

Accountable, Scalable and DoS-resilient Secure Vehicular Communication

Hongyu Jin* and Panos Papadimitratos

Networked Systems Security Group
KTH Royal Institute of Technology
Stockholm, Sweden
<https://www.eecs.kth.se/nss>

Abstract

Standardized Vehicular Communication (VC), mainly Cooperative Awareness Messages (CAMs) and Decentralized Environmental Notification Messages (DENMs), is paramount to vehicle safety, carrying vehicle status information and reports of traffic/road-related events respectively. Broadcasted CAMs and DENMs are pseudonymously authenticated for security and privacy protection, with each node needing to have all incoming messages validated within an expiration deadline. This creates an asymmetry that can be easily exploited by external adversaries to launch a clogging Denial of Service (DoS) attack: each forged VC message forces all neighboring nodes to cryptographically validate it; at increasing rates, easy to generate forged messages gradually exhaust processing resources and severely degrade or deny timely validation of benign CAMs/DENMs. The result can be catastrophic when awareness of neighbor vehicle positions or critical reports are missed. We address this problem making the standardized VC pseudonymous authentication [1, 2, 3] *DoS-resilient*. We propose efficient cryptographic constructs, which we term message verification *facilitators*, to prioritize processing resources for verification of potentially valid messages among bogus messages and verify multiple messages based on one signature verification. Any message acceptance is strictly based on public-key based message authentication/verification for *accountability*, i.e., *non-repudiation* is not sacrificed, unlike symmetric key based approaches. This further enables drastic *misbehavior detection*, also exploiting the newly introduced facilitators, based on probabilistic signature verification and cross-checking over multiple facilitators verifying the same message; while maintaining verification latency low even when under attack, trading off modest communication overhead. Our facilitators can also be used for efficient discovery and verification of *DENM* or any *event-driven message*, including *misbehavior evidence* used for our scheme. Even when vehicles are saturated by adversaries mounting a clogging DoS attack, transmitting high-rate bogus CAMs/DENMs, our scheme achieves an average 50ms verification delay with message expiration ratio less than 1% - a huge improvement over the current standard that verifies every message signature in a First-Come First-Served (FCFS) manner and suffers from having 50% to nearly 100% of the received benign messages expiring.

Keywords: Accountability, Non-repudiation, Privacy, Pseudonymous authentication, Efficiency

1. Introduction

Standardized Cooperative Awareness Messages (CAMs) and Decentralized Environmental Notification Messages (DENMs) enable Vehicle-to-Vehicle (V2V) data exchange [1, 2, 3]. CAMs, or *safety beacons*, are broadcasted at a rate from 1Hz to 10Hz, allowing each vehicle to maintain a view of neighboring vehicle mobility. Event-driven DENMs inform about, for example, abnormal or hazardous road situations. Standards mandate that CAMs and DENMs are secured, in particular pseudonymously authenticated, for security and message unlinkability across distinct pseudonym lifetime periods. More specifically, they are digitally signed (Elliptic Curve

Digital Signature Algorithm (ECDSA)) with each vehicle holding short-lived private/public keys and anonymized certificates, termed pseudonyms or Pseudonymous Certificates (PCs) [4, 5, 6, 7, 8, 9]. ECDSA is used instead of the more popular Internet/Web RSA algorithm, because Elliptic Curve (EC) key sizes and ECDSA signatures are much smaller than RSA ones for the same security level [10] (at the expense of higher signature verification delay).

High rate communication, often, dense Vehicular Communication (VC) network neighborhoods and frequently changing PCs, for improved unlinkability, create an inherent asymmetry: each sent message must be validated by all neighbors and, at each vehicle, incoming messages need to be validated within a short time and have an expiration deadline. ECDSA signature verification delay, τ , is in the order of milliseconds [11, 12, 13, 14, 15, 16, 17]. Even in moderately dense network settings, with a $\tau = 4 \text{ msec}$,

*Corresponding author

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

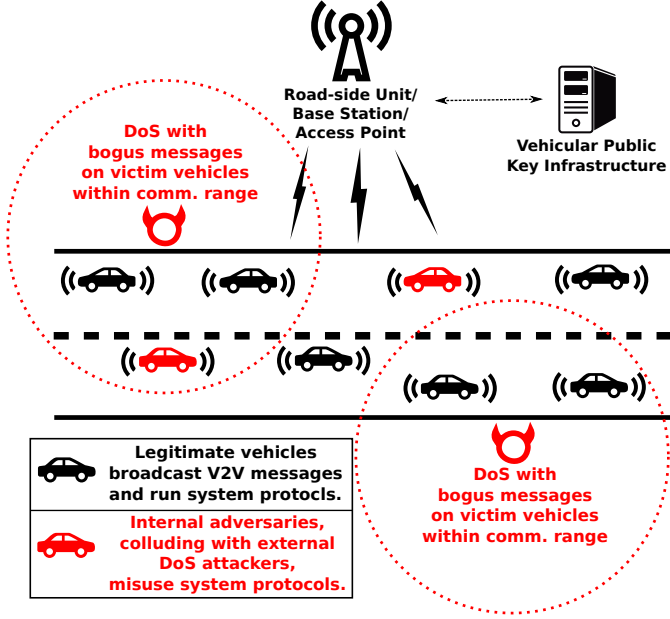


Figure 1: Vehicular Communication (VC) system under DoS attacks.

an On-Board Unit (OBU) can verify up to 250 messages per second; that is, for a typical rate of 10 CAMs/second transmitted by each OBU, from 25 neighboring vehicles. The larger/denser the neighborhood, the higher the traffic; around 2000 CAMs/second could be sent for the 802.11p default data rate of 6 Mbps [18] (CAMs of 300 bytes [19]), an order of magnitude more messages than what a typical OBU can cryptographically handle.

Adversaries can exploit this asymmetry by transmitting fast to generate bogus messages, with bogus signatures and PCs, forcing their receiving neighbors to validate them all, as illustrated in Fig. 1¹. Receivers will reject bogus messages, but validating one or two signatures per bogus message delays benign message processing, easily beyond the expiration deadline. Such clogging Denial of Service (DoS) attacks can target and affect both CAMs and DENMs (or any secure event-driven messages); high verification latencies and high expiration ratios, especially for highly critical DENMs, could be fatal.

Challenges. Handling security overhead in VC, important even without clogging DoS, has received attention: optimizations [20, 21, 11, 22, 23] reduce beacon size and skip PC validation if already verified and cached locally; with limited or no effect for high-rate beacon arrivals imposing excessive computation overhead. Combining public key and symmetric key based beacon authentication reduces overhead [24, 25, 26]. But such an approach has two shortcomings: it cannot provide an efficient way to discover that first beacon (requiring the sender’s signature and its PC verification) to ‘bootstrap’ subsequent symmetric key beacon verification. Moreover, it does not provide non-repudiation and accountability for symmetric-

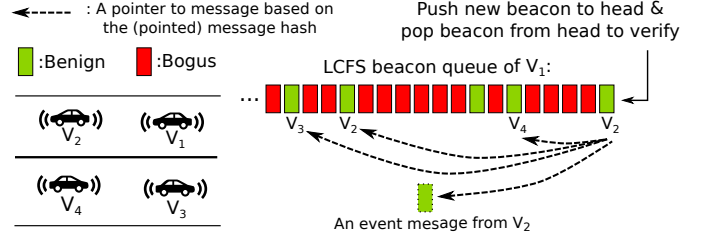


Figure 2: Illustration of cooperative verification under a DoS attack.

key-authenticated messages (see the explanation in Sec. 2 with Fig. 3); both mandatory for VC security requirements [27, 28] (*Challenge 1*).

Cooperating vehicles can share their verification efforts in order to accelerate queue processing [29, 30], but fail to protect vehicles from clogging DoS attacks. Moreover, there is no efficient way to detect misbehavior sharing false verification results (*Challenge 2*). The challenge lies exactly in that adversaries can transmit messages with bogus signatures and PCs forcing their receiving neighbors to verify signatures before dropping the bogus messages. What is lacking is an efficient approach to filter out bogus messages without verifying every message signature while preserving non-repudiation and accountability.

Last but not least, in spite the aforementioned works to render beacon verification efficient [20, 21, 11, 22, 24, 25, 26, 29, 30, 23], the literature has not considered DoS attacks targeting *event-driven messages* (or event messages, for simplicity) (*Challenge 3*). Unlike periodic high-frequency safety beacons, each event message (e.g., DENMs) could be independent from earlier messages. It can also carry information of varying vehicle safety criticality. Thus strict signature verification is necessary for event messages, without alternative efficient verification approaches (applicable for safety beacons). Simply put, high verification latencies and high expiration ratios caused by DoS attacks could be fatal.

Contributions. Concerned with cryptographic message verification², we address the three aforementioned challenges: we extend traditional, standard-compliant pseudonymous authentication for VC, integrating cooperative beacon verification and symmetric-key constructs to accelerate message triage. Our scheme encompasses/safeguards both CAMs (or beacons) and DENMs from clogging DoS, and to the best of our knowledge, is the first to support DoS-resilient V2V DENM validation. In summary, our contributions here are:

1. Non-repudiable DoS-resilient CAM and DENM authentication, strictly based on public key cryptography, assisted by resilient cooperative verification and neighbor vehicle discovery.
2. Misbehavior detection with probabilistic cooperative verifier checking and verifier cross-checking.

¹We explain in Sec. 3 the internal adversary.

²Message content validation is out of the scope of this paper; addressed by approaches such as [31, 32] orthogonal to our scheme.

3. DoS-resilient event-driven message verification, including DENM or any event-driven messages (e.g., misbehavior evidence used for our scheme).

We extend safety beacons with extra fields (e.g., message verification facilitators) that enable verification of queued beacons and facilitate benign event message verification. Intuitively, verifying one beacon signature can possibly validate more than one beacons, thus expediting message verification under high message arrival rate. Fig. 2 shows an example of cooperative verification with a Last-Come-First-Served (LCFS) queue processing (see Sec. 4 for the explanation of the LCFS design choice). Vehicle V_1 pops a beacon from the queue head. Once this beacon, sent by V_2 , is verified, the beacon can (cooperatively) allow verifying a beacon from V_3 and a beacon from V_4 . It can also validate immediate earlier beacons from V_2 , if not verified yet. It also carries a facilitator that points to the upcoming event message, to help receiver discovering the corresponding benign event message among bogus event messages. The event message was already triggered (and generated) before the beacon dissemination, but it was artificially delayed to facilitate the event message reception and verification. Although our scheme delays the event message dissemination, still, it outperforms the traditional approach under DoS attacks (see Sec. 6).

Hash chain elements are attached to beacons, to efficiently eliminate masqueraded beacons. Once a neighboring vehicle is discovered (based on beacon reception and validation), the hash chains help keeping track of subsequent beacons from the already discovered neighboring vehicles, which still need to go through signature verification to achieve non-repudiation and accountability unlike prior approaches [24, 26, 33] (*Contribution 1*). When overloaded by high rate benign beacons or DoS attacks, vehicles can expedite the discovery of new neighboring vehicles leveraging piggybacked information on the (discovered) neighbors' beacons. Moreover, probabilistic signature checking (of cooperatively verified beacons) and cross-checking of multiple validation elements effectively thwart malicious nodes that attempt to exploit the cooperative verification (*Contribution 2*). The event message facilitators in CAMs help to efficiently discover the corresponding upcoming DENMs or any event-driven messages, including misbehavior evidence introduced for our scheme in Sec. 4 (*Contribution 3*).

What we are after in this paper is a DoS-resilient efficient beacon verification scheme that preserves non-repudiation. Our scheme introduces extra communication overhead but provides timely beacon and event message validation under DoS attacks; something impossible for standardized secure VC. Moreover, our approach is agnostic to the underlying communication technology as it is proposed based on the functional specification of the safety application.

In the rest of the paper, we discuss related works (Sec. 2) and explain the system and the adversarial

models (Sec. 3). Then, we provide a detailed description of our scheme (Sec. 4). We analyze the security properties fulfilled by our scheme (Sec. 5), we quantitatively evaluate the scalability and resilience to DoS attacks and misbehaving malicious nodes, with simulations based on realistic vehicle mobility, communication module and processing delays (Sec. 6), before concluding (Sec. 7).

2. Background and Related Works

A number of studies have focused on addressing cryptographic computational overhead in V2V communication and mitigating DoS attacks. This section provides an overview of these related works, addressing the problem at hand with various viewpoints, with strengths and weaknesses in terms of functional and security properties we are after in this paper, summarized in Table 1.

Hardware Acceleration: This can be a solution for handling cryptographic overhead in an energy efficient manner with relatively lower cryptographic processing delays. However, flexibility is lacking: a hardware accelerator typically supports one or a few algorithms, and can fail in handling changes, in our context, in supporting new cryptographic primitives, e.g., new elliptic curves. Moreover, it may not be realistic to expect that powerful yet costly hardware is universally available for all vehicles on the road. On one hand, solutions available for both powerful or budget devices are necessary. On the other hand, if powerful low cost cryptographic hardware were universally available, a security level increase would be necessary to counter brute force attacks, which would then lead to higher cryptographic processing delays [38, 39]. As a result, hardware acceleration would still be insufficient to counter clogging DoS.

Recent results regarding hardware acceleration indeed indicate a lack of resilience to clogging DoS attacks. For example, a recent hardware accelerator for brainpoolP256r1 signature verification [17] takes around 1 ms with a standard double-and-add point multiplication [40]. It is only able to handle at most 1000 signature verification per second - much less than the 2000 messages per second the default 802.11p data rate could allow an adversary to transmit. One should additionally consider that expected performance in a real world deployment could be worse than the optimistic hardware benchmarks. Our proposal thwarts clogging DoS without specific hardware performance requirements, remaining effective as on-board processing (and data rates and security levels) may increase in the future.

