Faster verification of V2X BSM messages via Message Chaining

Eduardo Lopes Cominetti[1], Marcos Vinicius M. Silva[1], Marcos A. Simplicio Jr.[1],
Harsh Kupwade Patil[2], Jefferson E. Ricardini[2]

***Abstract*—Vehicular-to-Everything (V2X) communications enable vehicles to exchange messages with other entities, including nearby vehicles and pedestrians. V2X is, thus, essential for establishing an Intelligent Transportation System (ITS), where vehicles use information from their surroundings to reduce traffic congestion and improve safety. To avoid abuse, V2X messages should be digitally signed using valid digital certificates. Messages sent by unauthorized entities can then be discarded, while misbehavior can lead to the revocation of the corresponding certificates. One challenge in this scenario is that messages must be verified shortly after arrival (e.g., within centiseconds), whereas vehicles may receive thousands of them per second. To handle this issue, some solutions propose prioritization or delayed-verification mechanisms, while others involve signature schemes that support batch verification. In this manuscript, we discuss two mechanisms that complement such proposals, enabling the authentication of a sequence of messages from the same source with one single signature verification. Our analysis shows that the technique can reduce the number of verified signatures by around 90% for reliable communication channels, and by more than 65% for a maximum packet loss rate of 20%.**

***Index Terms*—Vehicular communications (V2X), security.**

I. Introduction

Reducing traffic congestion and improving safety for vehicles and pedestrians are among the main goals of an Intelligent Transportation System (ITS). To this end, vehicle-to-everything (V2X) communications are employed for distributing information about road conditions (e.g., accident or speed limit alerts) and nearby vehicles (e.g., position, speed, and direction) [1]. Given their critical nature, safety messages must be authenticated, preventing malicious entities from disseminating fake data. This is usually accomplished by establishing a Vehicular Public Key Infrastructure (VPKI) [2], where authorized vehicles receive valid digital certificates for signing their messages, and can be evicted/suspended from the system in case of misbehavior. Examples of prominent VPKIs are the Security Credential Management System (SCMS) [3], which is part of the IEEE 1609.2 standard [4], and the Cooperative Intelligent Transport Systems (C-ITS) [5], proposed by the European Telecommunications Standards Institute (ETSI). Both solutions provision vehicles with multiple, simultaneously valid pseudonym certificates (e.g., up to 100 [6]). As a result, vehicles can avoid long-term tracking by periodically changing the pseudonym employed when signing messages.

[1]Escola Politécnica, Universidade de São Paulo, Brazil. Email: {ecominetti,mvsilva,mjunior}@larc.usp.br. [2]LG Electronics, USA. Email: {harsh.patil,jefferson1.ricardini}@lge.com

Since vehicles should quickly react to any emergency, safety messages exchanged among them must follow strict timing requirements [7]. For example, to enable the quick distribution of such messages, they are designed to have a small size. Specifically, their payload is 50-300 bytes [8], plus security-related information, like a 117-byte pseudonym certificate and a 64-byte digital signature [9]. Since all messages must be signed, but four out of five might include only an 8-byte certificate identifier rather than the actual certificate [10], their total size should range from 122-481 bytes. Besides, the maximum latency for processing critical messages (e.g., those related to crashing avoidance) can be as low as 20 ms [11].

Likewise, it is also imperative to minimize the time taken for processing safety messages after arrival. Part of this processing time is due to the application(s) responsible for handling those messages, such as trajectory prediction and collision avoidance software. Another important component, though, refers to the messages' authenticity verification process, in special because the number of individual messages received per second by each vehicle may reach the order of thousands [7][12][13].

Considering the timing requirements, during the standardization of V2X message broadcasting in the United States, the Crash Avoidance Metrics Partnership (CAMP) considered three approaches for message authentication [12]: simply using ECDSA digital signatures [14], the TESLA authentication scheme (based on Messages Authentication Codes) [15], or a combination of both (named TADS). In the end, the decision was to use only ECDSA signatures, an approach that is also adopted in European standards [16]. In part, this choice was motivated by the fact that TESLA and TADS rely on the delayed disclosure of the secret keys employed for authentication, creating critical time dependencies [12, Sec 8.2.1].

In addition, to deal with the large number of messages received by vehicles, [12] proposes the adoption of a Verify-on-Demand (VoD) mechanism [7]. In summary, this means that received messages are first processed by the application layer, and then their authenticity is verified only when some warning or control action is needed (e.g., if they indicate the possibility of a collision). As a result, the security overhead becomes proportional to the number of relevant messages received by the vehicle, i.e., only messages that raise the system's threat level beyond a defined threshold are verified. In regular operations, this strategy is expected to be considerably more efficient than a "verify-then-process" approach, in which all received messages would have their authenticity verified before being handed over to the application layer. However, this higher efficiency due to on-demand verification may

be problematic when the application layer requires several messages to be verified at once. This is the case, for example, for commonly employed trajectory prediction algorithms that rely on a sliding-window of past messages [17], [18], [19], [20]; whenever a potential emergency is detected, all messages in the sliding window that led to the prediction would have to be verified in batch. The system would then be forced to decide between two strategies: (1) verify all messages, even if this delays potentially critical actions; or (2) verify as many messages as possible in an acceptable time interval, and then risk to take actions based on false data. Designing mechanisms to minimize the overhead of individual and multiple signature verifications in V2X communications is paramount to optimize either strategy and, thus, avoid the need for more expensive onboard hardware (one of the goals of VoD itself).

In this manuscript, our goal is to tackle this issue of efficiently verifying multiple messages sent by the same pseudonym. Specifically, we evaluate schemes that allow signature verifications to be traded by operations that orders of magnitude less resource intensive [21], like hash functions and Message Authentication Codes (MACs). For that purpose, we first revisit the TADS scheme from [12], noticing (1) that its underlying mechanisms can be adapted to efficiently verify multiple messages on demand, and (2) that CAMP and subsequent works on secure V2X communications (e.g., [22]) only consider ECDSA in a VoD setting, but not TADS. In addition to this modified, VoD-oriented TADS, we assess a similarly efficient mechanism that avoids TADS's inherent critical time dependencies. Specifically, instead of relying on delayed disclosure of keys, we consider a mechanism similar to the Efficient Multi-chained Stream Signature (EMSS) scheme [15]: the solution is based on hash chaining [23], meaning that each signed message is linked to its predecessors by carrying their hashes. In both cases, verifying the signature of one message is enough to ascertain the authenticity of prior messages; consequently, at the cost of a small increase in the messages' size, one can obtain considerable performance gains when compared to individual signature verifications. The resulting solutions are evaluated in different settings, including tests on the SUMO [24] vehicular network simulator.

The rest of this paper is organized as follows. Sec. II gives additional background on the contents and requirements commonly considered for V2X messages. Sec. III describes strategies for the batch verification of V2X messages, presenting the time-dependent TESLA and TADS schemes, and a time-independent scheme based on hash chaining. Sec. IV shows our experimental results. Sec. V discusses related works. Finally, Sec. VI presents our final considerations.

II. Background: V2X communications

To support V2X communications, vehicles must have access to reliable and low-latency radio technology. Some of the technologies developed for this purpose are Dedicated Short-Range Communication (DSRC) and Cellular-V2X (C-V2X), whereas more efficient and scalable alternatives like New Radio V2X and 802.11bd are currently under development [25]. Whichever the case, a 300m minimum transmission radius with a packet error rate below 10% is expected [10].

Using such communication capabilities, vehicles can exchange relevant information with their surroundings, including other vehicles, roadside equipment, and pedestrians. The goal of those messages is to inform about their respective locations and status at each time, so adequate actions can be taken (semi-)automatically to improve transportation safety and efficiency. For example, with the regular broadcast of position and brake status, collisions can be avoided if vehicles autonomously adjust their speed, or issue warnings to drivers not respecting a safe following distance; as another example, smart traffic lights can adjust their cycling times aiming to optimize the flow of vehicles and pedestrians. The expected frequency in which messages are sent by each vehicle is around 10 Hz (i.e., 10 messages per second) [10], although some specifications allow a lower rate, e.g., to avoid communication channel [1]. In a busy road, this may translate to more than 1,000 messages received per second by each vehicle [7], [12].

