Influence of Faulty Signatures in Batch Verification in VANET

Sujash Naskar*, Carlo Brunetta †, Gerhard Hancke‡, Tingting Zhang*, and Mikael Gidlund*
*Department of Information Systems and Technology, Mid Sweden University, Sweden
† Independent Researcher, was with Simula UiB, Bergen, Norway
‡Department of Computer Science, City University of Hong Kong, Hong Kong
Email: *{firstname.lastname@miun.se}, †{brunocarletta@gmail.com}, ‡{gp.hancke@cityu.edu.hk}

***Abstract*—Vehicular Ad-Hoc Networks (VANETs) enable vehicles to share critical data for safety and traffic management. To improve efficiency, batch verification is used to authenticate multiple vehicle-to-vehicle (V2V) messages at once. However, proposed solutions avoid error-prone environments with faulty signatures because of the higher analytical complexity, thus considering only error-free scenarios. This paper considers the error-prone scenario and proposes a novel strategy that allows optimal aggregation computations and reuse of such pre-computations to minimize the computational cost of identifying the faulty signature in a batch. Our analysis shows that batch verification outperforms standard methods when the error rate is below 40%, with advantages up to 63% in typical scenarios. We provide guidelines for when batch verification is more efficient and suggest improvements to further optimize its performance in VANETs, offering a practical solution for real-world applications.**

***Index Terms*—Batch Verification, Optimized Authentication, Invalid Signatures, ECDSA, Vehicular Ad-Hoc Networks (VANETs)**

I. Introduction

The rising demand for safer and more efficient transportation systems has accelerated the adoption of Vehicular Ad-Hoc Networks (VANETs) technologies. The main objective of these technologies is to facilitate direct communication between vehicles, enabling them to share crucial driving data that enhances the overall driving experience while ensuring greater passenger safety and comfort. Often referred to as vehicle-to-vehicle (V2V) communication, this allows vehicles to broadcast messages that typically include information such as speed, direction, route mapping, braking actions, alerts, and emergency signals directly to surrounding vehicles, fostering a more connected and responsive transportation network. However, since V2V communication occurs over public channels, it is crucial to authenticate the sender of each message before it is accepted by the receiving vehicles. This authentication process is essential to prevent unauthorized entities or adversaries from sending malicious messages that could lead to traffic hazards, accidents, or even fatalities. Therefore, verifying the authenticity of each V2V message is mandatory for ensuring passenger safety in VANETs.

A critical challenge in this context is the time-sensitive nature of message verification. Existing research indicates that V2V messages must be authenticated within approximately $300\mathsf{ms}$ [1]; otherwise, the messages become invalid and are discarded. This time constraint, known as the lifespan of a V2V message, presents significant challenges, especially in scenarios with a high density of vehicles, where each vehicle is broadcasting numerous driving-assisted messages. In such cases, verifying every individual message within such a short time frame becomes impractical using standard single-message verification methods. To address this issue, researchers have proposed batch verification methodologies, which allow a set of messages, or a *batch*, to be verified in a single operation. This approach enables the receiving vehicle to authenticate numerous V2V messages before they expire, offering a much more efficient solution than verifying each message individually (Figure 1).

However, most existing batch primitives verify the whole batch meaning that a single error can cause the verification to fail. This property is impractical for the VANET applications since message signatures transmitted over public channels might be corrupted or maliciously altered; thus, the application would require the algorithm to pinpoint the exact locations of faulty signatures if a fails verification. As a result, the impact of corrupted or faulty message signatures on the computational efficiency of batch verification strategies remains unclear, especially in understanding how many messages a vehicle can verify with the $300\mathsf{ms}$ threshold in the presence of invalid signatures. Therefore, the unexplored research question is, if the batch of L signatures has f invalid ones that the algorithm must correctly identify, is the batch verification still more efficient? This research gap, where the impact of faulty signatures on the effectiveness of batch verification remains unexplored, serves as the motivation for our study.

We consider the batch verification primitive ($\mathsf{ECDSA}*$) proposed by Kittur and Pais [2] and designed as a secure variant of the standard elliptic curve digital signatures. We propose a novel approach defined by an optimal aggregation procedure together with a verification strategy. Briefly, the aggregation updates sequentially with each new signature and stores specific pre-computed aggregation which are later used in the verification phase to minimize the number of computation and allowing the complete identification of all the bad signatures.