Security Overhead Optimization: The baseline, the standard approach is that vehicles perform signature verification for each and every received message in a First-Come First-Served (FCFS) manner. Optimizations [11, 23] reduce the number of required signature verifications on PCs, but not for the CAMs/DENMs. Context-adaptive beacon verification [21] processes message signatures probabilistically based on the context, minimizing the effect of

Table 1: Functional and security properties achieved by related works and our scheme.

Scheme	Hardware Flexibility ^a	CAM Support	DENM Support	Non-repudiation	DoS-resilience	Efficient Neighbor Discovery
Hardware Acceleration [17]	✗	✓	✓	✓	✗	✗
PC Verification Optimization [11, 23]	✓	✓	✓	✓	✗	✗
Context-adaptive Verification [21]	✓	✓	✗	✓	✗	✗
Symmetric Key-based Authentication [34, 24, 25, 26]	✓	✓	✗	✗	✓	✗
Cooperative Verification [29, 30]	✓	✓	✗	✓	✗	✗
DoS-resilient Cooperative Verification [33]	✓	✓	✗	✗	✓	✓
Physical Layer Fingerprinting [35]	✗	✓	✓	✓	✗ ^b	✗
Puzzles for V2I Communication [36, 37]	✓	✗	✗	✓	✓	✗
Our Scheme	✓	✓	✓	✓	✓	✓

^a The hardware flexibility column indicates the solutions rely on special hardware (✗) or are flexible in required hardware (✓).

^b This solution relies on non-cryptographic-based bogus message filtering with high inaccuracy and there is no evidence on fingerprinting efficiently, compared to cryptographic signature verification.

bogus beacons, but it is not able to efficiently filter out beacons forged precisely to mislead that they were sent by newly encountered neighboring vehicles.

Neighboring vehicles can collaborate to verify received beacons in order to benefit each other with locally and independently performed signature verifications [29, 30]. Vehicles share own recent message validation results, leveraging their own beacon disseminations so that a signature verification can enable the receiving neighbor to verify more than one beacons in its local queue [30]. However, cooperative verification requires an efficient approach to find out benign messages, among high rate bogus messages, to make use of shared verification. Alternatively, vehicles can leverage Roadside Unit (RSU)-aided collaborative verification based on ID-based signcryption [29]; each vehicle verifies a subset of messages and is informed about the rest of the message validations from other vehicles. However, processing delays of this RSU-aided approach are not provided, making it unclear whether the approach could outperform traditional public-key cryptography based schemes. Moreover, the approach considers verification of a set of given messages, but does not consider dynamic and continuous message reception and verification, as is the case for high-rate safety beacons (CAMs). Last but not least, the scheme is not resilient to compromised vehicles. Although the above optimizations relieve the computation overhead to a certain extent, essentially aiming at lower average message verification delay (namely τ in this paper), an overwhelming bogus beacon rate could force vehicles to dedicate/waste the majority of computation resource in verifying bogus signatures. Our proposal addresses this challenge by efficiently discovering potentially valid benign messages and filtering out bogus mes-

sages.

Symmetric Key based Authentication: Prediction-based approaches rely on the predictability of vehicle status given practical mobility limitations [25, 26]. However, the prediction-based approaches are not resilient to packet loss, which is common in a loaded Vehicular Ad-hoc Network (VANET). Timed Efficient Stream Loss-Tolerant Authentication (TESLA) [41]-based schemes [34, 24] leverage one-way hash chains as symmetric keys to authenticate beacons while coping with packet loss [34]. However, symmetric key based authentication is possible only after at least one beacon is verified, so that the hash chain elements can be connected to the authenticated chain anchor. Moreover, creating the asymmetry with the help of delayed symmetric key disclosure precludes non-repudiation and accountability because message authenticators can be forged once the symmetric keys are disclosed. DoS-resilient secure communication in wireless sensor networks [42] has been proposed to defend against clogging DoS attacks, discovering potentially valid messages leveraging hash chains. However, the trust establishment on the hash chain in a dynamic network is not addressed, clearly needed in VC systems. Our scheme adopts and extends the DoS-resilient message discovery [42], as a feature towards achieving DoS-resilient VC systems.

To the best of our knowledge, only [33] provides DoS-resilient features for newly encountered vehicle discovery. However, similar to [34, 24, 26], it relies on symmetric key based authentication for efficient beacon validation, which fails providing non-repudiation. Beacons can be properly signed on top of symmetric key based authentication in order to protect against misusing the absence

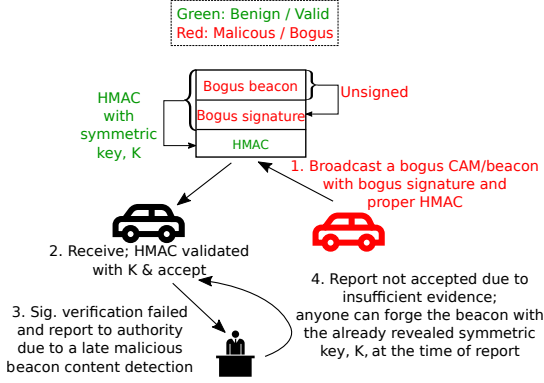


Figure 3: An attacker exploits the lack of non-repudiation in the symmetric key chain based solution.

of non-repudiation [24, 33], while receivers choose to verify message signatures if message non-repudiation is required. However, it is hard to define conditions for signature verification, while late detection of misbehavior could result in benign vehicles being misled by considerable amount of fake beacons without making the malicious vehicles accountable for their actions as shown in Fig. 3. Our scheme ensures non-repudiation of message authentication and accountability of vehicles by strictly accepting messages based on public key cryptography.

All the above approaches exploit high beacon frequency and correlation of successive beacons, but they cannot thwart DoS attacks on event-driven message. Event messages does not follow a specific time pattern, and are triggered by less correlated or independent events. Our proposal is the first to address this.

Physical Layer Fingerprinting: Radio frequency fingerprinting [35] can fingerprints the transmitting (Tx) devices based on their physical layer characteristics, e.g., frequency offset stemmed from imperfect hardware. This can be used to detect the messages sent from the same Tx devices even though they have different/changing higher layer network identifiers (e.g., IP addresses or MAC addresses), which can further efficiently filter out these excessive rate of bogus messages sent by adversarial Tx devices. However, there is no evidence that fingerprinting can be done efficiently, which could be even computationally heavier than signature verification. Moreover, fingerprinting approaches could be different for different communication technologies and obfuscation on the physical layer [43] could counter solutions relying on physical layer fingerprinting. Our proposal achieves resilience to DoS by augmenting software implementations without reliance on lower layer technologies that require modification of OBU communication modules and warrant a separate future investigation.

Solutions for Other Domains: Human effort or client computation resource can be involved to defend against DoS attacks. Captcha-based solutions are hard to machine automate as they require user input on their interfaces [44]. It is straightforward that requiring

user action or input in frequent V2V communication is unrealistic, especially when driving a car. Puzzle-based schemes [36, 37] can protect against DoS attacks on Vehicle-to-Infrastructure (V2I) communication for pairwise connections, but do not fit in the context of safety beacons and event-driven messages, which require low latency in connectionless V2V communication. Solutions against physical layer jamming [45] are orthogonal to our scheme and can co-exist with our scheme.

3. System and Adversary Models

In this section, we explain system and adversary model we consider in this paper, illustrated in Fig. 1.

3.1. System Model

Vehicles, termed *nodes*, are equipped with OBUs. Nodes exchange messages over multiple network interfaces, e.g., Dedicated Short Range Communication (DSRC) [46, 47] and possibly intermittently, the Internet through WiFi or cellular networks; their clocks are synchronized through Global Navigation Satellite System (GNSS) modules or Network Time Protocol (NTP) servers. Nodes are issued short-term credentials, Pseudonymous Certificates (PCs) by a Vehicular Public-Key Infrastructure (VPKI) [4, 5, 6, 7, 8, 9], and sign their messages with the corresponding private keys. Upon a PC change, all protocol stack identifiers, including IP and media access control addresses, change [48].

Each vehicle broadcasts beacons at a rate, γ , with the maximum rate is $\gamma_{max} = 10Hz$ (i.e., 10 beacons per second). Event-driven messages can include not only standardized DENMs [3] but also any V2V messages triggered by specific events (including misbehavior evidence that we introduce in Sec. 4). Due to their event-driven nature, they do not follow specific time patterns, while an event message can be repeated more than once based on its criticality [3]. All received (benign) messages must be verified to ensure VC security.

3.2. Adversary Model

External adversaries: We are concerned with clogging DoS attacks exploiting computationally expensive signature verification (or validation; the two terms are used interchangeably). We do not dwell on DoS attacks on the physical or medium access control layers. We focus on single-hop transmissions; multi-hop VC transmissions (e.g., Geocast packets [49]) are out of scope. An adversary can flood with bogus messages carrying fake signatures (i.e., random bits with same lengths as authentic signatures that are very fast to generate), irrespective of whether the adversary is a legitimate node or not [42]. We do not consider internal adversaries flooding with properly authenticated messages, because an abnormally high message frequency can be trivially detected and attributed

to each PC and eventually the long-term sender identity, followed by eviction [9].

An attacking node that floods bogus messages utilizing its full bandwidth can broadcast, for example, at a rate higher than 2000Hz considering a typical 6Mbps bandwidth for IEEE 802.11p [50, 51] and 300 bytes V2V messages [11]. Attackers attach (random) bogus signatures and plausible PCs so that receiving nodes have to deem the messages useful and verify the signatures. In the same spirit, they set randomized IP and media access control addresses for each beacon they transmit, so that receiving nodes cannot easily impose rate control based on the addresses. This is equivalent to an aggregate message rate from 200 neighboring vehicles broadcasting at $\gamma = 10$ messages per second, i.e., 10Hz. Multiple such attacking nodes can be deployed (simple adversary controlled devices or malware on infected OBUs) in a targeted area, essentially launching a distributed DoS attack. All benign nodes within the communication range are victims and have to proceed with signature verification to check validity of the messages. Unlike defenses against DoS towards client-server based architectures, e.g., [36, 37], a DoS-resilient V2V communication scheme should not introduce significant overhead or delay on the sending process that would degrade the overall performance of VC applications.

Internal adversary: Compromised nodes, termed *malicious nodes* in the rest of the paper, are equipped with valid credentials, and can actively inject authenticated false data to the network, in order to mislead other nodes. This is particularly relevant for cooperative verification: one adversary could falsely inform its neighbors that it validated bogus beacons transmitted in the same neighborhood. We are concerned with message signature validity, while we do not dwell on actual location or content verification on the authenticated beacons; these can be addressed with position verification [52, 53] or content validation [31, 32] approaches.

3.3. Requirements

Message and entity authentication: The message sender must be (pseudonymously) authenticated, allowing receivers to corroborate the sender legitimacy and verify they were not altered or replayed.

Non-repudiation and accountability: Nodes should not be able to deny actions performed, thus messages sent. They are accountable for their messages or actions, and any message transmission mandated our scheme should be non-repudiable. Any misbehaving node should have their long-term identities revealed and, if needed, evicted from the system.

Ano-/Pseudo-nymity and unlinkability: Messages should not be linkable to their sender's long-term identity. They should be linkable only over a protocol-selectable period, i.e., over a pseudonymous identity validity period.

Availability: Nodes should maintain their ability to timely validate legitimate messages (CAMs and DENMs)

Table 2: Notation

PC	<i>Pseudonymous Certificate</i>
PRL	<i>Pseudonymous Certificate Revocation List</i>
KRL	<i>Key Chain Revocation List</i>
$\{msg\}_{\sigma_{PC}}$	<i>Signed message attached signature and PC</i>
Pr_{check}	<i>Probability of checking cooperative verifier</i>
α	<i>Number of message facilitators in a beacon</i>
β_1	<i>Event message facilitator repeat counter</i>
β_2	<i>Event message repeat counter</i>
γ	<i>Beacon frequency</i>
k	<i>Number of self-chained verifiers in a beacon</i>
τ	<i>Average message verification delay</i>
T_{blife}	<i>Beacon lifetime</i>
$H()$	<i>Hash function</i>
$MAC / MAC_K(msg)$	<i>Message Authentication Code value / $H(K \parallel msg)$</i>
$Queue_{recv}$	<i>Beacon reception queue</i>
$Queue_{check}$	<i>Checked beacon queue</i>
v / Ver	<i>Beacon verifier / Beacon verifier set</i>
f / F	<i>Message facilitator / Message facilitator set</i>

even amidst an adversarial flood of fictitious traffic. We do not impose strict requirements on message verification deadline. Rather, the verification delay should be within the order of magnitude of message dissemination intervals (e.g., beacon intervals or event triggering intervals), in order to ensure persistent neighbor awareness and timely environment notification awareness, given certain extent of packet loss and vehicle mobility predictability.

4. Proposed Scheme

4.1. Overview

For timely validation of safety beacon and event-driven messages even under DoS attacks, our scheme extends pseudonymous authentication with several DoS-resilient features: (i) efficient (bogus) beacon filtering based on hash chains, (ii) cooperative verification, (iii) self-chained verification, and (iv) event message verification facilitators. We trade off communication overhead of such message verification facilitators for faster message verification. To enhance security, our scheme detects misbehaving nodes by probabilistically checking and cross-checking messages, to render cooperative verification robust to internal adversaries. Table 2 summarizes the notation used in the rest of the paper.

