligent Transportation Systems 19 (5) (2018) 1430–1444.

[10] X. Lin, X. Li, Achieving efficient cooperative message authentication in vehicular ad hoc networks, IEEE Transactions on Vehicular Technology 62 (7) (2013) 3339–3348.

[11] H. Jin, P. Papadimitratos, Proactive certificate validation for VANETs, in: IEEE VNC, Columbus, OH, 2016.

[12] Y.-c. Hu, K. P. Laberteaux, Strong vanet security on a budget, in: ESCAR, Berlin, Germany, 2006.

[13] A. Studer, F. Bai, B. Bellur, A. Perrig, Flexible, extensible, and efficient vanet authentication, Journal of Communications and Networks 11 (6) (2009) 574–588.

[14] H.-C. Hsiao, A. Studer, C. Chen, A. Perrig, F. Bai, B. Bellur, A. Iyer, Flooding-resilient broadcast authentication for vanets, in: ACM MobiCom, 2011, pp. 193–204.

[15] C. Lyu, D. Gu, Y. Zeng, P. Mohapatra, PBA: Prediction-Based Authentication for Vehicle-to-Vehicle Communications, IEEE TDSC 13 (1) (2016) 71–83.

[16] C. Sun, J. Liu, X. Xu, J. Ma, A privacy-preserving mutual authentication resisting dos attacks in vanets, IEEE Access 5 (2017) 24012–24022.

[17] P. Liu, B. Liu, Y. Sun, B. Zhao, I. You, Mitigating dos attacks against pseudonymous authentication through puzzle-based co-authentication in 5g-vanet, IEEE Access 6 (2018) 20795–20806.

[18] A. Sulaiman, S. K. Raja, S. H. Park, Improving scalability in vehicular communication using one-way hash chain method, Ad Hoc Networks 11 (8) (2013) 2526–2540.

[19] A. Wasef, X. Shen, Emap: Expedite message authentication protocol for vehicular ad hoc networks, IEEE transactions on Mobile Computing 12 (1) (2013) 78–89.

[20] S. Jiang, X. Zhu, L. Wang, An efficient anonymous batch authentication scheme based on hmac for vanets, IEEE Transactions on Intelligent Transportation Systems 17 (8) (2016) 2193–2204.

[21] M. Khodaei, P. Papadimitratos, Efficient, scalable, and resilient vehicle-centric certificate revocation list distribution in vanets, in: ACM WiSec, 2018.

[22] P. Papadimitratos, V. Gligor, J.-P. Hubaux, Securing vehicular communications-assumptions, requirements, and principles, in: ESCAR, Berlin, Germany, 2006.

[23] ETSI EN 302 637-2, Intelligent transport systems; vehicular communications; basic set of applications; part 2: Specification of cooperative awareness basic service (Nov. 2014).

[24] A. Festag, P. Papadimitratos, T. Tielert, Design and performance of secure geocast for vehicular communication, IEEE Transactions on Vehicular Technology 59 (5) (2010) 2456–2471.

[25] T. Leinmüller, C. Maihöfer, E. Schoch, F. Kargl, Improved security in geographic ad hoc routing through autonomous position verification, in: International Workshop on VANET, Los Angeles, CA, 2006.

[26] M. Raya, P. Papadimitratos, V. D. Gligor, J.-P. Hubaux, On data-centric trust establishment in ephemeral ad hoc networks, in: IEEE INFOCOM, Phoenix, AZ, 2008.

[27] H. H. Nguyen, H. Y. Jeong, Mobility-adaptive beacon broadcast for vehicular cooperative safety-critical applications, IEEE Transactions on Intelligent Transportation Systems 19 (6) (2018) 1996–2010.

[28] OMNeT++, https://omnetpp.org/.

[29] C. Sommer, R. German, F. Dressler, Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis, IEEE Transactions on Mobile Computing 10 (1) (2011) 3–15.

[30] D. Krajzewicz, J. Erdmann, M. Behrisch, L. Bieker, Recent development and applications of SUMO - Simulation of Urban MObility, International Journal On Advances in Systems and Measurements 5 (3&4) (2012) 128–138.

[31] C. Sommer, O. K. Tonguz, F. Dressler, Traffic information systems: efficient message dissemination via adaptive beaconing, IEEE Communications Magazine 49 (5) (2011) 173–179.

[32] R. K. Schmidt, T. Leinmuller, E. Schoch, F. Kargl, G. Schafer, Exploration of adaptive beaconing for efficient intervehicle safety communication, IEEE Network 24 (1) (2010) 14–19.