Two are the main data formats for conveying V2X safety information: Basic Safety Messages (BSM), defined in the SAE J2735 standard [26], [10] and adopted in IEEE 1609.2; and Cooperative Awareness Messages (CAM) [1], employed in C-ITS. BSM messages include the following mandatory fields (which, for privacy-preserving purposes, are not easily linkable to a specific vehicle, driver, or owner): *Message Count*, a 7-bit sequence incremented for every message, which facilitate detection of transmission errors ; *Temporary ID*, a random 4-byte number; *Time*, a timestamp using the Coordinated Universal Time (UTC); *Location*, the latitude and longitude of the vehicle; *Elevation*, with a 3-meter accuracy; *Speed*, reported in increments of 0.02 mph (0.032 km/h); *Heading*, i.e., motion of the vehicle's center; *Acceleration*, for both location and elevation; *Yaw rate* and *Steering wheel angle*, which help predicting if the vehicle is spinning; *Transmission state* (neutral, reverse, or forward); and *Vehicle size* (length and width). SAE J2735 also describes optional fields that can be sent periodically, possibly at a reduced rate compared to the overall BSM transmission rate [12]. Examples are brake system status, path history, path prediction, exterior lights and multiple event flags (e.g., airbag deployment or flat tire).

The structure of Cooperative Awareness Messages (CAMs) comprises the same or similar fields, including generation time, position, speed, heading, and acceleration. One main exception among the mandatory fields refers to the absence of the "Message Count" element. Given this similarity between CAMs and BSMs, we write "BSM" to indicate both types of safety messages in the rest of this document, unless explicitly stated otherwise to consider the differences between them.

III. Batch verification of V2X messages

In the V2X scenario, all safety messages must be digitally signed by the sender, so receiving vehicles can check their authenticity before acting upon them. As a result, messages are discarded if they have invalid signatures or are not associated with a certificate issued by an authorized Certificate Authority (CA). Given the large number of messages received per second, though, vehicles may be required to prioritize the verifications of messages considered more critical, using

Algorithm 1 Batch verification with ECDSA in a VoD setting. Cost: 1 signature verification per non-null message.

INPUT: $m_{[1..n]}$ ▷ Sequence of n messages to be verified
INPUT: U ▷ Signer's public key
OUTPUT: $V_{[1..n]}$ ▷ Sequence having only verified messages
1: **for** $(i \leftarrow n\ ;\ i > 0\ ;\ i \leftarrow i-1)$ **do**
2: **if** ($m_i \neq$ `null` **and** checkSig(m_i, U)) **then**
3: $V_i \leftarrow m_i$ ▷ message received with valid signature
4: **return** V ▷ 1 signature verification per message

a Verify-on-Demand (VoD) approach [7]. In particular, the safety applications running on the vehicle could provide a "threat level" associated with each set of messages received, based solely on their payload. Whenever the threat level surpasses a certain threshold, the authenticity of the n messages that raised the issue must be verified. Such a batch verification procedure is required, for example, in Forward Collision Warning (FCW) applications, where multiple messages are used for computing the trajectory of a vehicle ahead, in the same lane and direction. Ideally, this verification should be finished before any automatic action is taken by the vehicle or a warning is issued to its occupants, aiming to avoid hazardous responses or raising false alarms.

In principle, verifying the authenticity of n safety messages would require checking their n individual signatures. This is depicted in Algorithm 1, which takes as input a sequence of BSMs that may include: authentic messages, i.e., whose signatures are valid; forged messages, with invalid signatures; and gaps (identified as `null`), due to errors. The algorithm then filters out invalid messages, leaving them as gaps in the output array. We note that, in practice, gaps in the input sequence can be inferred by the application layer using the messages' timestamps. Actually, since BSMs (unlike CAMs) have a mandatory "Message Count" field [10], gaps can be even more easily identified when counting values are skipped.

Our goal in this section is to investigate techniques for avoiding the burden of verifying n individual signatures on demand as in Algorithm 1. In particular, we focus on mechanisms that considerably accelerate verification whenever (some of) those n messages come from the same source.

A. Schemes with time dependencies

We start by describing the two schemes considered in [12] for reducing the cost of V2X message verifications in comparison with regular ECDSA digital signatures: the Timed Efficient Stream Loss-tolerant Authentication (TESLA) scheme [27], and its merger with ECDSA (TESLA and Digital Signature - TADS). We then show how TADS can be adapted to a VoD setting, which is our objective with this discussion.

1) TESLA: The Timed Efficient Stream Loss-tolerant Authentication (TESLA) scheme relies on a Message Authentication Code (MAC) whose secret keys, generated via cryptographic pseudorandom functions, are disclosed in a timeliness manner. More formally, TESLA considers three sequential time instants, t_0, t_1, and t_2. At instant t_0, the message sender S commits to a key k by publishing its hash $\mathcal{H}(k)$ via an authenticated channel. This commitment is then used as a secure anchor for the authentication scheme. At t_1, message m and its authentication tag $\texttt{MAC}_k(m)$ computed with secret key k is sent by S. Such a message cannot be authenticated by receiver R until instant t_2, when S publishes k. Then, at t_2, R learns the disclosed k and performs two verifications for authenticating m. First, R computes $\mathcal{H}(k)$ and checks if it matches the secure anchor of t_0. Second, if the first check is successful, R verifies if $\texttt{MAC}_k(m)$ matches the authentication tag received with message m.

The scheme's security relies on: (1) the fact that it is infeasible to compute k from $\mathcal{H}(k)$, assuming $\mathcal{H}$ is a cryptographically secure hash function; and (2) the assumption that no message authenticated with k is accepted by receivers after instant t_2, when that key is published. Therefore, such a delayed authentication can be performed safely as long as the sender and receiver are loosely synchronized.

Building upon this single-message approach, TESLA can be expanded to authenticate multiple messages, sent at different time windows. This is accomplished by using a hash chain to generate multiple keys from the same random seed key k_s. First, the following hash chain is computed:

$$k_n = \mathcal{H}(k_s); k_{n-1} = \mathcal{H}(k_n); k_{n-2} = \mathcal{H}(k_{n-1}); \ldots; k_0 = \mathcal{H}(k_1);$$

Then, at instant t_0, the sender publishes k_0 using an authenticated channel as the secure anchor for the following messages. Thereafter, at instant t_i, for $1 \leq i \leq n$ and $t_i < t_{i+1}$, message m_i is sent with authentication tag $\texttt{MAC}_{k_i}(m_i)$. Finally, at instant t_{n+1}, k_s is published. The receiver can then compute the entire hash chain from k_s, and check if the last element calculated, k'_0, matches the broadcast value k_0. If that is the case, all the computed k_j $(0 \leq j \leq n)$ are considered authenticated, and the receiver proceeds by calculating all $\texttt{MAC}_{k_j}(m_j)$ and validating the corresponding messages. Aiming to shorten the authentication delay, the sender may also include key k_{i-a} (where $a \geq 1$) with message m_i, so the receiver does not have to wait until instant t_{n+1} to start verifying messages.

Despite its high efficiency, and besides the time dependency resulting from the (delayed) disclosure of keys, TESLA has two important limitations. The first is that the authentication scheme is highly dependable on the correct reception of the secure anchor. After all, if k_0 is not received at instant t_0, the receiver cannot authenticate the ensuing messages that depend on that anchor. The second deficiency is the time delay forcibly introduced in the verification procedure, since no message in the chain can be validated before its key is published. The TESLA and Digital Signature (TADS) scheme was designed to mitigate these problems.