Tracing benign beacons (and filtering out bogus beacons): Nodes in our scheme are issued short-term PCs by the VPKI. We consider a Sybil-resilient PC lifetime policy [7, 9], so that each node is equipped with PCs with non-overlapping lifetimes. Nodes authenticate messages by signing with their private keys corresponding to the PCs. In addition, nodes maintain a key chain (essentially a hash chain) for each of their PCs. Each key chain element is used as a (symmetric) *one-time beacon authentication key* (Sec. 4.2.1), authenticating strictly one beacon that broadcasted in the corresponding time interval. These key chains, once verified, assist the receiving nodes in filtering out bogus beacons at a lower cost than performing signature verifications (Sec. 4.2.2). Sending nodes piggyback key chain elements to their disseminated beacons and the receivers queue or drop the beacons based

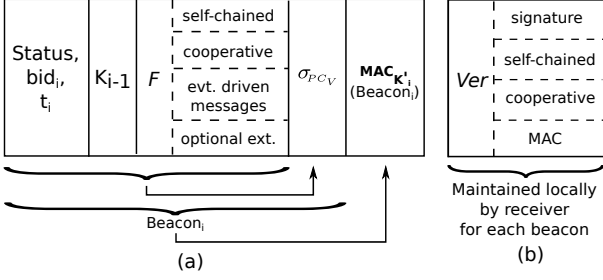


Figure 4: (a) The extended beacon format and (b) the locally maintained beacon verifier set.

on correctness of the one-time keys. The extended beacon piggybacking one-time key, as shown in Fig. 4, is:

$$B_i = \{S, bid_i, t_i, K_{i-1}, F\}_{\sigma_{PC}}. \quad (1)$$

It comprises “traditional” beacon fields, e.g., vehicle status (S , including location, velocity, direction, etc.), beacon id (bid_i) and timestamp (t_i), the corresponding one-time key (K_{i-1}), and message facilitators (F , explained below). Each signed beacon carries a one-time keyed Message Authentication Code (MAC). Unlike TESLA [41] based beacon authentication [24, 26], our scheme merely uses validation of their MACs to filter out bogus beacons and cross-check the correctness of cooperative verification (Sec. 4.2.4), while message acceptance in our scheme relies on signature verification, self-chained verification or cooperative verification. The actual broadcasted beacon message is:

$$M_i = \{B_i, MAC_i\}. \quad (2)$$

Expediting node discovery and beacon verification: Tracing and filtering beacons, however, is possible only after key chains are verified, i.e., at least one signed beacon (that carries a one-time key) from a neighboring node is verified (thus the node is discovered). To expedite node discovery (and, in general, beacon verification), apart from continuous queued beacon verification (Sec. 4.2.3), nodes share *cooperative verifiers* (as one type of message facilitator) that point to beacons they verified based on signatures. Sharing cooperative verifiers help neighbors to validate beacons or discover new neighbor nodes. More specifically, received cooperative verifiers could point to locally queued beacons of either non-discovered nodes or discovered nodes. The former ones are kept in a special-purpose queue, used for node discovery: the beacons in this queue wait to be verified based on signatures. The latter ones validate corresponding beacons with probabilistic signature checking, in order to detect any internal adversary that disseminate fraudulent cooperative verifiers for previously broadcasted bogus beacons. Strictly performing signature verification for non-discovered nodes’ beacons is important, because this sets a basis for subsequent efficient beacon verifications, establishing trust on both the vehicle status and key chain.

Apart from the signature verifier (the signature itself) and the cooperative verifier, beacon verification can also rely on the self-chained verifier. Each beacon carries the latter, pointing to messages disseminated in the immediate past, to verify more than one beacons by the same sender upon a single signature verification.

Misbehavior detection: The redundant verifiers for the same message, maintained by the receiver in a local data structure, Ver (Fig. 4), are cross-checked to detect misbehavior. The MAC, while not sufficient for non-repudiable beacon verification, can be used as an extra reference to cross-check any former verifier. If the verifiers conflict, the beacon signature must be examined, to reveal the faulty verifiers. Signature or self-chained verifiers both suffice to establish message validity, because they both rely on the verification of signature generated by the message sender. The source of a faulty verifier is blacklisted locally (and reported to the VPKI).

Facilitating event-driven messages: Beacons piggyback event message facilitators for DoS-resilient event message verification (Sec. 4.2.8). A facilitator is essentially the hash value (thus verifier) of an event message, but it is not used to verify the message. Rather, each event message facilitator is cached locally to wait for matching (artificially delayed) event message. An event message that does not match any locally cached facilitator is dropped. Event messages are always verified based on signatures, to ensure non-repudiation of the claimed events. Such matching based on proactive facilitator dissemination can help receivers to efficiently catch potentially valid event-driven messages, while filtering out irrelevant bogus messages.

We show a high-level flowchart of our scheme with Fig. 5, before detailed explanation on each scheme component. Each node chooses a beacon to verify from the local queue, $Queue_{recv}$ and $Queue_{check}$, and verifies the signature. We explain the purpose of the two queues when we present in detail the beacon verification below (Sec. 4.2). If the signature is valid, MACs of previous beacons (if any not verified) by the same sender are verified based on the symmetric key in this beacon. Each facilitator can be used to verify self-chained beacons or cooperatively validated beacons, or simply stored to wait for matching event-driven messages. For each cooperatively validated beacon, if the beacon sender was not discovered, then this first-validated beacon serves for node/neighbor discovery; otherwise, the beacon is accepted or probabilistically checked based on a probability Pr_{check} .

4.2. Scheme Components

4.2.1. Beacon Chaining

A unique key chain is generated by each node for each own PC. A key chain pertains to a PC lifetime and broadcasted beacons throughout the PC lifetime. The length of a key chain is the number of beacon that can be broadcasted under a PC. With $\gamma_{max} = 10Hz$, the length of a key chain $L = \tau \cdot \gamma_{max}$. Therefore, it guarantees that the

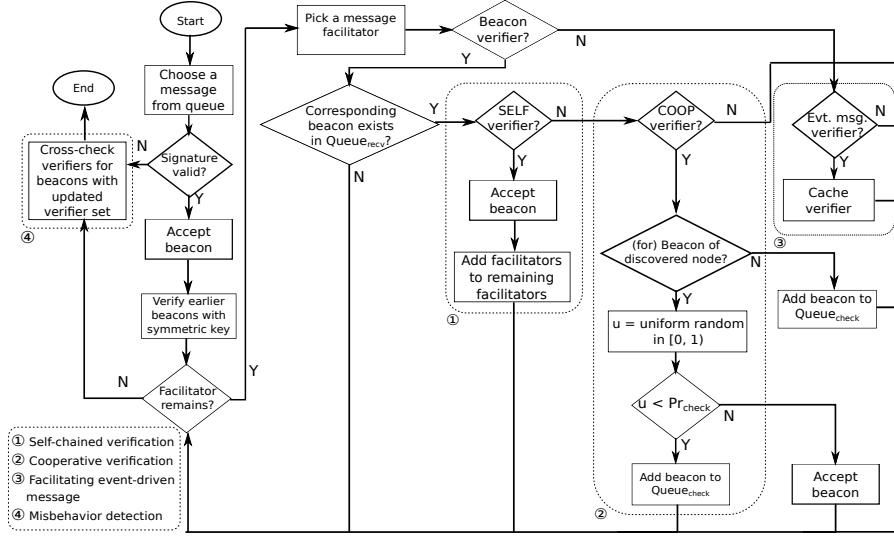


Figure 5: Flowchart of message verification.

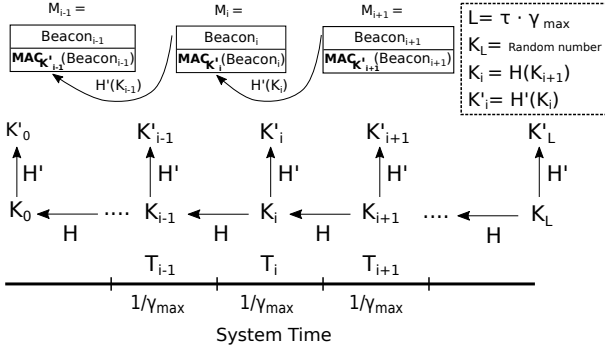


Figure 6: Beacon chaining.

key chain elements are sufficient for authenticating all beacons that could be disseminated with the corresponding PC. Fig. 6 shows key chain generation and beacon chaining process. The system time is divided into equally-sized time slots, and the length of each time slot is $1/\gamma_{max}$ (i.e., 100 msec): the shortest possible beacon interval according to the standard [54]. Each K'_i is used as an authentication key to compute a MAC for a beacon disseminated within the time slot, T_i , and the corresponding K_i is disclosed with the next beacon in the next time slot, T_{i+1} . Therefore, only the key chain owner has access to the key that can be used to authenticate the beacon in the current time interval/slot. Once a symmetric key is disclosed, it can be only used to verify MACs of earlier beacons, while any message authenticated with a disclosed symmetric key (by attackers) will be dropped. The trust on key chain is established based on signature verification. Once a beacon is verified, the one-time key on that beacon can be trusted and the rest of key chain elements can be verified.

In general, key chain generation is not an issue, considering the low computation overhead of hash functions. Consider a vehicle pre-generates key chains for a day, while

parked during the midnight. The vehicle requires at most 864,000 one-time keys for a day (i.e., 24 h, while an actual trip duration during the day could be much shorter) with $\gamma = \gamma_{max}$. Thus, the maximum required storage capacity is $864,000 * Size_{digest}$. With a $Size_{digest} = 160bits$, (e.g., SHA-1³), the required storage would be only around 14 MB, not an issue for storage in modern mobile device.

4.2.2. DoS-resilient Beacon Reception

Nodes handle received beacons according to Algorithm 1, before their actual signature or cooperative verifications. Each node maintains a local Pseudonymous Certificate Revocation List (PRL) that includes both periodically downloaded PRL from the VPKI and PCs of locally detected misbehaving nodes (not yet announced with the latest VPKI PRL). Trust on a key chain can be established once a beacon is verified based on its signature, and we consider the node as discovered. The DoS-resilient node discovery process is illustrated with Algorithm 5 (Sec. 4.2.6).

Once a beacon is received, the node checks whether the attached PC is revoked or not (1:1-3, i.e., lines 1-3 in Algorithm 1). Beacon messages that pass the PRL checking are extended with an empty verifier set (Fig. 4), Ver_{B_i} (1:4), storing upcoming verifiers to be cross-checked. There are four types of verifiers: signature verification (termed *SIG* verifier), self-chained (*SELF*) verifier, cooperative (*COOP*) verifier, and *MAC* verifier; we explain each verifier with in Secs. 4.2.3 to 4.2.6. If the PC owner was already discovered, the receiver fetches the latest cached information (i.e., $\{PC, K_{i-1}, t_i, Bs\}$) for the PC (1:5-6). PC_Bs stores information corresponding to each discovered or non-discovered nodes. For each

³Although collision attack has been successfully applied on SHA-1 [55], our scheme relies on second-preimage resistance of hash function, which is much harder to attack.

Algorithm 1: DoS-resilient Beacon Reception

```

1 Input:  $\{B_i, MAC_i\} = M_i$ 
2  $\{S, bid_i, t_i, K_{i-1}, F\}_{\sigma_{PC}} = B_i$ 
3 Check  $PC$  against PRL; proceed or drop  $M_i$  accordingly.
4  $M_i = \{B_i, MAC_i, Ver_{B_i} = \{\}\}$ 
5 if  $PC \in PC\_Discovered$  then
6    $\{PC, t_{i'}, K_{i'-1}, Bs\} = Search(PC, PC\_Bs)$ 
7   if  $t_i \in T_i$  and  $H_C^{i-i'}(K_{i-1}) == K_{i'-1}$  then
8      $Queue_{recv} = \{M_i\} \cup Queue_{recv}$ 
9     Update  $PC\_Bs$  with  $\{PC, t_i, K_{i-1}, \{M_i\} \cup Bs\}$ .
10     $M' =$  the latest beacon from  $Bs$ 
11    if  $M' \neq \{\}$  then
12      Input  $M'$  to Algorithm 3
13  else
14    Drop  $M_i$ .
15 else
16   if  $PC \in PC\_Bs$  then
17      $\{PC, nil, nil, Bs\} = Search(PC, PC\_Bs)$ 
18     if  $K_{i-1} ==$  any one-time key in  $Bs$  then
19       Drop  $M_i$ 
20     else
21        $Queue_{recv} = \{M_i\} \cup Queue_{recv}$ 
22       Update  $PC\_Bs$  with  $\{PC, nil, nil, \{M_i\} \cup Bs\}$ .
23   else
24      $Queue_{recv} = \{M_i\} \cup Queue_{recv}$ 
25      $PC\_Bs = PC\_Bs \cup \{\{PC, nil, nil, \{M_i\}\}\}$ 

```

discovered node, it stores the latest one-time key ($K_{i'-1}$) and timestamp ($t_{i'}$) fetched from the latest beacon, and the list of non-expired beacons (Bs) that successfully “passed” hash chain checks (explained below). For the non-discovered nodes, it stores only the beacon list, and the other two fields are updated once the node is discovered (if any benign beacon exist in the beacon list). With the received beacon, the receiver checks the correctness of K_{i-1} against $K_{i'-1}$: the number of intervals between the two keys should be consistent with the difference between t_i and $t_{i'}$ (1:7). This preliminary hash chain check can efficiently drop bogus beacons attached incorrect or replayed overheard disclosed one-time keys. The beacon is then queued for verification and the information in PC_Bs is updated (1:8-9). The MAC of the latest beacon (before this reception) in Bs is checked with Algorithm 3 (1:10-12; see Sec. 4.2.4). Any received beacon, that fails hash chain test, is simply dropped (1:13-14). If the sender node is not yet discovered, the receiver checks whether the piggybacked one-time key was seen from any previous beacon attached the same PC; if not, the beacon is added to Bs of the PC and queued for verification (1:16-25).