Beyond analyzing the strategies complexity, we rigorously

This work was part of the project "Next Generation Industrial IoT (NIIT)," funded by the Swedish Knowledge Foundation (KKS).

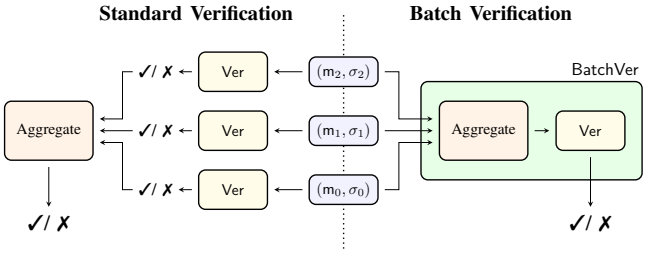

Fig. 1: Comparison of single and batch verification strategies

evaluate the impact of faulty signatures on verification efficiency and demonstrate through empirical analysis that our approach allows a simplified analysis of batch efficiency in the presence of errors and how batch improves the verification computation time under various error scenarios, offering insights that are absent in the literature.

A. Contributions

Our main research contributions are as follows:

- **Optimized** ECDSA$*$ **Batch Verification:** We introduce a novel strategy for optimizing the aggregation and verification of batch signatures in ECDSA$*$, which significantly reduces computational overhead. Our method not only minimizes the required computations but also provides precise identification of faulty signatures within the batch, enhancing the robustness of the verification process.
- **Comprehensive Empirical Efficiency Analysis:** We conduct extensive simulations to evaluate the performance of both batch and standard verification algorithms under various error probabilities. The resulting data offers detailed insights into computational costs across different scenarios. Additionally, we derive specific relationships between error rates, batch sizes, and input bandwidth for a fixed time threshold of 300ms, illustrating the practical impact of these parameters on the number of verifiable signatures.

B. Related Works

A common cryptographic approach to achieve batch verification of message signatures is the use of bilinear pairing [3], defined as $e : G_1 \times G_2 \rightarrow G_T$, where G_T is the target group for two elliptic curve group G_1, G_2. This strategy is vastly used in literature by Bagga et al. [4], Maurya et al. [5], Liu et al. [6], Feng et al. [7]. However, if one or more signatures are invalid, the entire batch gets rejected. None of the state-of-the-art schemes has considered the presence of faulty message signatures, and therefore, it is unclear what happens when a batch fails verification. Also, pairing-based operations are computationally expensive [8], which makes them inefficient for verifying a significantly large number of V2V messages [9] within a given 300ms time threshold.

Another popular approach is to use ECDSA$*$ signature verification scheme modified to perform batch verification. As proposed by Kittur and Pais [2]. ECDSA$*$ signatures are more computationally lightweight than pairing-based signatures [8], they are highly efficient for verifying numerous V2V messages. The computation cost for an ECDSA$*$ batch verification is the same as computing a single standard verification plus some extra computation for the arithmetic aggregation of signatures that depends on the batch size. This strategy is adapted in several contribution by Lin et al. [10], Yan et al. [9], Zhang et al. [11], Dwivedi et al. [12]. However, both the extra arithmetic computations and the number of standard verifications increase in the presence of error, motivating the community's doubt about its efficiency when introducing faulty signatures. Also, the general ECDSA$*$ batch verification techniques are vulnerable to signature forging attacks that allow an adversary to forge malicious message signatures to get verified as valid.

C. Paper Organization

The paper is organized as follows: Section II covers ECDSA$*$ batch verification preliminaries. Section III introduces the proposed recursive batch verification algorithm with optimizations. Section IV evaluates its efficiency in various error scenarios against single verification. Finally, Section V provides conclusions.

II. Preliminaries

ECDSA$*$ *Signatures:* The ECDSA$*$ algorithm is a variation of the Elliptic Curve Digital Signature Algorithm (ECDSA$*$) [13]. Briefly, after initializing and generating the appropriate key pairs, the primitive computes a signature σ on a message m using the $\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, \mathsf{m})$ algorithm. Unlike ECDSA, where a signature consists of a coefficient and the x-coordinate of an elliptic point, ECDSA$*$ signatures consist of an elliptic curve point and a finite field coefficient. Standard single verification of a signature is performed via the $\mathsf{Ver}(\mathsf{pp}, \mathsf{pk}, (R, c), \mathsf{m})$ algorithm. We consider the ECDSA$*$ batch verification primitive (Figure 2) proposed by Kittur and Pais [2] which introduces additional pairwise co-primes coefficient b_i which guarantees the unforgeability of the signature scheme.