2) TADS: The TESLA and Digital Signature (TADS) scheme [12] was proposed to address two deficiencies of the standalone TESLA: its critical dependency on the distribution of secure anchors, and delayed message verification. Albeit quite simple, the solution of adding a digital signature in complement to the TESLA scheme is effective. Specifically, all messages in TADS carry, in addition to their authentication tag, a digital signature. Hence, receivers can fall back to signature verification when validating messages (1) if the secure anchor transmission is lost or (2) if verification is required before the corresponding key (or its seed) is published.

Algorithm 2 Adapting TADS to VoD. Valid signature enables the verification of multiple messages using their MACs.

INPUT: $m_{[1..n]}$ ▷ Sequence of n messages to be verified
INPUT: U ▷ Signer's public key
OUTPUT: $V_{[1..n]}$ ▷ Sequence having only verified messages
for $(i \leftarrow n\ ;\ i > 0\ ;\ i \leftarrow i-1)$ **do**
  if $m_i \neq$ `null` **and** checkSig(m_i, U) **then**
    $V_i \leftarrow m_i \quad ; \quad k \leftarrow k_i$
    break ▷ valid signature enables MAC computation
for $(i \leftarrow i-1; i > 0\ ;\ i \leftarrow i-1)$ **do**
  $k \leftarrow \mathcal{H}(k \,\|\, U)$ ▷ MAC key for this message
  if $(m_i \neq$ `null` **and** checkMac$(m_i, k))$ **then**
    $V_i \leftarrow m_i$ ▷ message received with valid MAC
return V ▷ 1 signature verification if all valid

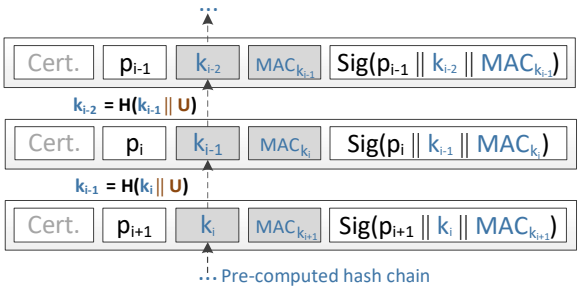


Fig. 1: Message verification with VoD-adapted TADS.

The main drawback of TADS in comparison with TESLA lies, thus, in the additional bandwidth required to transmit digital signatures with every message.

3) TADS and VoD: It is reasonable simple to adapt the TADS framework for a VoD setting (see Algorithm 2 and Fig. 1). Indeed, by using the variant of TESLA where every message m_i carries the key k_{i-1} employed in the authentication of the immediately preceding message m_{i-1}, any message acts as a secure anchor for the key hash chain. Because each message m_i is also signed, verifying this signature indirectly validates key k_{i-1}, as well as all previous keys k_{i-j} (with $j > 1$) that can be derived from it. Note that, with this adaptation, there is no need for the sender to commit to a specific security anchor k_0. After all, the signature verification over k_{i-1} already ensures the legitimacy of any pre-image of k_{i-1}, including the actual anchor k_0. The resulting scheme can then efficiently handle transmission errors: even if there is a gap in the sequence of messages to be verified, the receiver can still use k_{i-1}, validated via signature verification, to compute the MAC keys for messages placed before and after the gap.

One additional tweak required in this adapted TADS, though, refers to how the hash chains linking the different MAC keys are computed: instead of computing $k_{i-1} = \mathcal{H}(k_i)$, vehicles should make $k_{i-1} = \mathcal{H}(k_i, U)$, where U is the public key enclosed in the pseudonym certificate employed by the signer when sending message m_i. The reason for this modification in the original TADS is to avoid birthday attacks similar to those discussed in [28]. More precisely, let the keys employed in the system be ℓ bits long, so the hash function employed is truncated to match that length; for example, in [29], a value of $\ell = 96$ is suggested. Also, suppose that an attacker $\mathcal{A}$ picks random ℓ-bit values x and computes their corresponding hash chains $\mathcal{H}^c(x)$, for some arbitrarily small c. Assume that $\mathcal{A}$ builds a table containing 2^{ℓ_1} pairs $\{\mathcal{H}^c(x), x\}$ for some arbitrary $\ell_1 < \ell$, and is later able to observe 2^{ℓ_2} secret keys k_i disclosed by legitimate vehicles. According to the birthday paradox, as $\ell_1 + \ell_2$ approaches ℓ, there is a high probability that at least one of those k_i will match one of the table indices $\mathcal{H}^c(x)$. Whenever that occurs, $\mathcal{A}$ can repeatedly hash the corresponding x to obtain some $k_{j>i}$ that are valid pre-images of k_i. Assuming that such $k_{j>i}$ match the ones employed by the legitimate sender, rather than being second pre-images, they can be used by $\mathcal{A}$ to compute valid MACs for forged messages in subsequent time frames $t_{j>i}$. This does not allow $\mathcal{A}$ to compute valid signatures for those forged messages, though. In particular, the message that triggers the verification procedure is likely to have its signature evaluated by the receiver, and, thus, it might be difficult to trigger such alerts using this attack technique. Nevertheless, after a legitimately signed message triggers an alert, attackers would be able to trick the receiver into accepting some invalid messages, which in itself may be harmful to the system.

To put this threat in perspective, consider the choice of $\ell = 96$ from [29]. Suppose an attacker $\mathcal{A}$ places 200,000 V2X-enabled communication devices at very busy road points. Each device is then expected to receive up to 3,000 messages per second from passing vehicles [13], which roughly translates to 2^{28} messages per day per device. If the attacker builds a table with $2^{50.5}$ entries, the fact that $2^{50.5} \times 200,000 \times 2^{28} > 2^{96}$ means that, daily, at least one vehicle's message would match a table entry with high probability. The forgery of MACs would then be possible for that vehicle. The table itself could be accessed from some cloud storage, as long as the corresponding server has an upstream throughput of at least $200,000 \times 3,000 \times 12 = 7.2$ GBps and can perform the 600 million table look-ups with low latency. Assuming that the price of a repository with 1 petabyte can be as low as U$35,000 [30], storing $2^{50.5}$ of such 24-byte entries would cost less than $175,000.

Fortunately, adding the public key U to the computation of hash chains as hereby described frustrates such birthday attacks. The reason is that, analogous to security strings in SCMS's certificate linkage procedure [28], each attack table would only be useful for a specific U. Therefore, the aforementioned storage costs would have to be multiplied by the (huge) number of keys in the system. Also, V2X communications operate with short-lived pseudonyms that are revealed only when the vehicle starts transmitting messages under that pseudonym [3], [31]. Consequently, any computational resource invested for building such attacks would go to waste after the corresponding pseudonym certificate expires (e.g., within one week [3]). In the end, besides being thwarted from building attack tables beforehand, attackers that do so on the fly would hardly have a chance to find collisions with that table's entries given the limited number of legitimate messages generated under the same U. For example, suppose that a vehicle sends safety messages uninterruptedly, and alternate between 40 pseudonym certificates along a week (i.e., every certificate is used for a total of $24 \times 7/40 = 4.2$ hours); even if an attacker can somehow eavesdrop on all messages sent under

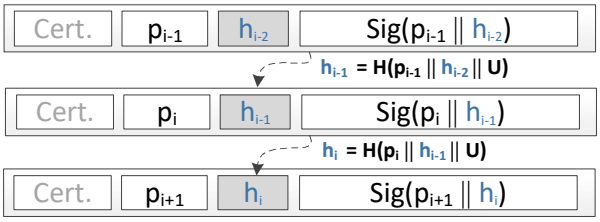


Fig. 2: Message verification via hash chaining.

the same pseudonym in this extreme scenario, there would be only $4.2 \times 36000 \approx 2^{17}$ messages to use in the attack. Consequently, the attack table would require $2^{\ell-17}$ entries, which should be infeasible for $\ell \geqslant 96$.