Although actual beacon lifetime (the time duration each beacon stays in OBU upon reception) depend on various system parameters (e.g., content relevance and beacon rate), without loss of generality, we assume each beacon is given a lifetime, T_{blife} . Without such a lifetime, OBU memory may be saturated by outdated beacons, which could never be verified due to higher message arrival rate than message processing rate. Both newly received beacons, and verified beacons are kept for T_{blife} to cross-

check verifiers for the same beacon. Moreover, existing schemes, e.g., for content verification [56] and adaptive beacon rate [57], rely on message redundancy, thus require the verified messages to be stored for an extra period.

4.2.3. Beacon Verification

Algorithm 2: Beacon Verification

```

1 while  $M = \{\}$  and  $Queue_{check}$  is not empty do
2    $\{B_i, MAC_i, Ver_{B_i}\} = Queue_{check}$  head,  $M$ 
3   if  $M$  is only for node discovery and  $PC$  of  $M \in PC\_Discovered$  then
4     if  $Ver_{B_i}$  contains positive COOP verifier then
5       Accept  $M$  with prob.  $1 - Pr_{check}$ ;  $M = \{\}$ .
6 if  $M \neq \{\}$  and  $Queue_{recv}$  is not empty then
7    $Condition : t_{beacon} + 1/\gamma_{max} > t_{now}$ 
8    $Beacon\_Set_D / ND =$  discovered/non-discovered nodes' beacons in  $Queue_{recv}$  that meet  $Condition$ 
9   if  $Beacon\_Set_D \cup Beacon\_Set_{ND} == \phi$  then
10     $M = Queue_{recv}$  head
11   else if  $time_D / (time_D + time_{ND}) \leq Ratio_D$  or  $Beacon\_Set_{ND} == \phi$  then
12     $M =$  randomly chosen from  $Beacon\_Set_D$ 
13   else
14     $M =$  randomly chosen from  $Beacon\_Set_{ND}$ 
15 if  $M \neq \{\}$  then
16    $\{B_i, MAC_i, Ver_{B_i}\} = M$ 
17    $\{S, t_i, K_{i-1}, F\}_{\sigma_{PC}} = B_i$ 
18    $validity = B_i$  signature validity; update  $time_{D/ND}$ .
19   if  $validity = true$  then
20     Accept  $M$ .
21     if  $PC \notin PC\_Discovered$  then
22       Add  $PC$  to  $PC\_Discovered$ .
23        $\{PC, nil, nil, Bs\} = Search(PC, PC\_Bs)$ 
24       for each  $M_j$  in  $Bs$  from head do
25         if  $M_j$  is  $Bs$  head then
26            $\{B_j, MAC_j, Ver_{B_j}\} = M_j$ 
27            $\{S, bid_j, t_j, K_{j-1}, F\}_{\sigma_{PC}} = B_j$ 
28           if  $t_i$  is within  $T_i$  period and  $H_C^{i-j}(K_{i-1}) == K_{j-1}$  then
29             Update  $PC\_Bs$  with  $\{PC, t_j, K_{j-1}, Bs\}$ .
30           else
31             Drop  $M_j$ .
32         else
33           Input  $M_j$  into Algorithm 3.
34       Input  $\{M_i, SIG\}$  into Algorithms 4 and 5.
35    $v_{new} = \{nil, nil, validity, SIG\}$ 
36    $Ver_{B_i} = \{v_{new}\}$ 

```

Each node maintains two queues: $Queue_{recv}$ and $Queue_{check}$. $Queue_{recv}$ queues newly received beacons for a semi-randomized (explained below) LCFS verification, and $Queue_{check}$ stores potentially valid beacons from non-discovered nodes and beacons chosen for probabilistic checking. $Queue_{check}$ is given higher priority for quicker node discovery and misbehavior detection. We choose the LCFS strategy for beacon verification, because our scheme can efficiently validate older beacons based on self-chained verifier once a newer beacon is verified, and

fresher cooperative verifiers can benefit neighboring nodes to a greater extent, i.e., verifying latest received beacons. Moreover, with high bogus beacon rate under DoS attacks and a given beacon lifetime, LCFS verification guarantees that fresh beacons are verified, instead of the oldest non-expired beacons with a FCFS verification.

If $Queue_{check}$ is not empty, the beacon at $Queue_{check}$ head is chosen for verification. If the beacon was chosen for node discovery, but the beacon sender was already discovered while waiting in $Queue_{check}$, then the beacon is simply accepted with a probability $1 - Pr_{check}$; otherwise, if not yet discovered, the protocol proceeds with signature verification (2:3-5). If the first phase (2:1-5) did not conclude with a beacon for verification, then a beacon will be chosen from $Queue_{recv}$.

In order to guarantee timely benign beacon verification under DoS attacks, we assign explicitly CPU time ratios for verifying discovered nodes' and non-discovered nodes' beacons. Discovered nodes' beacon verifications occupy a time ratio of $Ratio_D$, and non-discovered nodes' beacon verifications occupy a time ratio of $Ratio_{ND}$, where $Ratio_D + Ratio_{ND} = 1$). $time_D$ and $time_{ND}$ maintain time used for verifying beacons from each category. Moreover, a beacon is randomly chosen among beacons that fulfill the condition: $t_{beacon} + 1/\gamma_{max} > t_{now}$, where t_{beacon} is the original timestamp on each beacon. This time condition ensures that cooperative verifiers of relatively fresher beacons are piggybacked; the randomization maximizes the benefit from cooperative verifiers, reducing the chance closely located vehicles verifying (thus piggybacking the cooperative verifiers to) the same beacons, which would be the case if beacons are strictly chosen from their $Queue_{recv}$ heads. The nodes choose a beacon from $Queue_{recv}$ based on the above design choices (2:6-14).

If a beacon is chosen from $Queue_{check}$ or $Queue_{recv}$, the signature is verified (2:15-18). If the valid beacon belongs to a non-discovered node, PC is added to $PC_{Discovered}$ (2:19-22), and all queued beacons attached PC are then checked with MAC (2:23-24,33). The latest beacon attached the correct one-time key is only used to update the cached information (2:25-29), because a newer beacon is necessary for its MAC validation. The valid beacon is then used for self-chained verification and cooperative verification (2:34), and the signature verification result is cross-checked with previous verifiers to detect any misbehaviors (2:35-36). After the conflict checking, this signature verification result replaces all previously stored verifiers, because the new verifier is the definitive *SIG* verifier and any other verifier is unnecessary.

Cooperative verifier maintenance: Each node maintains a list of recently verified beacons based on signatures, to be piggybacked on own beacons as cooperative verifiers. The α latest (in terms of their timestamps) verified beacons are kept. Each of the α beacons could be either a valid beacon, or a bogus beacon that attached discovered node's PC and correct one-time key, i.e., a

Algorithm 3: MAC Validation

```

1 Input:  $\{B_i, MAC_i, Ver_{B_i}\} = M$ 
2  $\{S, t_i, K_{i-1}, F\}_{\sigma_{PC}} = B_i$ 
3  $\{PC, t_{i'}, K_{i'-1}, Bs\} = Search(PC, PC\_Bs)$ 
4  $validity = (H_C^{i-i'}(K_{i-1}) == K_{i'-1} \text{ and } MAC_{H'(K_{i'-1})}(B_i) == MAC_i)$ 
5 if  $validity = true$  and  $M$  was not validated and  $PC \notin KRL$  then
6   Input  $\{M, MAC\}$  to Algorithm 5.
7    $v_{new} = \{pcid_{PC}, bid_{i'}, true, MAC\}$ 
8    $Ver_{B_i} = Ver_{B_i} \cup \{v_{new}\}$ 

```

cooperative verifier could be either positive or negative. A negative cooperative verifier can potentially reveal a misbehavior that malicious nodes attempt to validate the corresponding bogus beacon with a false positive cooperative verifier, because the two cooperative verifiers would conflict. We provide qualitative and quantitative security analysis for the effect of negative cooperative verifiers on malicious node detection in Secs. 5 and 6.

4.2.4. MAC Validation

MACs of discovered nodes' beacons are checked once newer beacons that piggybacked correct one-time keys are received (Algorithm 3). Each node maintains a Key Chain Revocation List (KRL) that stores all detected node PCs attempting to misuse one-time key beacon authentication by attaching correct one-time keys and MACs but bogus signatures. In this algorithm, the MAC correctness is checked first, and, if validated and PC is not in KRL, the *COOP* verifiers in the beacon is used to find potentially valid beacons for non-discovered nodes (3:4-6). The beacon verifiers are cross-checked if the beacon is not already in $Queue_{check}$, and the new MAC verifier is added to the verifier set (3:7-8). If a PC is in KRL, even if the MAC checks are passed, the facilitators in the corresponding beacons will not be used.

4.2.5. Self-chained Verification

Each beacon is piggybacked with k *SELF* verifiers, pointing to immediate previous k own beacons. A *SELF* verifier is a tuple of beacon id (bid) and the corresponding beacon digest. For each *SELF* verifier, the node checks whether a beacon with the same bid is received (4:4-7). If such a beacon exists and was not verified based on *SIG* or *SELF* verifiers, the beacon digest is compared with the one in the verifier (4:8-9). If they are equal, M is accepted and used further for self-chained verification and cooperative verification (4:10-12). The *SELF* verifier is used to cross-check existing verifiers for misbehavior detection (4:13-15). Only this *SELF* verifier is kept (same as *SIG* verifier in Algorithm 2), because it is equivalent to signature verification: the beacon signature corroborates the current beacon and the previous k own beacons.

Algorithm 4: Self-chained Verification

```
1 Input:  $\{B_i, MAC_i, Ver_{B_i}\} = M$ 
2  $\{S, t_i, K_{i-1}, F\}_{\sigma_{PC}} = B_i$ 
3  $\{PC, K_{i'}, t_{i'+1}, Bs\} = Search(PC, Cache_{PC})$ 
4 for each  $f$  in  $F$  do
5   if  $f$  is SELF verifier then
6      $\{bid, Digest\} = f$ 
7      $M' = Search(bid, Bs)$ 
8     if  $M \neq \{\}$  and  $M$  was not validated based on
       signature or SELF verifier then
9        $\{B_j, MAC_j, Ver_{B_j}\} = M'$ 
10       $validity = (H(B_j) == Digest)$ 
11      if  $validity == true$  then
12        Accept  $M'$  if was not accepted.
13        Input  $\{M', SIG\}$  into Algorithms 4 and 5.
14       $v_{new} = \{pcid_{PC}, bid_i, validity, SELF\}$ 
15       $Ver_{B_j} = \{v_{new}\}$ 
```

Algorithm 5: Cooperative Verification

```
1 Input:  $\{B_i, type\}$ 
2  $\{S, bid_i, t_i, K_{i-1}, F\}_{\sigma_{PC}} = B_i$ 
3 for each verifier  $f$  in  $F$  do
4    $\{pcid, bid, validity, Digest\} = f$ 
5    $\{PC, K_{i'}, t_{i'+1}, Bs\} = Search(PC, Cache_{PC})$ 
6    $M = Search(bid, Bs)$ 
7   if  $M \neq \{\}$  then
8      $\{B_j, MAC_j, Ver_{B_j}\} = M$ 
9     if  $H(B_j) == Digest$  then
10      if  $M \in Queue_{recv}$  then
11        Remove  $M$  from  $Queue_{recv}$ .
12      if  $PC'$  (on  $M$ )  $\notin PC\_Discovered$  and
         $validity == true$  then
13         $Queue_{check} = Queue_{check} \cup \{M\}$ .
14      else if  $type == SELF$  or  $SIG$  then
15         $Queue_{check} = Queue_{check} \cup \{M\}$  with
        a prob. of  $Pr_{check}$ ; otherwise,
        accept/reject  $M$  based on
         $validity == true/false$ .
16       $v_{new} = \{pcid_{PC}, bid_i, validity, COOP\}$ 
17       $Ver_{B_j} = Ver_{B_j} \cup \{v_{new}\}$ 
```

4.2.6. Cooperative Verification

The *COOP* verifier, points to beacons previously the beacon sender verified and can be used to verify matching beacons in $Queue_{recv}$ (Algorithm 5). Similar to *SELF* verifier, each *COOP* verifier is a tuple of PC id ($pcid$), beacon id (bid), the claimed validity ($validity$), and the corresponding beacon digest (5:3-6). When a beacon, M , with matching $pcid$ and bid is found, the hash value of M is compared to the verifier (5:7-9). If M is still in $Queue_{recv}$, the beacon is used for node discovery or cooperative verification and it is then removed from $Queue_{recv}$ (5:10-11). If the verifier is positive and the beacon is from non-discovered node, the beacon is added to $Queue_{check}$ for node discovery (5:12-13). If M carried an already discovered node PC and the current *COOP* verifier was from a beacon verified based on *SIG* or *SELF* verifier, M is further checked with a probability of Pr_{check} . Otherwise,

with probability $1 - Pr_{check}$, M is accepted or rejected based on the claimed *validity* (5:14-15). Finally, the new verifier is cross-checked with the existing verifiers, if M is not already in $Queue_{check}$ (5:16-17).

4.2.7. Misbehavior Detection

Misbehavior detection is achieved through cross-checking verifiers to the same beacon. Whenever a new verifier of a beacon is added from any verification, it is compared against the existing ones. If there is any conflict and the beacon hasn't been verified based on signature verification or self-chained verification, the beacon will be pushed to $Queue_{check}$ for signature verification. Both signature verification and self-chained verification results can be used as the proof for beacon (in)validity. Any proven bogus beacon verification attempt based on *COOP* verifier will be reported to the authority, and the misbehavior be added to local PRL. However, MAC verification is insufficient as a misbehavior evidence, because symmetric key based authentication does not provide non-repudiation (see Fig. 3). The misbehavior misusing MAC will be added to KRL, and the follow-up MAC s from the misbehavior will not be trusted.