VANET: We assume a public key infrastructure-based authentication scenario in VANET following the system model proposed by Naskar et al. [8], where registered vehicles broadcast signed messages using public parameters (pp) provided by trusted certification authorities or CAs. Signed messages can then be verified using their corresponding public key and the public parameters. As presented in Figure 2, vehicle perform the $\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, \mathsf{m})$ algorithm to sign a V2V message and broadcast it to public channel. A receiver vehicle uses the $\mathsf{Ver}(\mathsf{pp}, \mathsf{pk}, \sigma, \mathsf{m})$ to perform single verification or uses $\mathsf{BatchVer}((\mathsf{pp}, b_I), \mathsf{pk}, \sigma_I, \mathsf{m}_I)$ to batch verify multiple received messages.

III. Proposed Batch Verification and Strategy

Optimizing execution time is essential for the VANET application to speed up the verification process. For this reason, we provide a comprehensive description of strategies

Init

$\mathsf{pp} \leftarrow (\mathbb{G}, p, G)$

return pp

KGen(pp)

$r \leftarrow\$ \mathbb{Z}_p$

$(\mathsf{sk}, \mathsf{pk}) \leftarrow (r, r \cdot \mathsf{pp}.G)$

return $(\mathsf{sk}, \mathsf{pk})$

Sign($\mathsf{pp}, \mathsf{sk}, \mathsf{m}$)

$r \leftarrow\$ \{2, \ldots, p-2\}$

$R \leftarrow r \cdot \mathsf{pp}.G$

$x \leftarrow x(R)$

$c \leftarrow nr^{-1}(\mathsf{H}(\mathsf{m}) + \mathsf{sk} \cdot x)$

$\sigma \leftarrow (R, c)$

return σ

Ver($\mathsf{pp}, \mathsf{pk}, \sigma, \mathsf{m}$)

$x \leftarrow x(R)$

$w \leftarrow c^{-1}$

$u \leftarrow \mathsf{H}(\mathsf{m}) \cdot w$

$v \leftarrow x \cdot w$

$\overline{R} \leftarrow u \cdot \mathsf{pp}.G + v \cdot \mathsf{pk}$

return $x(\overline{R}) \stackrel{?}{=} x$

BatchVer($(\mathsf{pp}, b_I), \mathsf{pk}, \sigma_I, \mathsf{m}_I$)

for $i \in I$ **do**

$x_i \leftarrow x(R_i)$

$w_i \leftarrow c_i^{-1}$

$u_i \leftarrow \mathsf{H}(\mathsf{m}_i) \cdot w_i$

$v_i \leftarrow x_i \cdot w_i$

$\overline{R} \leftarrow \sum_{i \in I} b_i \cdot R_i$

$\overline{U} \leftarrow \left(\sum_{i \in I} b_i u_i \right) \cdot \mathsf{pp}.G$

$\overline{V} \leftarrow \left(\sum_{i \in I} b_i v_i \right) \cdot \mathsf{pk}$

return $\overline{R} \stackrel{?}{=} \overline{U} + \overline{V}$

Fig. 2: Algorithms for ECDSA$*$ signature scheme with both standard and batch verification algorithms.

and algorithms that optimally minimize the batch verification execution time at any vehicle.

We split the batch verification into three phases: a preparatory phase where message-signatures are manipulated to simplify the computations in next phases; an aggregation phase where the prepared content is aggregated into a singular element; and a verification phase where the aggregated element is verified. We observe that the batch verification (Figure 2) has a linear system structure $\overline{R} \stackrel{?}{=} \overline{U} + \overline{V}$, meaning that the sum can be partitioned into two smaller sets $I = I_1 \cup I_2$, formally,

$$\sum_{i \in I} (\cdots) = \sum_{i \in I_1} (\cdots) + \sum_{i \in I_2} (\cdots)$$

which corresponds to two independent batch verification where the verification for I_2 is obtained from the subtraction between the other verifications. This property allow us to store pre-computed aggregated values which are used to identify bad signatures during the batch verification.