B. Avoiding time dependencies via message chaining

Albeit efficient, the adaptation of TADS for a VoD setting described in Sec. III-A3 still critically depends on senders and receivers being (loosely) synchronized. Otherwise, after key k_j is disclosed, an attacker may modify the original message m_j, turning it into m'_j, and recompute its authentication tag as $\mathtt{MAC}_{k_j}(m'_j)$. Although forging the corresponding ECDSA signature for m'_j would still be unfeasible, that would be enough to trick the receiver into accepting m'_j as valid whenever only this message's authentication tag is verified.

In principle, this need for synchronization should not be a major issue in V2X communications. After all, (1) safety messages carry GPS-based timestamps; and (2) the time skew between vehicles' clocks and the reference clock should be below 1.5 ms [29], much lower than the 100 ms interval between BSMs [10]. Nevertheless, since the NHTSA study indicated that TADS's time dependencies are a burden to its adoption in V2X communication [12, Sec 8.2.1], it is relevant to consider alternatives without such dependencies. For this reason, we hereby consider an approach similar to the Efficient Multi-chained Stream Signature (EMSS) scheme, based on hash chaining (see Fig. 2): instead of relying on the delayed disclosure of secret keys, every BSM can carry the hash of its predecessor, or a few predecessors. The signature of one BSM can, thus, validate a series of previously received BSMs.

1) Basic scheme: Suppose that the i-th safety message sent by a vehicle corresponds to the triple $m_i = (p_i, h_{i-1}, sig_i)$, where: p is the message's payload (described in Sec. II); $h_{i-1} = \mathcal{H}(p_{i-1} \,\|\, h_{i-2} \,\|\, U)$ is the preceding message's hash, m_{i-1}, combined with the sender's public key U; and $sig_i = Sig(p_i \,\|\, h_{i-1})$ is the signature of m_i, computed using the sender's private key whose public counterpart U is enclosed in a valid pseudonym certificate. In this scenario, if m_i's signature is authentic, the previously transmitted $m_{i-1} = (p_{i-1}, h_{i-2}, sig_{i-1})$ can be validated by checking whether $h_{i-1} = \mathcal{H}(p_{i-1} \,\|\, h_{i-2} \,\|\, U)$ [15]. After all, attackers should be unable to forge a message with the same hash as m_{i-1} unless they find a second pre-image for h_{i-1}, which should be infeasible for a second pre-image resistant hash function [32]. Similarly, m_{i-2} can be considered valid as long as its hash matches the value of h_{i-2} enclosed in an authentic m_{i-1}, and so forth. Hence, when checking a sequence of n messages from a single sender, there is no need to verify each individual signature $sig_{i-1} \ldots sig_{i-n}$, since the data authenticated by such signatures are also indirectly authenticated by sig_i. This allows the receiver to trade signature verifications for hash computations, amortizing the costs of the former over multiple messages. Namely, the only situation in which the described hash chaining approach would not provide any gain is when every other message is lost during transmission. In this peculiar case, all $\lceil n/2 \rceil$ received messages need to have their signatures verified because the hashes that would enable their faster verification would be unavailable.

Algorithm 3 Batch verification with hash chaining. At the cost of 1 hash computation, some valid messages can be indirectly verified and invalid messages are filtered out.

INPUT: $m_{[1..n]}$ ▷ Sequence of n messages to be verified
INPUT: U ▷ Signer's public key
OUTPUT: $V_{[1..n]}$ ▷ Sequence having only verified messages
1: **for** $(i \leftarrow n \; ; \; i > 0 \; ; \; i \leftarrow i - 1)$ **do**
2: **if** $(m_i \neq \mathtt{null}$ **and** checkSig$(m_i, U))$ **then**
3: $V_i \leftarrow m_i$ ▷ message received with valid signature
4: **while** $(m_{i-1} \neq \mathtt{null}$ **and** $\mathcal{H}(m_{i-1} \,\|\, U) = h_i)$ **do**
5: $V_{i-1} \leftarrow m_{i-1}$ ▷ Match to h_i, carried by valid m_i
6: $i = i - 1$ ▷ skip message: indirectly verified
7: **return** V ▷ 1 to $\lceil n/2 \rceil$ signature verifications if all valid

We note this approach's resilience to birthday attacks like those described in Sec. III-A3 against the original TADS. Once again, this happens because the public key U included in the hash computation acts as a salt, limiting the attackers' capability of pre-building an attack table and gathering enough legitimate messages to find collisions against its entries.

Algorithm 3 shows this verification approach based on hash chaining. Algorithm 3 assumes that the order of the messages can be inferred from their timestamps or message count values (for BSMs). Nevertheless, it can be easily adapted to handle inputs where multiple messages appear to occupy the same time slot, i.e., that have very close timestamps or the same message count. Specifically, since this situation should not occur unless fake messages are inserted in the network, it would suffice to look for the first message whose signature or hash value is valid among the duplicates. If none is found to be valid, then the authentic message was probably lost during transmission, and the algorithm would keep `null` in the output array's corresponding position to indicate that loss.

2) Extended chaining: The basic approach described in Sec. III-B1 can be extended in a manner that each m_i carries multiple hashes instead of one. The motivation for an extended chaining can be understood with a simple example. Suppose that each m_i includes only the hash for m_{i-1}, as shown in Fig. 2. If a vehicle receives m_{i-2} and m_i, but misses the transmission of m_{i-1}, then the link between m_{i-2} and sig_i is lost. Therefore, the authenticity of the received messages can only be ascertained by verifying sig_{i-2} and sig_i directly.

To avoid this penalty, m_i could carry the hashes for m_{i-2} and m_{i-1}, so any non-contiguous message loss would not incur additional signature validations. More generally, a few hashes of previous (possibly non-contiguous) messages could

Algorithm 4 Message verification algorithm: extended chaining, assuming that each message carries e hashes. At the cost of 1 hash computation, some valid messages can be verified and invalid messages are filtered out.

INPUT: $m_{[1..n]}$ ▷ Sequence of n messages to be verified
OUTPUT: $V_{[1..n]}$ ▷ Sequence having only verified messages
$M \leftarrow$ new $Map()$ ▷ Messages to verify, mapped by index
for all $(m_i \neq \texttt{null})$ **do**
  $M.put(\{key : i, value : m_i\})$
while $(M \neq \varnothing)$ **do**
  $m_\ell \leftarrow M.removeMaxKey()$ ▷ Latest m to be verified
  if (checkSig(m_ℓ)) **then** ▷ Message's signature is valid
    $Q \leftarrow$ new $Queue(m_\ell)$ ▷ Enqueue verified messages
    while $(Q \neq \varnothing)$ **do** ▷ Try some indirect verifications
      $m_i \leftarrow Q.Dequeue()$
      $V_i \leftarrow m_i$
      for all $(h_{i-j} \in m_i)$ **do** ▷ Check message's hashes
        if $(m_{i-j} \in M)$ **then** ▷ m_{i-j} not yet validated
          if $(\mathcal{H}(m_{i-j} \,\|\, U) = h_{i-j})$ **then**
            ▷ Match to h_{i-j}, carried by valid m_i
            $Q.Enqueue(m_{i-j})$
          ▷ Remove m_{i-j} from M: it is valid (above),
          ▷ or it is invalid due to failed hash verification
          $M.removeKey(i-j)$
return V ▷ 1 to $\lceil n/(e+1) \rceil$ signatures verified if all valid

be included in each broadcast. Nevertheless, in the specific context of V2X, such an "extended chaining" approach should be considered with care, as discussed in what follows.

First, critical applications are the most prone to benefit from faster verification of a sequence of messages, so they can help drivers to react quickly. However, they are also less likely to require the verification of many old messages, in particular messages older than the driver's natural reaction time. Since the average reaction time for an attentive driver is around 1 second (see [33] and references therein), the ability to quickly verify sequences containing much more than 10 or 20 messages may not be of much use for critical applications.