4.2.8. Event-driven Message Dissemination

Our scheme can be readily used to facilitate event-driven message verification. Due to the event-driven natural, nodes deem event messages highly critical and strictly verify message signatures. Before dissemination of each actual event message, a facilitator for that message is disseminated with an immediate next beacon. Here, the facilitator is an identifier to the event message, including the message digest. Once the beacon is validated based on any verifier, the event message facilitator can be cached. Next, when the actual event message is received, if the message hash value matches the cached message digest, the message is kept for signature verification.

Each event message facilitator is disseminated β_1 times with β_1 consecutive beacons. After each beacon dissemination that carries the facilitator, actual event message is disseminated after the corresponding one-time key is disclosed, i.e., after the next beacon dissemination. For each of β_1 repetitions, actual event messages are disseminated β_2 times to ensure successful delivery. More specifically, event message facilitator is repeated β_1 times and actual event messages are repeated $\beta_1 * \beta_2$ times. Both parameters are flexible and can be adjusted based on message criticality, network condition, etc.

We consider standardized DENM [3] and misbehavior evidence as two examples of event-driven messages in our evaluation (see Sec. 6). Once a misbehaving PC is detected, apart from reporting the evidence to the authority, it can also be disseminated to the neighboring nodes for quicker malicious node eviction.

5. Security and Privacy Analysis

We provide a qualitative security and privacy analysis of our scheme, before a simulation-based quantitative result in Sec. 6.

5.1. Privacy

Our scheme does not introduce any additional privacy concern compared to the standard [2, 3, 11], in terms of message or (pseudonymous) identity linkability. All messages are properly pseudonymously authenticated. Messages, piggybacking one-time keys from the same key chain, are linkable, but they can already be trivially linked based on the attached same PC. Beacon chaining does not exceed the PC lifetime/usage period. Key chain lengths are aligned with PC lifetimes, thus messages are still only linkable over a PC lifetime. Piggybacked cooperative verifiers imply correlations of nodes in terms of their geographical locations (i.e., nodes are within each other's communication range). This is already explicitly available based on the location information included in their messages. Event messages are linkable to their facilitator carriers, however, this was also possible based on the attached same PC.

5.2. Corroborating Legitimate Participation

Vehicular credential verification is a fundamental component for cooperative awareness and safety in VC. All PCs are authenticated and issued by the VPKI. PCs cannot be linked to long-term vehicle identifiers, but it is important to prevent adversaries from introducing any phantom vehicles/nodes with bogus beacons. If messages and the attached PCs are verified, the legitimacy of the neighboring vehicles can be proven. Moreover, issuance of PCs with non-overlapping lifetimes ensure each vehicle is equipped at most one valid PC at any point in time, thus preventing Sybil-based behavior. In our scheme, beacons could be also accepted based on self-chained verifiers or cooperative verifiers. Self-chained verifiers always correspond to already discovered nodes, so that they cannot be used to validate any beacon-attached non-verified PC. We prove below that adversaries are not able to introduce any phantom node by abusing cooperative verifiers, even though cooperative verifiers could point to third nodes.

Theorem 1. *If the underlying public-key cryptography is secure and the VPKI policy mandates issuance of PCs with non-overlapping lifetimes, an adversary cannot introduce phantom nodes/vehicles.*

Proof. In order to introduce a phantom node, an adversary has to disseminate a bogus beacon carrying a forged PC. If such a beacon is chosen from $Queue_{recv}$ for signature verification, it will be rejected immediately due to the verification failure of the beacon signature. The adversary can attempt having the bogus beacon validated with a cooperative verifier piggybacked on an authentic beacon.

The cooperative verifier would need to point to the bogus beacon. A benign node would insert the bogus beacon to $Queue_{check}$ based on the cooperative verifier. However, the bogus beacon will be rejected once the beacon signature verification fails. Therefore, any bogus beacon carrying a non-verified PC will be proven invalid upon the signature verification and will be dropped (or it will simply expire). As a result, none of the receiving nodes will perceive this (fictitious PC and sender). \square

5.3. Message Integrity and Authentication

The standard-compliant pseudonymously authenticated V2V communication entails message signing and verification with public/private key pairs that authenticated by the VPKI, thus, provides message integrity and authentication. Our scheme, while extending the beacon with additional fields, requires strict pseudonymous authentication on the messages, thus inheriting the message integrity and authentication. Message timestamps prevent message replays, providing entity authentication upon successful message verifications.

5.4. Non-repudiation and Accountability

Message content validation [31, 32] is out of the scope here. However, upon detection of adversarial messages violating our scheme specification, internal adversaries should be held accountable. We are especially concerned with internal adversaries attempt to validate the bogus messages (attached bogus signatures) using message verification facilitators.

Given Theorem 1, we know malicious nodes can only attempt validating bogus beacons, each attached a pair of valid PC and correct one-time symmetric key, exploiting malicious self-chained verifiers or cooperative verifiers. In order to take advantage of malicious cooperative verifiers, two or more malicious nodes need to collude. A receiving node, once verified a properly signed beacon piggybacking malicious cooperative verifiers, might accept the corresponding bogus beacons. Our scheme counters this misbehavior with probabilistic checking of signatures (Sec. 4.2.6) and by cross-checking verifiers (Sec. 4.2.7). *SIG* and *SELF* verifiers (assuming successful local verification of a said beacon) are the definitive verifiers, used to compare with *COOP* or *MAC* verifiers. When any conflict exists, corresponding *COOP* verifier providers are added to PRL and corresponding key chain owners are added to KRL respectively. In our scheme, nodes share not only positive *COOP* verifiers, but negative verifiers too. With positive verifiers alone, malicious nodes would be detected through signature probabilistic checking. By introducing negative *COOP* verifiers, verifiers with opposite validities from benign and malicious nodes can cause conflict, thus makes the receiver checks signature and this way evict the lying malicious node(s). The two countermeasures can detect and evict malicious nodes effectively and minimize the amount of falsely accepted

beacons, thwarting malicious node capability. We provide an extensive evaluation of misbehavior detection in Sec. 6 based on simulation.

Once any falsely validated bogus message is detected, corresponding malicious nodes should be evicted to prevent any further attempt on bogus message validation. This is established on non-repudiation of such misbehavior by corresponding malicious nodes, which is proven below. We first define Lemmas 1 and 2 to prove Theorem 2.

Lemma 1. *Let B_j be an authentic beacon that piggybacks a self-chained verifier that validates an earlier beacon B_i sent from the same node (i.e., the same PC) and $i < j$. If the hash function is secure, the successful signature verification on B_j proves authenticity of both B_j and B_i , and preserves non-repudiation of disseminating B_i .*

Proof. A successful signature verification on B_j is a proof of authentication of B_i (including its self-chained verifiers) by the sender. The sender cannot deny the dissemination of B_i , because the existence of a third beacon matches the hash value of B_i is practically impossible. Similarly, such authentication is transitive to earlier beacons that match self-chained verifiers in B_i . \square

Lemma 2. *If a beacon, B_i , corresponds to a self-chained verifier or a cooperative verifier in an already verified beacon based on the signature, the sender of the latter cannot deny the validation of B_i .*

Proof. A beacon could be accepted if a self-chained verifier or a cooperative verifier piggybacked on a verified beacon matches the hash value of the former beacon. In order to deny such a validation, there should exist another beacon, B'_i that matches the same verifier. More specifically, if B_i is bogus, there should exist an authentic beacon, which has the same hash value, in order to successfully deny the attempt to validate the bogus beacon, B_i . Such a hash collision is impossible with a secure hash function, especially considering the time limitation to find such a hash collision, due to ephemeral nature of safety messages. Therefore, the self-chained verifier or the cooperative verifier must have been computed based on B_i . \square

Theorem 2. *Any successful message validation can be always traced back to the validation by a legitimate node in a non-repudiable manner.*

Proof. It is straightforward that strict signature verification on event messages ensures non-repudiation. Conditional anonymity provided by the underlying VPKI holds nodes accountable for their messages. Similarly, any beacon verified based on the sender's signature provides non-repudiation and ensures the accountability of the sender.

We proved in Lemmas 1 and 2 that validation of a beacon based on a self-chained verifier or a cooperative verifier can also be traced back to a legitimate node in a non-repudiable manner. This is especially important for accountability (and follow-up eviction) of malicious nodes at-

tempting to validate bogus beacons based on self-chained verifiers or cooperative verifiers. \square

5.5. DoS-resilience

For safety beaconing, the challenge is two-fold: efficient neighboring node discovery and keeping track of the discovered nodes; both achieved in terms of beacon reception and verification. We provide an analysis of the DoS-resilience of our scheme before a thorough simulation-based quantitative evaluation in Sec. 6.

Node discovery is facilitated by: 1) continuously queued beacon verification (Sec. 4.2.3), and 2) cooperative verification (Sec. 4.2.6). As in a traditional scheme, each node continuously verifies beacons from the queue for node discovery. However, merely verifying beacons from the queue does not guarantee timely discovery of new nodes, because the majority of beacons in the queue could be bogus beacons, when bogus beacon rate overwhelms benign beacon rate. For more efficient and targeted node discovery, every time a beacon passed any of the three validations, i.e., signature verification, self-chained verification and MAC verification, the positive cooperative verifiers can be used to find non-discovered nodes' beacons. The beacons found with the positive cooperative verifiers are highly probable to be the benign ones. The efficient malicious node detection sets a basis for low (non-detected) malicious node ratio. As a result, extensive cooperative verifiers from honest benign nodes greatly facilitate node discovery. Even though the cooperative verifiers could be malicious, due to mandatory signature verifications on the found beacons, they will not be accepted and the resultant signature verifications detect misbehaving nodes.

After successful node discovery, one-time keys (i.e., hash chain elements) can be used to keep track of subsequent potentially valid beacons (Sec. 4.2.2). Second preimage resistance of hash function ensures that only the key chain owner knows and can disclose the correct one-time keys. Therefore, even if multiple (single-hop) beacons, piggybacking the correct {one-time key, PC} pairs, are received during each time interval, only the first of them needs to be queued; the rest can be dropped, because they are sent after overhearing the original beacon. In case the original beacon was not received due to packet loss, a received masqueraded beacon may be considered as the potentially valid one. However, the effect of such bogus beacons is limited, because at most one bogus beacon is queued for each time slot for each PC: a significant improvement over the traditional scheme, with which all bogus beacon signatures need to be verified (or until the valid one is verified for that time slot) before they can be dropped. Moreover, negative cooperative verifiers and self-chained verifications can help eliminate such masqueraded beacons.

We explicitly assign CPU time ratios for discovered/non-discovered nodes' beacon verification: $Ratio_D$ of time for discovered nodes' beacons and $Ratio_{ND}$ for node

discovery, while $Ratio_D + Ratio_{ND} = 1$. The checks on key chain help to filter out bogus beacons attached the discovered PCs, and the explicitly assigned CPU time ratio enable receivers to verify their discovered nodes' beacons, even if attackers flood and overwhelm receiver queues with bogus beacons attached random signatures and PCs (thus non-discovered nodes' beacons, from the perspective of receivers).

Event message verification is established on beacon verification. The DoS-resilient beacon verification ensure timely caching of event message facilitator before the corresponding event message dissemination. Nodes efficiently capture and queue benign event messages matching the cached facilitators and drop bogus event messages.

6. Quantitative Evaluation

We present detailed simulation results with realistic mobility and communication models, and processing delays. We show that our scheme provides timely message validation with low beacon expiration ratio and low node discovery delay even under loaded networks or DoS attacks than the baseline scheme (the prior approaches that strictly verify every message signature).⁴ Considering internal malicious nodes that abuse features of the scheme, notably the cooperative verification, we show our scheme is resilient to false validations and can effectively detect malicious nodes. With an evaluation of event-driven message dissemination, we show our scheme can facilitate reception and validation of various message types.

6.1. Simulation Setting

We use OMNeT++ [58] for a packet-level simulation with IEEE 802.11p module for V2V communication provided by Veins [59]. We consider a maximum communication range of 200m with a default 6 Mbps bit rate. SUMO provides a microscopic mobility simulation module that serves Veins with mobility traces. We use the SUMO [60] TAPASCologne scenario [61] to simulate vehicle mobility, with a penetration ratio of 50%, i.e., 50% of nodes disseminate safety messages and participate in our scheme. We consider a base value for processing power and thus τ , that reflects recent literature [11, 12, 13, 14, 15, 16]. Then, we increase processing power by an order of magnitude, to the minimum τ latest OBUs claim to support.

We simulate with vehicle traces from 12:00 pm, and vehicles start safety beaconing from 12:30 pm and continue for 2 minutes before simulations conclude: the first 30 minutes are used for filling up the network with vehicles. Fig. 7 shows a heat map of node density at 12:30 pm. We use a $2\text{ km} \times 2\text{ km}$ area ($Region_{sim}$) with the densest

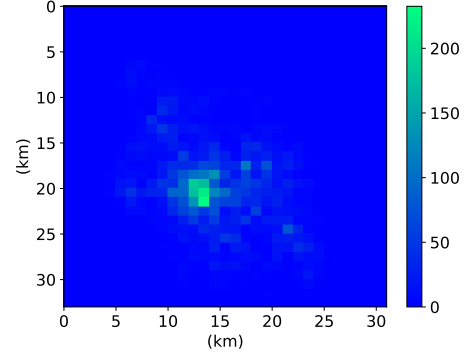


Figure 7: Node density in TAPASCologne scenario at 12:30 pm.