Consider the timing to execute a finite field sum/product as $\mathcal{T}_+, \mathcal{T}_\times$ and, similarly, an elliptic curve sum/ product as $\mathcal{T}_{\mathbb{E}+}, \mathcal{T}_{\mathbb{E}\times}$. We denote $\mathcal{M}_\mathbb{F}, \mathcal{M}_\mathbb{E}$ the space necessary to store a field or curve element, respectively while $\mathcal{T}_\mathsf{h}$ indicates the timing for computing a hash. We assume that testing the equality between two elements will have zero cost as it takes negligible computation cost, while subtractions/inversions will have an equivalent computation cost, such as additions/multiplications, as they require nearly same execution times. This section considers a batch size of any $L \in \mathbb{N}$, provides a thorough analysis of the costs for each phase, both for standard and batch verification. Finally, we analyse the verification phase cost in the presence of errors.

Sequential aggregation $\Psi_{i,S}(x)$

$i \leftarrow i + 1$

$i = \sum_j 2^j \cdot i_j, i_j \in \{0, 1\}$

$m = \min_j \{i_j \neq 0\}$

$t \leftarrow x$

for $j \in [0, m-1]$ **do**

$t \leftarrow t + S[i + 1 - 2^j]$

S.**append** (t)

Final aggregation $\Psi(i, S)$

$i = \sum_{j=0}^{m} 2^j \cdot i_j, i_j \in \{0, 1\}$

$\widehat{\sigma_I} \leftarrow S[i-1]$

for $j \in [0, m-1]$ **do**

if $i_j = 1$ **then**

$l \leftarrow \sum_{l=0}^{j} 2^l \cdot i_l$

$\widehat{\sigma_I} \leftarrow \sigma_I + S[i-1-l]$

return $\widehat{\sigma_I}$

Fig. 3: Sequential aggregation strategy.

A. Preparatory Phase

During the preparatory phase, both single and batch verification algorithms must compute (u_i, v_i) from any message-signature pair received and store such values in memory later used for verification. The total of the costs is

$$L \cdot (\mathcal{T}_\mathsf{h} + 3\mathcal{T}_\times) \tag{1}$$

Additionally, the batch verification must multiply each (R_i, u_i, v_i) by b_i with bit-length $b \ll 256$. Due to the significant difference in size, we consider a naive implementation of the double-and-add algorithm for elliptic curve scalar multiplication, with timing $\mathcal{T}_{\mathbb{E}\times_\mathsf{b}}$ for a scalar of length b bits. We obtain a total additional preparatory cost of:

$$L \cdot (2 \cdot \mathcal{T}_\times + \mathcal{T}_{\mathbb{E}\times_\mathsf{b}}) \leq L \cdot (2 \cdot \mathcal{T}_\times + 2 \cdot b \cdot \mathcal{T}_{\mathbb{E}+}) \tag{2}$$

Observe, if $L = 1$, no b_i multiplication is necessary.

B. Aggregation Phase

The aggregation phase computes the sum of the L tuples $(b_i R_i, b_i u_i, b_i v_i)$ for a total cost of:

$$(L-1) \cdot (2 \cdot \mathcal{T}_+ + \mathcal{T}_{\mathbb{E}+}) \tag{3}$$

To minimize the device's computational overload, the aggregation is executed as soon as each signature is prepared sequentially. The specific pre-computed aggregations are stored into the list S that will contain L partial aggregations used later by the verification strategy if an error occurs with a memory-cost of $L \cdot (2\mathcal{M}_\mathbb{F} + \mathcal{M}_\mathbb{E})$. We consider the strategy of Figure 3 instantiated with an empty list S and counting index $i = 0$. For every new prepared signature x, the device executes $\Psi_{i,S}(x)$ which updates both the counting index and adds an entry to S. When the aggregation is concluded, the final aggregation $\Psi(i, S)$ is executed which produces the total aggregated value $\widehat{\sigma_I} = (\sum_I b_i R_i, \sum_I b_i u_i, \sum_I b_i v_i)$. The stored values are the left branches (I_1) of all the possible verifications.

C. Verification Phase

For the standard verification, the total cost is trivial to compute and is not affected by errors. For each of the L signature, the algorithm checks if $R \stackrel{?}{=} u_i \cdot G + v_i \cdot \mathsf{pk}$ meaning,

$$L \cdot (2 \cdot \mathcal{T}_{\mathbb{E}\times} + \mathcal{T}_{\mathbb{E}+}) \tag{4}$$

Strategy $\Phi(I, C)$

if $C = 0$ **then return** I

if $|I| = 1$ **then return** $\{\}$

$I_1 \cup I_2 \leftarrow I \ \wedge \ |I_1| = \max_{2^j}\{2^j < |I|\}$

$C_1 \leftarrow \mathsf{BatchVer}((\mathsf{pp}, b_{I_1}), \mathsf{pk}, \sigma_{I_1}, \mathsf{m}_{I_1})$

$C_2 \leftarrow C - C_1$

return $\Phi(I_1, C_1) \cup \Phi(I_2, C_2)$

Fig. 4: Verification strategy, recursive algorithm.