Moreover, the bandwidth overhead of each additional hash is not totally negligible when compared to the size of V2X messages. More precisely, assume 16-byte hashes, so the system's resistance against second pre-image attacks is 2^{128}. Each hash corresponds, thus, to an overhead of 3% to 13% over the V2X messages' expected size of 122-481 bytes; since the larger size is applies to messages carrying certificates, which correspond to 20% of all messages, the average communication overhead for each hash is around 8.3%. We note that this overhead considers only mandatory BSM fields and, thus, it should be proportionately smaller if messages also include optional fields like those mentioned in Sec. II. Given that reducing the safety messages' size is a core requirement in V2X environments, adding much more than one or two hashes per message may be too much of a bandwidth burden.

Finally, one of the main benefits of this extended chaining is to make the system more resilient to lossy communication channels (albeit not as resilient as TADS). In particular, including the hashes for non-contiguous packets in each message is useful to handle burst packet losses: e.g., if two hashes are placed in m_i, then choosing (h_{i-1}, h_{i-3}) instead of (h_{i-1}, h_{i-2}) allows m_{i-3} to be indirectly verified even if both m_{i-1} and m_{i-2} are lost. However, in some cases the indirect verification would not be possible without incurring delayed verification: e.g., if m_i carry h_{i-1} and h_{i-3} and m_{i-1} is lost, the indirect verification of m_{i-2} would not be possible until the arrival of m_{i+1} (carrying h_i and h_{i-2}). Since this delay translates to about 100 ms, which should be time enough to verify the signature of m_{i-2} directly, this benefit of the extended chaining with non-contiguous hashes would dwindle. Moreover, empirical studies indicate that burst data loss is actually not the most common in V2X environments [34], [35]. Indeed, such studies show that Inter-Packet Gaps (IPGs) of up to 400 ms (i.e., up to 3 contiguous packets lost) are to be expected, while longer IPGs are usually correlated (e.g., they indicate that the sender moved too far from the receiver).

C. Impact of changing pseudonym certificates

Both TADS and the described hash chaining approach use the vehicle's pseudonym certificate public key U in the computation of chaining sequences. Although this key is regularly changed during communications, this procedure's impact on the performance of both schemes is expected to be minimal in practice. The reason is that pseudonym changes in V2X communications should not occur much more often than 2-6 minutes, aiming to preserve the accuracy of safety applications [36], [37]. The simulation from [29, Appendix G-1] considered an even worse scenario, where vehicles exchanged their certificates every minute. With 10 messages sent per second, this would translate to a scenario where any negative impact from pseudonym changes occurs only once every 600 messages, when a new chain has to be created. Even so, the simulation results showed that less than 1% of the messages are lost due to the certificate change when TESLA was used.

We also note that, if a vehicle changes its pseudonym, messages sent under different identities are not expected to be linked and cannot benefit from the hereby discussed batch verification methods. After all, the objective behind a pseudonym certificate system is to provide unlinkability between messages from different certificates belonging to the same vehicle. Obviously, a pseudonym change must also lead to the usage of different chains to ensure privacy preservation.

D. Compatibility with existing standards

At least in principle, the hereby described approaches benefit from the IEEE 1609.2 packet format [4]. The reason is that this format assumes that the payload to be signed may include both: (1) data explicitly transported within the packet; and (2) the hash value of data not explicitly transported within the packet, but that the sender wants to cryptographically bind to the signature. Hence, in the case of the hash chaining approach, this latter "external data" field could correspond to the hash of the previously sent message (or omitted for the first message sent under a pseudonym). For TADS, the field could carry the message's authentication tag and the key employed in the preceding message; in this case, and even though a

MAC is essentially a keyed hash function, this can be seen as a loose interpretation of this field. With either approach, the only modification required on current V2X standards would be on the security profile defined by the J2945/1 format for IEEE 1609.2 messages: this profile should allow additional data to be signed for BSMs, which is forbidden in [37, Table 10].

As an alternative, the fields added by both solutions (highlighted in grey in Figures 2 and 1) could become part of the payload's data format, so no other field is required. Actually, if the mechanism adopted for fast batch verification adds more than 256 bits to each BSM, this alternative would be inevitable, given that this is the maximum size of the aforementioned "external data" field in the J2945/1 message format.

IV. Simulation and analysis

To evaluate both the schemes described in Sec. III, we simulated TADS and several hash chaining configurations aiming to determine their average verification costs. Our simulator, available at [38], takes into account the following parameters:

- Window size: we consider scenarios in which the sequence of messages to be verified fits different batch sizes. This is useful to assess how each solution would perform in different application scenarios (e.g., different window sizes used by a trajectory prediction algorithm).
- Missing packets: the number and pattern of missing packets inside a message window influence the number of signatures that must be directly verified by the hash chaining approach. Thus, our simulations consider all possible patterns for each window size, going from a scenario where only m_{i-1} is missing to a scenario where m_{i-1} through m_{i-w+1} are lost. We also single out patterns containing only burst data losses, aiming to evaluate which chaining configurations would better fit some cases. Albeit such patterns are not likely to be too common in V2X environments, the corresponding tests enable the comparison with related works in Sec. V.
- Hash chaining configurations: every m_i may carry one, two or three hashes. We denote by $C[p, j, k]$ the configuration where m_i carries $(h_{i-p}, h_{i-j}, h_{i-k})$, and can thus be indirectly verified using either m_{i+p}, m_{i+j} or m_{i+k}). For a window size of w, we evaluate all possible hash configurations, going from $C[1]$ (i.e., the basic scheme described in Sec. III-B1) to $C[w-1, w-2, w-3]$.
- psv metric: for each scheme and window size, we compute the percentage of signature verifications (psv) for all patterns of missing packets in that window. For example, when $\texttt{psv} = 0.1$, this means that 1 message out of 10 needs to have its signature verified, while the other 9 can have its authenticity checked indirectly, via hash or MAC computations. Given the negligible cost of hashes and MACs when compared to signature verifications [21], the corresponding performance gains are then $1 - \texttt{psv}$ (e.g., a $\texttt{psv} = 0.1$ translates to a performance gain of 90%). We then organize those patterns according to the total number of packets missed, i.e., going from the most reliable to a highly lossy communication environment.

First, to determine the best hash chaining configuration for window sizes w covering different numbers of messages, we simulated every possible combination of hash configurations for $w = 8$ to $w = 20$ (i.e., windows of 0.8 to 2 seconds). The result is that, on average, the contiguous configurations (namely, $C[1]$, $C[2, 1]$ and $C[3, 2, 1]$) display the best psv in all settings. This is shown in Fig. 3, which is a plot of the simulation results for $w = 10$ (left) and $w = 20$ (right). For comparison purposes, this figure also shows configurations where each message carries hashes for similarly-spaced messages – i.e., for a window size of w the configurations having $C[1+w/2, 1]$ for two hashes, and $C[1+2w/3, 1+w/3, 1]$ for three hashes. The potential interest of the latter configurations is that, according to [15] and to our simulations, they would lead to the best psv against burst data losses. Nevertheless, as depicted in Fig. 3, they perform worse than their continuous counterparts in a V2X scenario, where packet loss is not expected to follow an exclusively burst pattern [34], [35].

	psv reduction	% Missed packets 0	10	20	30	40	50	60	70	80	90
($w = 10$)	Standard − TADS	.90	.89	.88	.86	.83	.80	.75	.67	.50	.00
	Standard − $C[1]$	.90	.79	.68	.57	.46	.36	.25	.15	.06	.00
	$C[2,1] - C[1]$	.00	.10	.17	.21	.23	.22	.19	.13	.06	.00
	$C[3,2,1] - C[2,1]$	.00	.00	.02	.06	.10	.13	.13	.11	.06	.00
($w = 20$)	Standard − TADS	.95	.94	.94	.93	.92	.90	.88	.83	.75	.50
	Standard − $C[1]$	.95	.85	.74	.64	.53	.43	.32	.22	.12	.03
	$C[2,1] - C[1]$	.00	.09	.16	.21	.24	.24	.21	.17	.11	.03
	$C[3,2,1] - C[2,1]$	.00	.01	.03	.06	.10	.13	.14	.13	.09	.03