Bitrate	6 , 27 Mbps
Pr_{check}	0, 0.2 , 0.5, 0.8
α	0, 1, 2, 3 , 4
k	3
T_{blife}	1 sec
τ	0.4, 2, 4 msec
γ	10 Hz
β_1	1 , 2
β_2	2, 3
γ_{DoS}	250 , 500, 1000 Hz
$Ratio_{adv}$	0.1, 0.3, 0.5
$\{Ratio_S, Ratio_V\}$	{0.5, 0.5}
$\{Ratio_D, Ratio_{ND}\}$	{0.5, 0.5}

Table 3: System Parameters (**Bold** for Default Setting)

node population of the city. Nodes start safety beaconing within the central $1\text{ km} \times 1\text{ km}$ area ($Region_{beacon}$) of the simulated area, and results are collected from nodes within the central $0.5\text{ km} \times 0.5\text{ km}$ ($Region_{result}$) area. Although vehicles, in a real-world scenario, keep beaconing without any region restriction, we choose such an area, although much smaller than the full city size, in order to keep the simulations manageable, while being able to capture performance in the most significant area. Internal adversaries start their attacks when they enter the central $0.5\text{ km} \times 0.5\text{ km}$ area ($Region_{attack}$), affecting more nodes in the denser area.⁵ 16 bogus beacon generators (i.e., DoS attackers), which form a 4×4 matrix, are placed at the center $0.6\text{ km} \times 0.6\text{ km}$ area. Nodes start disseminating safety beacons at 12:30 pm. This captures a situation that vehicles change their PCs simultaneously, as per the privacy-preserving PC changing policy [9], which is the most challenging scenario in terms of node discovery. As a result, some nodes will be attacked by more than one bogus beacon generators. Table 3 shows simulation parameters. We consider an average τ that includes both signature verification and relevant scheme operations (e.g., hash computation and string comparison). The former could be generally one or two order(s) of magnitude more computationally expensive than the latter one. We consider a beacon size to be 300 bytes, and each hash digest to be 20 bytes (i.e., SHA-1 hash size). Results are averaged over

⁴We do not compare with our earlier work [33], because, as explained earlier, the adopted symmetric key based authentication does not provide non-repudiation and accountability.

⁵If the adversaries start attacking in $Region_{beacon}$, they can be detected more easily with signature verifications, because less nodes exist at the border of $Region_{beacon}$, thus less beacon arrivals.

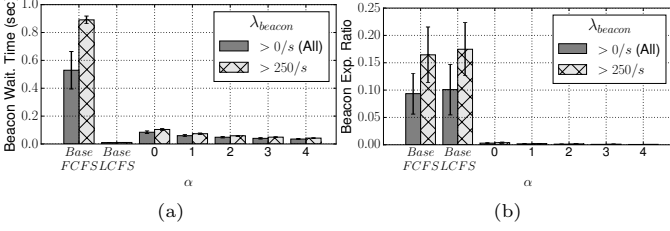


Figure 8: Beacon validation metrics as a function of α when in benign network.

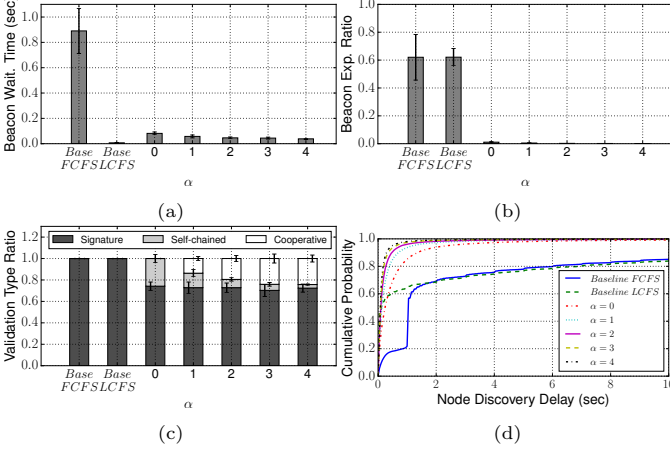


Figure 9: Beacon validation metrics as a function of α when under the DoS attack.

five randomly seeded runs for each simulation setting.

6.2. Resilience to DoS Attacks

We evaluate different beacon validation metrics: **waiting time**, **expiration ratio**, **validation type ratio** and **node discovery delay**. *Waiting time* is defined as the time the beacon stays in queue until its verification. *Expiration ratio* is the ratio of beacons that expired before any validation. *Validation type ratio* shows percentage of beacon validation based on each method. Another important criterion for availability of the scheme is short discovery delay of new neighboring nodes. The continuous awareness of a neighboring node is only possible if beacons from the node can be continuously verified and accepted. With the ability to efficiently track potentially valid subsequent beacons based on hash chains, the challenge lies in verification of a beacon from the new neighboring node so that the corresponding hash chain can be trusted. We evaluate this criteria with *node discovery delay*: delay between the reception of the first valid beacon from a new neighboring node and the verification of at least one (possibly later) beacon from that same node.

We evaluate the baseline scheme and our scheme in benign and DoS scenarios. The baseline scheme, with none of the DoS-resilient features, nodes verify beacons independently with pure (i.e., not semi-randomized) FCFS or LCFS queue processing. Figs. 8 and 9 show the metrics for the baseline scheme (referred as *Base* in the figures)

and our scheme. For the benign scenario (Fig. 8), apart from average beacon waiting time and expiration ratio for all nodes ($\lambda_{beacon} > 0/s$), we evaluate the metrics for nodes that λ_{beacon} is higher than $1/\tau = 250/s$ (i.e., higher message arrival rate than a queue can sustain). This essentially captures the performance of our scheme in benign networks when nodes are overloaded with benign beacons, thus evaluates the scalability provided by our scheme. For the benign scenarios, we see the FCFS baseline scheme exhibits high beacon waiting time while the LCFS baseline scheme provides low waiting time: the former approach verifies the earliest beacon in the queue, while the latter approach always verifies the latest beacon. Due to the consistent τ , the amount of verifiable beacons per time unit are roughly the same for the both approaches. Therefore, the two approaches exhibit similar expiration ratios. As expected, the baseline scheme exhibits high beacon expiration ratios: Fig. 8b shows around 10% and 16% of beacons from benign nodes have to be dropped due to expiration, for $\lambda_{beacon} > 0/s$ and $\lambda_{beacon} > 250/s$ respectively. When our scheme is adopted, expiration ratios significantly decreases, while maintaining reasonably low beacon waiting times (see Figs. 8a and 8b with $\alpha \geq 0$). Higher α introduces higher communication overhead, but nodes become more scalable and resilient to both DoS attacks and malicious nodes, while such improvement becomes moderate as α increases.

For the DoS scenarios (Fig. 9), the LCFS baseline scheme still provides very low waiting time and the FCFS baseline scheme provides an average waiting time close to 1s, i.e., the beacon lifetime, because the queues are always loaded with high-rate beacons and the verified beacons almost reached the end of their lifetimes. The expiration ratios increases to around 60% due to computation power wasted on bogus beacon verification. Thanks to continuous (potentially) valid beacon tracking leveraging hash chains and explicit time allocation for discovered and non-discovered nodes, our scheme guarantees low beacon expiration ratios even under the DoS attacks: roughly same as those for the benign scenario (see Figs. 8b and 9b with $\alpha \geq 0$).

Fig. 9c shows ratios of verified beacons based on each validation methods. With higher α , higher ratios of beacons are validated based on *COOP* verifiers, while lower ratios of beacons need to wait for self-chained verifications. This results in lower average waiting time as α increases (Figs. 8a and 9a): *COOP* verifiers can validate fresher beacons while *SELF* verifiers validate beacons that are received several beacon intervals earlier.

Fig. 9d shows Cumulative Distribution Functions (CDFs) for node discovery delay. For the baseline scheme, each node independently discovers neighbors and beacon arrival rate is higher than beacon verification rate for some nodes in the simulations, which result in higher node discovery delays. There is a significant improvement even with $\alpha = 0$ thanks to expedited queue processing based on self-chained verifications. There is

slight improvement with $\alpha > 0$ than with $\alpha = 0$, while the lines almost overlap with higher positive α values. Improvements with positive α values (i.e., $\alpha > 0$) are still observable, while the lines almost overlap with higher positive α values. When $\alpha = 0$, around 77% of nodes can be discovered within 0.5s, and increases to around 90% when $\alpha = 1$. It improves further, but only moderately, with higher α values, e.g., around 96% and 97% of nodes are discovered within 0.5s when $\alpha = 3$ and 4 respectively.

We show beacon validation metrics as a function of Pr_{check} (Fig. 10). In general, our scheme still maintains reasonably low waiting time, expiration ratio and node discovery delay (Figs. 10a, 10b and 10d). In the evaluation, we categorize beacons that probabilistically checked into cooperatively verified. With higher Pr_{check} , less beacons can be verified based on signatures (Fig. 10c), because more cooperatively validated beacons need to be checked, which results in more beacons validated based on the alternative methods.

We continue evaluation with lower message verification delay and higher bogus beacon rates (Fig. 11). With a lower $\tau = 2 \text{ msec}$ (Figs. 11a, 11b, 11d and 11e), beacon expiration ratios are still high for the baseline scheme and significantly lower with our scheme, and node discovery delays improve with higher α values. For example, when $\tau = 2 \text{ msec}$ and $\gamma_{DoS} = 500 \text{ Hz}$ (Fig. 11e), only around 59% of nodes can be discovered within 0.5 sec with $\alpha = 0$, but the value significantly increases to 88% with $\alpha = 3$. Moreover, the CDF converges towards 100% much faster than the baseline scheme. Next, we evaluate our scheme considering high-end vehicular OBUs. We decrease τ by one order of magnitude from the default value, thus $\tau = 0.4 \text{ msec}$, and set the bitrate of each node to the maximum 27 Mbps [50, 18]. With this setting, we see generally improved results even under higher bogus beacon rates, $\gamma_{DoS} = 1000 \text{ Hz}$, due to one order of magnitude shorter beacon processing delays. With the baseline scheme, around 95% of nodes can be discovered within 0.5s, while the values are 91% and 97% for our scheme with $\alpha = 0$ and 3 respectively. Our scheme slightly outperforms the baseline scheme only with a positive α value (e.g., 3), but our scheme ensures much lower expiration ratios (almost 0%), compared to around 10% and 20% of expiration ratios for the general and loaded cases respectively. Although the expiration ratios for the baseline scheme are less significant than those in Figs. 11a and 11b, in highly congested networks that already experience high packet loss rates, even the seemingly relatively low expiration ratio, e.g., 20%, could be critical for safety application functionality. Similarly, we see reasonably low waiting time values for the three scenarios from the simulation results (not shown due to space limitation). In general, our scheme provides less significant improvements with lower processing delays, due to the relatively moderate bandwidth increase (i.e., 4.5 times higher) compared to the improved (10 times higher) computational power. However, we can expect better performance with our scheme

when advanced communication technology is used, e.g., 5G with higher bandwidth and better congestion control, with which much higher message rates are expected.

6.3. Resilience to Malicious Nodes

We continue with evaluation of resilience to internal malicious nodes, i.e., nodes that falsely validate bogus beacons. The only way to make a benign node (i.e., victim) accepting bogus beacons, is through cooperative verifiers. Moreover, this is only possible after the PC on the bogus beacons has been discovered already by the victim and the subsequent bogus beacons are piggybacked with correct one-time keys and MACs. In the simulations, we assume $Ratio_{adv}$ of nodes are malicious, and $Ratio_S$ of malicious nodes are malicious senders that disseminate bogus beacons with correct one-time keys and $Ratio_V$ of malicious nodes are malicious validators that overhear the malicious bogus beacons and validate the bogus beacons through their own false positive cooperative verifiers. Our scheme has two features for malicious node detection: probabilistic checking and verifier cross-checking. In this evaluation, we record, for each benign and malicious nodes pair, numbers of bogus beacons the malicious (validator) node made the benign node accept, through cooperative verifiers, before the malicious node is detected by the benign node. We term accepted bogus beacon as affected beacon. A benign node could be attacked by multiple malicious validators and a malicious validator could successfully attack multiple benign nodes, but we consider each benign and malicious nodes pair as one instance and count the number of affected beacon for each pair.

Figs. 12a to 12c show histograms of numbers of affected beacon under different $Ratio_{adv}$ values without any detection features. In this case, detection is possible only when a node verified signature of a cooperatively validated bogus beacon, for node discovery. A malicious validator can make a benign node accept more than 100 (more than 300, for the worst case when $Ratio_{adv} = 0.3$ and 0.5) bogus beacons. With such amount of bogus beacons accepted by a benign node, vehicle operations and even vehicle safety can be heavily affected. Next, we evaluate scenarios with probabilistic checking only. With a positive $Pr_{check} = 0.2$, (Fig. 12d), we see the amount of affected beacons significantly decrease. Although this shows the importance of probabilistic checking for malicious node detection, a benign node could still falsely accept around 20 bogus beacons validated by a malicious node. We can expect the number should decrease with higher Pr_{check} value, but it would incur higher computation overhead, conflicting with the goal of our scheme. We evaluate the performance with the adoption of both detection features (Fig. 12e). The results significantly improves due to higher chances for misbehavior detection based on cross-checking (see Sec. 5).

We see from the results that only few bogus beacons are accepted while an overwhelming majority of benign beacons can still provide redundancy to tolerate such bogus beacon consumption. The only way to disrupt the system

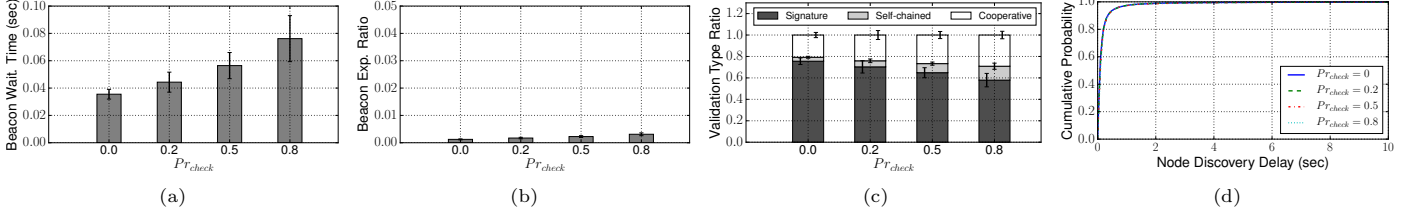


Figure 10: Beacon validation metrics as a function of Pr_{check} under the DoS attack. Default: $\alpha = 3$.

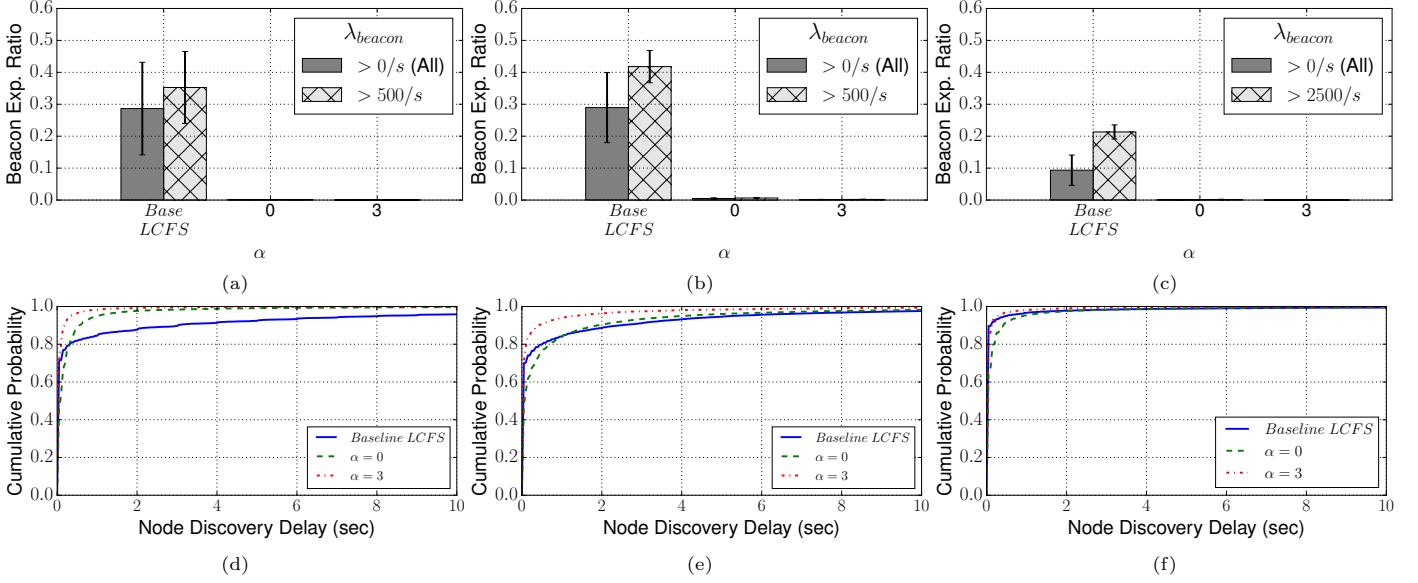


Figure 11: Beacon expiration ratio and node discovery delay as a function of α when under a clogging DoS attack. (a), (d) Bitrate = 6 Mbps, $\tau = 2ms$ and $\gamma_{DoS} = 250 Hz$. (b), (e) Bitrate = 6 Mbps, $\tau = 2ms$ and $\gamma_{DoS} = 500 Hz$. (c), (f) Bitrate = 27 Mbps, $\tau = 0.4ms$ and $\gamma_{DoS} = 1000 Hz$.

is still providing authenticated false data. We see from the results that our scheme does not degrade security, while significantly improves system performance when nodes are loaded with high-rate beacons.

6.4. Resilient Event-driven Message Dissemination

Our scheme facilitates event-driven messages dissemination and reception leveraging safety beacons. We evaluate with two example scenarios: misbehavior evidence and DENM. In the first example, every time a new malicious PC is detected with a solid evidence (i.e., a bogus beacon and a malicious validating beacon, both properly signed; detected either locally or based on a received evidence), then the evidence is disseminated to neighbors. In the second example, nodes disseminate more general event-driven messages, i.e., DENMs, at a given rate along with misbehavior evidences, and we evaluate with two event validation metrics: event waiting time and event acceptance ratio. In this evaluation, we consider an average DENM generation interval of 30s that follows exponential distribution by each node. The first of β_2 repetitions is disseminated 20ms after the one-time key disclosure, and the subsequent $\beta_2 - 1$ repetitions are disseminated with an interval of 20ms from the previously one.

We start with the first example: misbehavior evidence dissemination. In our simulation, misbehavior evidence facilitator occupies one of α positions, so that $\alpha - 1$ cooperative verifiers are piggybacked when a misbehavior evidence needs to be disseminated. Moreover, if a misbehavior evidence dissemination is ongoing, another new misbehavior evidence will be queued until the $\beta_1 * \beta_2$ repetitions conclude. We are especially concerned with the effectiveness of this new component on malicious node detection. Fig. 13 shows affected beacon amounts when both probabilistic checking and cross-checking mechanisms are applied. We see the results improve again (over Fig. 12e) thanks to proactive malicious node eviction based on the misbehavior evidences. With more realistic $Ratio_{adv}$ values (0.1 and 0.3 in Figs. 13a and 13b), our scheme effectively thwarts the malicious nodes and significantly minimizes the vulnerability window introduced by cooperative verification. Our scheme still guarantees timely verification and acceptance of benign beacons, while the above detection features protect benign nodes from being attacked by the malicious nodes and DoS attacks.

For the second example, we are concerned with event waiting time and event acceptance ratio. In this scenario, each beacon could carry at most one misbehavior evidence

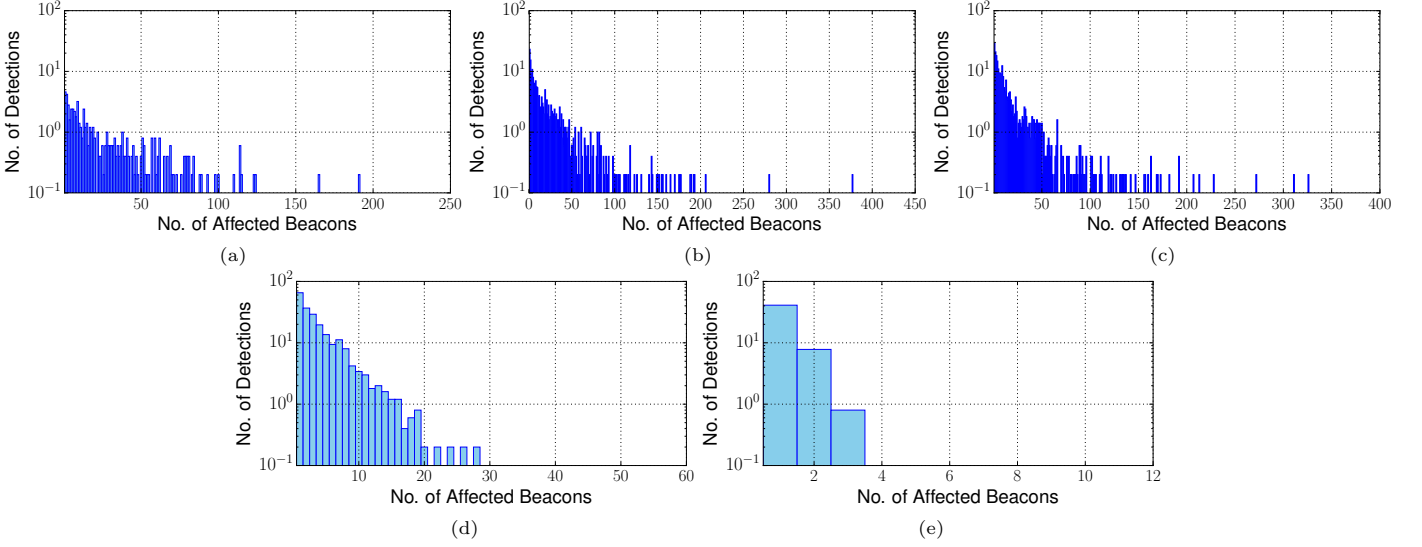


Figure 12: Histogram of numbers of affected beacons when under the DoS attack and in the presence of malicious nodes. (a),(b),(c) No protection with $Ratio_{adv} = 0.1, 0.3, 0.5$. (d) Probabilistic checking only. (e) Both probabilistic checking and cross-checking. (Default: $Ratio_{adv} = 0.5, Pr_{check} = 0.2$)

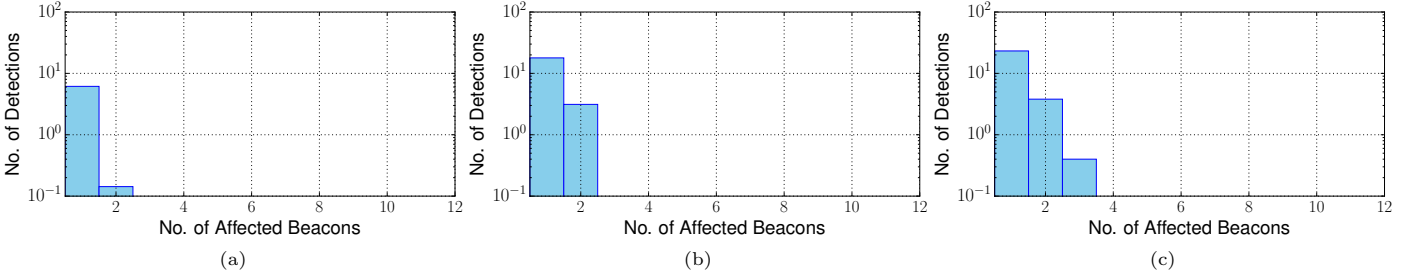


Figure 13: Histogram of numbers of affected beacons with full protection and misbehavior evidence dissemination under the DoS attack and the presence of malicious nodes. $Ratio_{adv} = 0.1, 0.3, 0.5$.

facilitator and at most one DENM facilitator; both share α positions with cooperative verifiers. *Event waiting time* denotes delay between event generation at sender and successful event verification at receiver, and *event acceptance ratio* denotes ratio of received distinct event messages that match locally cached event facilitators out of total received distinct event messages. A received valid event message, without a local cached matching facilitator, will be dropped immediately, because it is indistinguishable from bogus ones. Here, we consider a type of DENM disseminated for collision avoidance [62]. Once a DENM is triggered, it is repeated three times with an interval of 100 ms, thus $\beta_1 = 1$ and $\beta_2 = 3$ by default. We consider a DENM lifetime of 2s, according to the standard [62]. We assume DENMs are assigned higher priority than safety beacons. Whenever DENMs are received, they are verified before verifying queued safety beacons.

We evaluate the two metrics for the baseline scheme and our scheme with different $\{\beta_1, \beta_2\}$ combinations (Figs. 14a and 14b). For the baseline scheme, event acceptance ratio is calculated with $(1 - expiration\ ratio)$. In the benign network, the baseline scheme provides a low waiting time with almost 100% DENM acceptance, due

to low DENM arrival rate and higher priority given to the DENMs. We consider two DoS attack scenarios: the attackers flood with both beacons and DENMs, or DENMs only. In the former scenario, each attacker broadcasts beacons and DENMs both at 125Hz respectively (thus, 250Hz in total). In the latter scenario, each attacker broadcasts only DENMs at 250Hz. As expected, event waiting time for the former scenario is around 0.6s, lower than around 1.6s for the latter scenario. At the same time, the former scenario exhibits an event acceptance ratio of around 90%, compared to around 50% for the latter scenario. With our scheme, average event waiting times for all combinations are around 0.17s, and around 80% of received event messages are kept and verified. Moreover, it provides reasonably low beacon waiting time and expiration ratio (Figs. 14c and 14d): a significant improvement from the baseline scheme. Beacon waiting time for the DoS-with-DENM-only scenario shows a large confidence interval, due to insufficient sample size collected for beacon waiting time, given high beacon expiration ratio (Fig. 14d). For example, in one of the seeded simulation runs, only one sample for beacon waiting time was recorded with very low value, while all

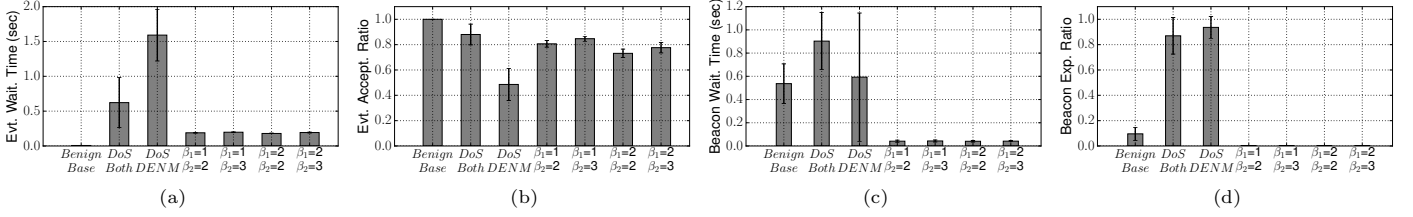


Figure 14: Event and beacon validation metrics with full protection under the DoS attack.

the rest are expired.