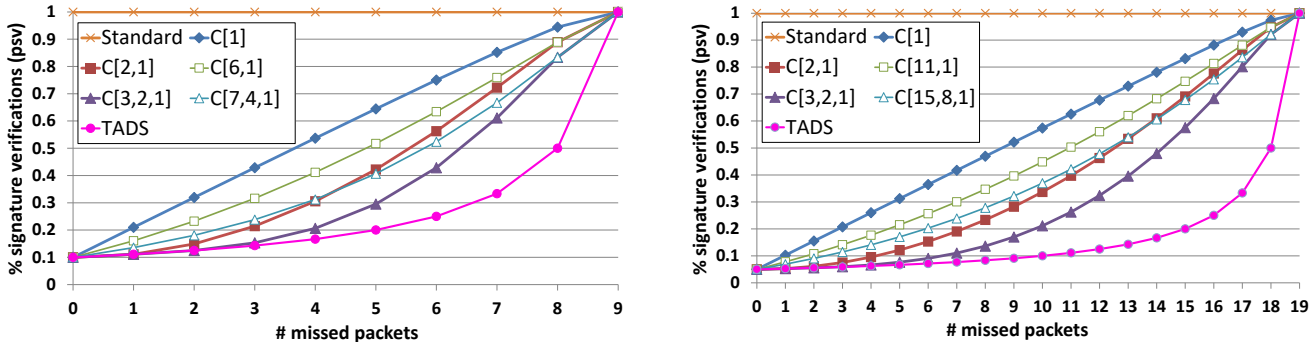
TABLE I: Reduction of psv brought by each additional hash carried by safety messages, for window sizes $w = 10$ and $w = 20$ (the higher, the better). TADS is included as reference.

We also evaluated the savings provided by each additional hash when vehicles need to authenticate a sequence of safety messages. The goal, in this case, is to assess the difference between the main curves in Fig. 3, understanding how the extra bandwidth overhead brought by each hash translates to lower processing. As shown in detail in Table I, the extra psv reduction for each hash after the first is small when less than 20% of the packets are lost. In such situations, a single hash is enough to ensure processing savings between 68% and 95% when compared to a scenario where all signatures are directly verified. A second hash may then be useful for less reliable communication scenarios. Indeed, when the packet losses reach up to 50%, this additional hash helps to keep the processing savings above 58%. A third hash, on the other hand, does not add much in terms of savings for the simulated messages windows. More precisely, this extra hash is particularly useful for packet losses higher than 50%, when the extra processing savings reach up to 14%.

In contrast, the TADS scheme always requires a single signature verification for any $w \geq 1$. Therefore, it provides lower psv in all scenarios, and is particularly interesting for larger window sizes and less reliable communication channels. TADS achieves savings from 80% to 95% when up to 50% of the packets are lost, considering both window sizes evaluated.

A. Validation with SUMO

In complement to the tests using a dedicated simulator, we also analyzed the hash chaining and TADS schemes in a traffic scenario under the Simulation of Urban Mobility (SUMO) tool



(a) $w = 10$ (b) $w = 20$

Fig. 3: Average percentage of signatures verified (psv) for tested window sizes (w) and configurations (lower means better).

Verification Strategy	Window Size $w_{max} = 10$	Window Size $w_{max} = 20$
Standard	115787 (psv = 1)	217064 (psv = 1)
TADS	14779 (psv = 0.13)	14779 (psv = 0.07)
Chaining $C[1]$	32681 (psv = 0.28)	52804 (psv = 0.24)
Chaining $C[2, 1]$	17956 (psv = 0.16)	22020 (psv = 0.10)
Chaining $C[3, 2, 1]$	15310 (psv = 0.13)	16185 (psv = 0.07)

TABLE II: Number of signatures to be validated by cars in the SUMO simulation using different verification approaches.

[24]. The goal of such simulation is to evaluate a more realistic V2X communication scenario, rather than assuming that all patterns for missing packets are equiprobable.

Our scenario uses the street map of Eichstätt, Germany. The map is based on the one available at [39], with minor modifications: routes restricted for pedestrians were removed; routes restricted for special vehicles were replaced by regular routes. This map is populated with 300 cars, driving at 3 different speeds (equivalent to 40, 50 and 60 km/h). When cars get in communication range, the equivalent to 0.5 to 30 meters, they exchange messages, at a rate of 10 messages per second. Then, we once again assume that each vehicle may have to verify all past messages in a window $w=10$ or $w=20$ once a critical message is received (this window includes the critical message). Each exchanged message has an unconditional 1% probability of being critical and a 20% unconditional probability of being missed. Moreover, since vehicles move at different speeds, they can get in communication range briefly before a critical message is received, resulting in smaller window sizes. In this case, the vehicle analyzes all the messages, even if they do not reach the expected window size. The total simulation time was 300 s, and the results use the number of messages verified by all vehicles. We emphasize that this simulation scenario is kept small because using a larger number of vehicles did not lead to more relevant results: after all, our goal is to evaluate how many signature verifications can be avoided whenever messages must have their authenticity validated, not to somehow "stress" the underlying virtual hardware. In a real-world, large-scale scenario, the same results hereby obtained are expected to be observed assuming that the vehicle can handle as many on-demand verifications as needed (and the hereby evaluated approaches help with meeting this goal). The resulting map and simulation scripts are available at [38].

The results of the simulations are summarized in Table II. As shown in this table, a total of 115787 and 217064 messages have to be verified for $w = 10$ and $w = 20$, respectively. Hence, when no chaining mechanism is employed (i.e., psv=1), these numbers correspond to the total number of signature verifications. When each message carries the hashes of predecessors or uses TADS, though, this number drops dramatically. Specifically, for the $C[1]$ hash chaining configuration, the savings are such that psv $= 0.28$ and psv $= 0.24$ for $w = 10$ and $w = 20$, respectively. When messages are verified using TADS, we obtained a psv$=0.13$ for $w=10$, and psv$=0.07$ for $w=20$. We note that these numbers are very close to the theoretical results from Fig. 3 for a 20% packet loss. Conversely, the $C[2, 1]$ and $C[3, 2, 1]$ configurations have a slightly worse performance for $w=10$. Nevertheless, for $w=20$, the results for $C[2, 1]$ and $C[3, 2, 1]$ are also similar to our theoretical results. Indeed, for $w=10$, we obtained psv $= 0.16$ for $C[2, 1]$ and psv $= 0.13$ for $C[3, 2, 1]$, when the expected values would be psv $= 0.15$ and psv $= 0.09$, respectively; for $w = 20$, the result was psv $= 0.10$ for $C[2, 1]$ and psv $= 0.07$ for $C[3, 2, 1]$, matching the expected psv$=0.10$ and psv$=0.07$ depicted in Fig. 3b. Such results corroborate the analysis from Sections III-A3, III-B2, showing that: (1) inserting at least one hash into BSMs enables relevant signature verifications savings; (2) for a verification window as small as 1 or 2 seconds and low packet loss rates, including a single hash (or maybe two) on each message is likely to be enough; and (3) when TADS's time dependencies can be tolerated, the savings are very significant.

B. Comparison: hash Chaining vs. TADS

As shown in Sec. IV, TADS is particularly interesting for reducing verification costs in lossy communication channels: it requires a single valid signature to verify every received message in a window even if many messages are lost. For reliable channels, on the other hand, the hash chaining approach can lead to competitive results. Specifically, $C[2, 1]$ and $C[3, 2, 1]$ can match TADS's performance for packet losses smaller than 10%, and 25%, respectively, whereas $C[1]$ only matches TADS's performance when no message is lost.

Each strategy also provides different bandwidth overheads. For attaining a 128-bit security level, each hash value em-

Method	Bandwidth overhead	Time dependencies?	Tolerance to data loss
$C[1]$	8.3%	No	Low
TADS	16.5%	Yes	Highest
$C[2,1]$	16.5%	No	Average
$C[3,2,1]$	24.8%	No	High

TABLE III: Comparison between the discussed methods.

ployed in a chaining-based approach should be 16-bytes long. Hence, the overhead of this approach is 16, 32 and 48 bytes for $C[1]$, $C[2,1]$ and $C[3,2,1]$, respectively (i.e., when using one, two and three hashes). Similarly, messages protected with TADS would carry a MAC of up to 16 bytes and a 16-byte key, so the overhead is at most 32 bytes per message. As discussed in Sec. III-B2, adding a 16-byte block to V2X messages translates to a communication overhead of $\approx$8.3%.

The results of these comparisons are compiled in Table III.

V. Related works

There are in the literature works aimed at enabling security in V2X communications (for a survey on security-related challenges, see [40]). However, most of the recent studies aims at efficiently provisioning [3][31] and revoking pseudonym certificates of misbehaving nodes [41][42][43][44]. These solutions are orthogonal to this study, whose focus is on the regular use of pseudonym certificates by non-revoked vehicles.

Many works in the literature discuss the challenge and importance of accelerating BSM verification in V2X environments [7][12][45][46][47] Solutions involve both the provisioning of highly optimized implementations [45] and enabling batch verification of signatures [46][47]. Indeed, as described in [48], ECDSA is amenable to batch verification for small batch sizes ($\leqslant 8$); when compared to the individual verification of signatures, the speed-ups obtained can reach $6\times$ when all signatures come from the same signer, and $2\times$ for different signers. In addition, a variant of ECDSA named ECDSA* [49] enables accelerations for larger batches, with speed-ups of $7\times$ and $4\times$ when the signatures belong to the same and to distinct signers, respectively [50]. In the V2X scenario, such gains can also be combined with the verification of the corresponding pseudonym certificates [46], [47] to amortize the latter operation's costs. Albeit appealing, batch verification has one important limitation: if it fails (i.e., at least one invalid signature is in the batch), vehicles must fall back to single signature verification [51]. Hence, attackers can force vehicles to repeatedly check individual signatures by inserting fake messages into the communication. This would not only nullify the gains expected from batch verification, but turn the attempt of verifying the batch into a waste of processing time. Conversely, the hereby evaluated strategies can trade signature verifications for simpler operations (hash and MAC computations) when verifying messages from the same vehicle. Hence, even in the presence of fake messages, there is no wasted effort in trying to perform such a "chained verification" after a valid signature is identified. Besides, the technique can be combined with batch signature verification schemes like those in [48],[50] for higher speed-ups whenever the batch contains some messages from the same signer.

The TADS scheme, whilst analyzed in [12] where it was originally proposed, has not been evaluated in a VoD setting. Besides closing this gap, we also contribute to the original TADS design by improving its resilience to birthday attacks.

Finally, the hash chaining approach hereby described shares similarities with EMSS [15]. In that scheme, messages are not individually signed, but instead carry one or more hashes of previous messages. Periodically, the sender broadcasts a digitally signed packet containing the hashes of the last sent message (and optionally of some extra messages, if robustness against packet loss is desired). Then, receivers need to verify only the authenticity of the signed packet, which indirectly validates all messages whose hashes are enclosed in either (1) the signed packet itself, or (2) any other validated message. This delayed verification of EMSS does not appear in the hereby evaluated hash chaining scheme, though, because in the latter all messages include their own signatures to be verified whenever needed. This design difference between the schemes also influences our experimental analysis, which shows that the best performance is achieved when hashes carried by each BSM refer to contiguous predecessors. This strikingly contrasts with EMSS's recommendations, aimed at increasing resilience to burst packet loss, where such a configuration would lead to the *worst* results [15, Sec. 3.1].

VI. Conclusion

In this manuscript, we discuss the challenge of timely verifying V2X safety messages. In particular, our goal is to ensure that V2X messages can be verified within tens of milliseconds, even though each vehicle may receive thousands of them per second. To this end, we evaluate two mechanisms in a VoD setting: the TADS scheme proposed in [12], which is modified to resist attacks based on hash collisions; and a strategy involved hash chainings, whose main benefit is to avoid the time dependencies observed in TADS. Our theoretical and experimental analyses show that the use of the TADS scheme can lead to savings of more than 80% when the communication channel packet loss rate is below 50%. Moreover, we also show that a hash chaining approach can present similar savings with lower overhead for highly reliable communication channels (less than 10% packet losses), or similar savings for slightly higher overheads. These methods can be further combined with other mechanisms intended for alleviating the load of V2X message validation, such as prioritization [7] and batch signature verification [46][47], to efficiently handle messages from different sources.

For future work, a study of the communication channel in V2X environments would be useful to evaluate the performance of the hereby described solutions in real-world scenarios. This analysis should assess the channel limitations, such as maximum acceptable bandwidth overhead, usual packet loss ratios, and the ability of modern V2X technologies (e.g., New Radio V2X and 802.11bd [25]) in handling TADS's time dependencies. The investigation of these variables is necessary to determine the actual suitability of TADS and the multiple hash chaining configurations evaluated in this manuscript.

Acknowledgment. This work was supported by LG Electronics, by Ripple's University Blockchain Research Initiative, and

by the Brazilian CAPES (Finance Code 001) and CNPq (grant 304643/2020-3).

REFERENCES

[1] ETSI, "EN 302 637-2 (v1.4.1) intelligent transport systems (ITS), vehicular communications, basic set of applications – part 2: Specification of cooperative awareness basic service," European Telecom Standards Institute, Tech. Rep., Apr 2019.

[2] M. Khodaei and P. Papadimitratos, "The key to intelligent transportation: Identity and credential management in vehicular communication systems," *IEEE Veh. Technol. Mag.*, vol. 10, no. 4, pp. 63–69, Dec 2015.

[3] B. Brecht, D. Therriault, A. Weimerskirch, W. Whyte, V. Kumar, T. Hehn, and R. Goudy, "A security credential management system for V2X communications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 12, pp. 3850–3871, Dec 2018.

[4] IEEE, "IEEE standard for wireless access in vehicular environments–security services for applications and management messages," *IEEE Std 1609.2-2016 (Revision of IEEE Std 1609.2-2013)*, pp. 1–240, Mar 2016.

[5] ETSI, "TR 102 731 – intelligent transport systems (ITS); security; security services and architecture," European Telecommunications Standards Institute, Tech. Rep., Set 2010.

[6] EU, "Certificate policy for deployment and operation of european cooperative intelligent transport systems (C-ITS) release 1.1," European Commission, Tech. Rep., Jun 2018.

[7] H. Krishnan and A. Weimerskirch, ""verify-on-demand": A practical and scalable approach for broadcast authentication in vehicle-to-vehicle communication," *SAE International Journal of Passenger Cars - Mechanical Systems*, vol. 4, pp. 536–546, Jun 2011.

[8] ETSI, "TS122 185 v14.3.0: LTE; service requirements for V2X services," European Telecommunications Standards Institute, Tech. Rep., Mar 2017.

[9] DOT, "Vehicle safety communications – applications (VSC-A). final report: Appendix volume 3 - security," US Department of Transportation (USDOT), Tech. Rep., Sep 2011.

[10] CAR, "Review of the NHTSA proposal to mandate V2V communication for safety," Center for Automotive Research, Tech. Rep., Dec 2016.

[11] NHTSA, "Vehicle safety communications project: HS 810 591 Final Report," US Dept. of Transportation, Tech. Rep., Apr 2006.