6.5. Discussion

Our scheme is resilient to DoS attacks and reliable in an highly-loaded benign network. However, in lightly-loaded benign networks (see Figs. 14a and 14b for example), the baseline scheme performs better: low waiting time and low expiration ratio, rendering the extra communication overhead and the semi-randomized LCFS queue unnecessary. To achieve optimal performance in all conditions, our future work will include dynamic switching from the baseline to our scheme, based on real-time network conditions.

We provide a preliminary evaluation for one type of DENM, covering two important metrics - message waiting time and acceptance ratio. However, considering the existence of various DENMs and event-driven messages, an extended evaluation for the applicability of our scheme is required, especially addressing functional requirements of various safety applications.

Our simulations are built on standard-compliant IEEE 802.11p for V2V communication. As an evolution of IEEE 802.11p, the proposed IEEE 802.11bd provides a potential maximum rate of 87.75 Mbps [63], much higher than the maximum data rate of 27 Mbps for IEEE 802.11p. Moreover, the current 5G and upcoming 6G for C-Vehicle-to-Everything (V2X) [64] could also coexist with IEEE 802.11p/bd. Continuously developing communication standards together with relatively constant cryptographic delays (for sustainable security, as discussed in Sec. 2) would aggravate clogging DoS attacks. The higher the bandwidth provided by upcoming advanced communication technologies (thus, the higher maximum message rate an attacker has at her disposal), the more relevant a scheme as this one is: high bogus message rates can overwhelm even high-end CPU provisioning.

7. Conclusions

We propose a scalable, DoS-resilient safety message verification scheme, provides vehicles with efficient message verification and protection against DoS attacks, orthogonal to the underlying physical layer communication technology. Simulation results show average beacon verification latency of 50ms with less than 1% expiration ratio even under DoS attacks. Our scheme minimizes vulnerability to misuse of cooperative verification, thanks to

probabilistic checking and verifier cross-checking. It also facilitates reception and verification of different types of V2V messages leveraging safety beacon format extension. With a realistic DENM dissemination model, simulation results show 80% of message acceptance ratio with an average latency less than 200ms, compared to 50% - 100% message expiration ratio with the baseline scheme.

Acknowledgement

This work was supported in parts by the Swedish Research Council and the Knut och Alice Wallenberg Foundation.

References

- [1] IEEE Std 1609.12, Ieee standard for wireless access in vehicular environments (wave) - identifier allocations (2016).
- [2] ETSI EN 302 637-2, ITS; Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service (2019).
- [3] ETSI EN 302 637-3, ITS; Vehicular Communications; Basic Set of Applications; Part 3: Specifications of Decentralized Environmental Notification Basic Service (2019).
- [4] P. Papadimitratos, L. Buttyan, T. Holczer, E. Schoch, J. Freudiger, M. Raya, Z. Ma, F. Kargl, A. Kung, J.-P. Hubaux, Secure vehicular communication systems: Design and architecture, IEEE Communications Magazine 46 (11).
- [5] F. Kargl, P. Papadimitratos, L. Buttyan, M. Muter, E. Schoch, B. Wiedersheim, T.-V. Thong, G. Calandriello, A. Held, A. Kung, J.-P. Hubaux, Secure vehicular communication systems: Implementation, performance, and research challenges, IEEE Communications Magazine.
- [6] P. Papadimitratos, A. La Fortelle, K. Evenssen, R. Brignolo, S. Cosenza, Vehicular communication systems: Enabling technologies, applications, and future outlook on intelligent transportation, IEEE Communications Magazine 47 (11).
- [7] M. Khodaei, H. Jin, P. Papadimitratos, Towards deploying a scalable & robust vehicular identity and credential management infrastructure, in: IEEE VNC, Paderborn, Germany, 2014.
- [8] M. Khodaei, P. Papadimitratos, The key to intelligent transportation: Identity and credential management in vehicular communication systems, IEEE VT Magazine 10 (4).
- [9] M. Khodaei, H. Jin, P. Papadimitratos, Secmace: Scalable and robust identity and credential management infrastructure in vehicular communication systems, IEEE Transactions on ITS 19 (5) (2018) 1430–1444.
- [10] I. S. 1609.2, Ieee standard for wireless access in vehicular environments-security services for applications and management messages, IEEE Std 1609.2-2016.
- [11] G. Calandriello, P. Papadimitratos, J.-P. Hubaux, A. Lioy, On the performance of secure vehicular communication systems, IEEE TDSC 8 (6) (2011) 898–912.

- [12] J. Petit, Z. Mammeri, Authentication and consensus overhead in vehicular ad hoc networks, *Telecommunication systems* 52 (4) (2013) 2699–2712.
- [13] M. A. R. Baee, L. Simpson, E. Foo, J. Pieprzyk, Broadcast authentication in latency-critical applications: On the efficiency of IEEE 1609.2, *IEEE TVT* 68 (12) (2019) 11577–11587.
- [14] J. Pan, J. Cui, L. Wei, Y. Xu, H. Zhong, Secure data sharing scheme for vanets based on edge computing, *EURASIP Journal on Wireless Communications and Networking* 2019 (1).
- [15] PRESERVE, Deliverable 3.2 for trial 2 results (Jul. 2015).
- [16] CAMP VSC5, Security credential management system proof-of-concept implementation - requirements and specifications supporting scms software release 1.1 (May 2016).
- [17] M. A. Mehrabi, A. Jolfaei, Efficient cryptographic hardware for safety message verification in internet of connected vehicles, *ACM Transactions on Internet Technology* 22 (4) (2022) 1–16.
- [18] M. Sepulcre, J. Gozalvez, B. Coll-Perales, Why 6 mbps is not (always) the optimum data rate for beaconing in vehicular networks, *IEEE Transactions on Mobile Computing* 16 (12).
- [19] CAR 2 CAR Communication Consortium, Survey on its-g5 cam statistics (Dec. 2018).
- [20] G. Calandriello, P. Papadimitratos, J.-P. Hubaux, A. Lioy, Efficient and robust pseudonymous authentication in vanet, in: *ACM VANET*, New York, USA, 2007.
- [21] E. Schoch, F. Kargl, On the efficiency of secure beaconing in vanets, in: *ACM WiSec*, Hoboken, NJ, 2010.
- [22] M. Feiri, J. Petit, F. Kargl, Formal model of certificate omission schemes in vanet, in: *IEEE VNC*, Paderborn, Germany, 2014.
- [23] H. Jin, P. Papadimitratos, Proactive certificate validation for VANETs, in: *IEEE VNC*, Columbus, OH, 2016.
- [24] A. Studer, F. Bai, B. Bellur, A. Perrig, Flexible, extensible, and efficient vanet authentication, *Journal of Communications and Networks* 11 (6) (2009) 574–588.
- [25] H.-C. Hsiao, A. Studer, C. Chen, A. Perrig, F. Bai, B. Bellur, A. Iyer, Flooding-resilient broadcast authentication for vanets, in: *ACM MobiCom*, Las Vegas, NV, 2011.
- [26] C. Lyu, D. Gu, Y. Zeng, P. Mohapatra, PBA: Prediction-Based Authentication for Vehicle-to-Vehicle Communications, *IEEE TDSC* 13 (1) (2016) 71–83.
- [27] P. Papadimitratos, V. Gligor, J.-P. Hubaux, Securing vehicular communications-assumptions, requirements, and principles, in: *ESCAR*, Berlin, Germany, 2006.
- [28] ETSI TR 102 893, Intelligent transport systems (its); security; threat, vulnerability and risk analysis (tvra) (2017).
- [29] X. Lin, X. Li, Achieving efficient cooperative message authentication in vehicular ad hoc networks, *IEEE Transactions on Vehicular Technology* 62 (7) (2013) 3339–3348.
- [30] H. Jin, P. Papadimitratos, Scaling VANET security through cooperative message verification, in: *IEEE VNC*, Japan, 2015.
- [31] 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.
- [32] S. Gisdakis, T. Giannetos, P. Papadimitratos, SHIELD: A Data Verification Framework for Participatory Sensing Systems, in: *ACM WiSec*, New York, NY, 2015.
- [33] H. Jin, P. Papadimitratos, Dos-resilient cooperative beacon verification for vehicular communication systems, *Ad Hoc Networks* 90 (2019) 101775.
- [34] Y.-c. Hu, K. P. Laberteaux, Strong vanet security on a budget, in: *ESCAR*, Berlin, Germany, 2006.
- [35] S. Dongre, H. Rahbari, Message sieving to mitigate smart grid-lock attacks in v2v, in: *ACM WiSec*, 2021.
- [36] C. Sun, J. Liu, X. Xu, J. Ma, A privacy-preserving mutual authentication resisting dos attacks in vanets, *IEEE Access* 5 (2017) 24012–24022.
- [37] 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.
- [38] A. K. Lenstra, E. R. Verheul, Selecting cryptographic key sizes, *Journal of cryptology* 14 (2001) 255–293.
- [39] H. Jin, Z. Zhou, P. Papadimitratos, Future-proofing secure v2v communication against clogging dos attacks, in: *ARES*, 2024.
- [40] D. Hankerson, A. J. Menezes, S. Vanstone, Guide to elliptic curve cryptography, Springer Science & Business Media, 2006.
- [41] 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*, San Francisco, CA, 2000.
- [42] Q. Dong, D. Liu, P. Ning, Providing dos resistance for signature-based broadcast authentication in sensor networks, *ACM Transactions on Embedded Computing Systems* 12 (3) (2013) 73.
- [43] H. Givehchian, N. Bhaskar, A. Redding, H. Zhao, A. Schulman, D. Bharadia, Practical obfuscation of BLE physical-layer fingerprints on mobile devices, in: *IEEE Symposium on Security and Privacy*, San Francisco, CA, 2024.
- [44] L. Von Ahn, B. Maurer, C. McMillen, D. Abraham, M. Blum, recaptcha: Human-based character recognition via web security measures, *Science* 321 (5895) (2008) 1465–1468.
- [45] G. Twardokus, H. Rahbari, Vehicle-to-nothing? securing c-v2x against protocol-aware dos attacks, in: *IEEE INFOCOM*, 2022.
- [46] CAMP VSC2, Vehicle safety communications - applications (vsc-a) final report (Sep. 2011).
- [47] J. B. Kenney, Dedicated short-range communications (dsrc) standards in the united states, *Proceedings of the IEEE* 99 (7) (2011) 1162–1182.
- [48] J. Petit, F. Schaub, M. Feiri, F. Kargl, Pseudonym schemes in vehicular networks: A survey, *IEEE communications surveys & tutorials* 17 (1) (2015) 228–255.
- [49] A. Festag, P. Papadimitratos, T. Tielert, Design and Performance of Secure Geocast for Vehicular Communication, *IEEE TVT* 59 (5) (2010) 2456–2471.
- [50] F. A. Teixeira, V. F. e Silva, J. L. Leoni, D. F. Macedo, J. M. Nogueira, Vehicular networks using the IEEE 802.11 p standard: An experimental analysis, *Vehicular Communications* 1 (2) (2014) 91–96.
- [51] C.-Y. Chang, H.-C. Yen, D.-J. Deng, V2V QoS guaranteed channel access in IEEE 802.11 p VANETs, *IEEE TDSC* 13 (1).
- [52] M. Fiore, C. E. Casetti, C. F. Chiasserini, P. Papadimitratos, Discovery and Verification of Neighbor Positions in Mobile Ad Hoc Networks, *IEEE TMC* 12 (2) (2013) 289–303.
- [53] M. Poturalski, P. Papadimitratos, J. P. Hubaux, Formal Analysis of Secure Neighbor Discovery in Wireless Networks, *IEEE TDSC* 10 (6) (2013) 355–367.
- [54] ETSI EN 302 637-2, Intelligent transport systems; vehicular communications; basic set of applications; part 2: Specification of cooperative awareness basic service (Nov. 2014).
- [55] M. Stevens, E. Bursztein, P. Karpman, A. Albertini, Y. Markov, The first collision for full sha-1, in: *CRYPTO*, Santa Barbara, CA, 2017, pp. 570–596.
- [56] M. Raya, P. Papadimitratos, I. Aad, D. Jungels, J.-P. Hubaux, Eviction of misbehaving and faulty nodes in vehicular networks, *IEEE JSAC* 25 (8) (2007) 1557–1568.
- [57] H.-H. Nguyen, H.-Y. Jeong, Mobility-adaptive beacon broadcast for vehicular cooperative safety-critical applications, *IEEE Transactions on ITS* 19 (6) (2018) 1996–2010.
- [58] A. Varga, R. Hornig, An overview of the omnet++ simulation environment, in: *SIMUTOOLS*, Marseille, France, 2008.
- [59] C. Sommer, R. German, F. Dressler, Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis, *IEEE TMC* 10 (1) (2011) 3–15.
- [60] 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.
- [61] S. Uppoor, O. Trullols-Cruces, M. Fiore, J. M. Barcelo-Ordinas, Generation and analysis of a large-scale urban vehicular mobility dataset, *IEEE TMC* 13 (5) (2013) 1061–1075.
- [62] C2C-CC, Triggering conditions and data quality exchange of ircs (Aug. 2018).
- [63] A. Triwinarko, I. Dayoub, S. Cherkaoui, Phy layer enhancements for next generation v2x communication, *Vehicular Communications* 32 (2021) 100385.

- [64] C.-X. Wang, J. Huang, H. Wang, X. Gao, X. You, Y. Hao, 6g wireless channel measurements and models: Trends and challenges, *IEEE Vehicular Technology Magazine* 15 (4).