[12] ——, "Vehicle safety communications. Applications (VSC-A): HS 811 492A Final Report," US Dept. of Transportation, Tech. Rep., Sep 2011.

[13] S. Banani, S. Gordon, S. Thiemjarus, and S. Kittipiyakul, "Verifying safety messages using relative-time and zone priority in vehicular ad hoc networks," *Sensors*, vol. 18, no. 4, 2018.

[14] NIST, *FIPS 186-4: Digital Signature Standard (DSS)*, National Institute of Standards and Technology, July 2013.

[15] A. Perrig, R. Canetti, J. D. Tygar, and Dawn Song, "Efficient authentication and signing of multicast streams over lossy channels," in *Proc. of IEEE Symposium on Security and Privacy.*, May 2000, pp. 56–73.

[16] ETSI, "TS 103 097 v1.3.1 – intelligent transport systems (ITS); security; security header and certificate formats," European Telecommunications Standards Institute, Tech. Rep., Out 2017.

[17] P. Kumar, M. Perrollaz, S. Lefèvre, and C. Laugier, "Learning-based approach for online lane change intention prediction," in *IEEE Intelligent Vehicles Symposium (IV)*, 2013, pp. 797–802.

[18] S. Lefèvre, J. Petit, R. Bajcsy, C. Laugier, and F. Kargl, "Impact of V2X privacy strategies on intersection collision avoidance systems," in *IEEE Vehicular Networking Conference*, 2013, pp. 71–78.

[19] S. Yoon and D. Kum, "The multilayer perceptron approach to lateral motion prediction of surrounding vehicles for autonomous vehicles," in *IEEE Intelligent Vehicles Symposium (IV)*, 2016, pp. 1307–1312.

[20] F. Altché and A. de La Fortelle, "An LSTM network for highway trajectory prediction," in *IEEE 20th Int. Conf. on Intelligent Transportation Systems (ITSC)*, 2017, pp. 353–359.

[21] "eBACS: ECRYPT benchmarking of cryptographic systems," https://bench.cr.yp.to (accessed 20 March 2021), 2021.

[22] IEEE, *IEEE 1609.2.1-2020 - Standard for Wireless Access in Vehicular Environments (WAVE) – Certificate Management Interfaces for End Entities*, IEEE Standards Association, Dec 2020.

[23] L. Lamport, "Password authentication with insecure communication," *Commun. ACM*, vol. 24, no. 11, pp. 770–772, 1981.

[24] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "SUMO - simulation of urban mobility: an overview," in *Proc. of the 3rd Int. Conf. on Advances in System Simulation*. ThinkMind, 2011.

[25] G. Naik, B. Choudhury, and J. Park, "IEEE 802.11bd 5G NR V2X: Evolution of radio access technologies for V2X communications," *IEEE Access*, vol. 7, pp. 70 169–70 184, 2019.

[26] SAE, "J2735: Dedicated short range communications (DSRC) message set dictionary," SAE International, Tech. Rep., Mar 2016.

[27] A. Perrig, R. Canetti, J. D. Tygar, and D. Song, "The TESLA broadcast authentication protocol," *RSA CryptoBytes*, vol. 5, no. 2, pp. 2–13, 2002.

[28] M. Simplicio, E. Cominetti, H. Kupwade-Patil, J. Ricardini, L. Ferraz, and M. Silva, "Privacy-preserving certificate linkage/revocation in VANETs without Linkage Authorities," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–11, 2020.

[29] NHTSA, "Vehicle safety communications – applications (VSC-A): Final report (DOT HS 811 492D)," U.S. Department of Transportation (USDOT), Tech. Rep., Sep 2011.

[30] A. Klein, "Petabytes on a budget: 10 years and counting," backblaze.com/blog/petabytes-on-a-budget-10-years-and-counting, 2019.

[31] M. Simplicio, E. Cominetti, H. Kupwade-Patil, J. Ricardini, and M. Silva, "The unified butterfly effect: Efficient security credential management system for vehicular communications," in *IEEE Vehicular Networking Conference (VNC'18)*, 2018.

[32] Q. Dang, *SP 800-107 Rev1 – Recommendation for Applications Using Approved Hash Algorithms*, National Institute of Standards and Technology (NIST), Gaithersburg, MD, USA, Aug 2012.

[33] L. Davis, "Modifications of the optimal velocity traffic model to include delay due to driver reaction time," *Physica A: Statistical Mechanics and its Applications*, vol. 319, pp. 557–567, 2003.

[34] D. LeBlanc, S. Bogard, and R. Goodsell, "Connected commercial vehicles: Retrofit safety device kit project. Model deployment operational analysis report," US Dept. of Transportation, Tech. Rep., Mar 2014.

[35] S. Carpenter and M. Sichitiu, "Analysis of packet loss in a large-scale DSRC field operational test," in *Int. Conf. on Performance Evaluation and Modeling in Wired & Wireless Networks (PEMWN)*, 2016, pp. 1–6.

[36] ETSI, "TR 103 415 V1.1.1 intelligent transport systems (its);security; pre-standardization study on pseudonym change management," European Telecom Standards Institute, Tech. Rep., Apr 2018.

[37] SAE, "J2945/1: On-board system requirements for V2V safety communications," SAE International, Tech. Rep., Mar 2016.

[38] E. Cominetti, M. Silva, and M. Simplicio, "BSM chaining simulator," Code Ocean: https://doi.org/10.24433/CO.9668891.v3, 2021.

[39] "SUMO user documentation: German city Eichstätt," sumo.dlr.de/docs/Tutorials/OSMActivityGen/eichstaett.net.xml.html, 2021.

[40] A. Ghosal and M. Conti, "Security issues and challenges in V2X: A survey," *Computer Networks*, vol. 169, p. 107093, 2020.

[41] M. Simplicio, E. Cominetti, H. Kupwade-Patil, J. Ricardini, and M. Silva, "ACPC: Efficient revocation of pseudonym certificates using activation codes," *Ad Hoc Networks*, p. (in press), 2018.

[42] E. Verheul, C. Hicks, and F. Garcia, "IFAL: Issue first activate later certificates for V2X," in *IEEE European Symposium on Security and Privacy (EuroSP)*, 2019, pp. 279–293.

[43] M. Khodaei and P. Papadimitratos, "Scalable resilient vehicle-centric certificate revocation list distribution in vehicular communication systems," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2020.

[44] M. Simplicio, E. Cominetti, H. Kupwade-Patil, J. Ricardini, and M. Silva, "Revocation in vehicular public key infrastructures: Balancing privacy and efficiency," *Veh. Commun.*, vol. 28, p. 100309, 2021.

[45] S. Lee, H. Seo, B. Chunng, J. Choi, H. Kwon, and H. Yoon, "OpenCL based implementation of ECDSA signature verification for V2X communication," in *Advanced Multimedia and Ubiquitous Engineering*. Springer Singapore, 2019, pp. 711–716.

[46] H. Kwon and M. Lee, "Fast verification of signatures with shared ECQV implicit certificates," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 4680–4694, 2019.

[47] P. Bottinelli and R. Lambert, "Accelerating V2X cryptography through batch operations," Cryptology ePrint Archive, Report 2019/887, 2019.

[48] S. Karati, A. Das, D. Roychowdhury, B. Bellur, D. Bhattacharya, and A. Iyer, "Batch verification of ECDSA signatures," in *International Conference on Cryptology in Africa*. Springer, 2012, pp. 1–18.

[49] A. Antipa, D. Brown, R. Gallant, R. Lambert, R. Struik, and S. Vanstone, "Accelerated verification of ECDSA signatures," in *Selected Areas in Cryptography*. Springer, 2006, pp. 307–318.

[50] J. Cheon and J. Yi, "Fast batch verification of multiple signatures," in *Public Key Cryptography*. Springer, 2007, pp. 442–457.

[51] D. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang, "High-speed high-security signatures," Cryptology ePrint Archive, Report 2011/368, 2011.