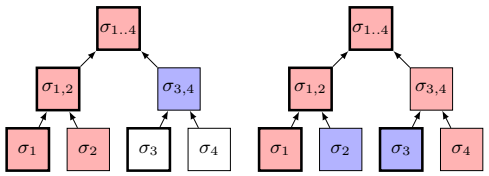

Fig. 5: Influence of two invalid signatures and the amount of verifications required. Nodes with thicker border are the computed verifications, red nodes have $C_i \neq 0$, blue nodes have $C_i = 0$ and white nodes are left unused by the strategy.

For batch verification, the total cost depends on the presence and positioning of errors. We consider the strategy Φ (Figure 4) instantiated with all the indices $I = \{1, \ldots, L\}$ and $C = \mathsf{BatchVer}((\mathsf{pp}, b_I), \mathsf{pk}, \sigma_I, \mathsf{m}_I)$ where $C = \overline{R} - \overline{U} - \overline{V}$ and verification is done using the precomputed $\widehat{\sigma_I}$.

Briefly, if the current verification is valid, the strategy returns all the current indices. Otherwise, if $C \neq 0$ then there are errors thus the partitioning plus the verification on I_1 obtaining C_1 used to compute C_2 via subtraction and not by executing a verification. The I_1's precomputed inputs are stored into S thus each verification costs like a single standard verification (Equation (4)). The strategy splits the problem into two smaller ones and correctness is maintained thanks to linearity.

The invalid signatures' positioning influences the number of verification required by the strategy, an example with four signatures is highlighted in Figure 5 where the errors are either contiguous (σ_1, σ_2) or not (σ_1, σ_4).

The strategy executes one standard verification on the total aggregated value. If errors are present, the cost depends on the number of additional single verification required. Since our strategy is deterministic, a best and worst scenario can be evaluated, e.g. our strategy implies that the best scenario happens whenever all the errors are contiguous and at the end of the batch while the worst scenario happens whenever the errors maximise the number of subtrees affected. We denote with c_f such a total amount which includes the first one too. Furthermore, for each additional verification, the strategy computes a standard verification and a curve subtraction between the verification results for a total of:

$$c_f \cdot (2 \cdot \mathcal{T}_{\mathbb{E}\times} + \mathcal{T}_{\mathbb{E}+}) + (c_f - 1) \cdot \mathcal{T}_{\mathbb{E}+} \tag{5}$$

TABLE I: Execution times of cryptographic operations.

Symbol	Description	$\approx$ Timing (ms)
$\mathcal{T}_{+}$	Addition of two field elements	0.0016
$\mathcal{T}_{\times}$	Multiplication of two field elements	0.0017
$\mathcal{T}_{\mathsf{exp}}$	Exponentiation operation	0.0321
$\mathcal{T}_{\mathsf{E}+}$	Point addition on secp256k1	0.0044
$\mathcal{T}_{\mathsf{E}\times}$	Point multiplication on secp256k1	0.7439
$\mathcal{T}_{\mathsf{h}}$	Hash operation	0.0048

IV. Empirical Evaluation

We adopt a methodology used by several prior studies [14, 15, 16, 17] for the computational comparison between algorithms. The approach analytically evaluates the costs of the algorithm and later estimates an execution timing using the empirical timing for basic operations. The basic operation execution timings are reported in Table I. The execution timings are obtained using an Intel i7-6500U @ 2.50GHz CPU, 16GB of RAM. The cryptographic primitives are implemented in C using the OpenSSL library [18] on a Linux virtual environment. We use the secp256k1 curve [19] which is defined on a 256-bit prime field with a 256-bit order.

The VANET application requires messages to be verified with a maximum delay of 300ms. We consider such timing as a threshold and want to answer the question: *how many messages can be verified within the threshold in the presence of invalid signatures?* We consider the device to have signatures in input measured in signature per second (sps) which can be identified as input bandwidth and with an error-rate of ϵ. In the evaluation, we assume the two following scenarios: i) error-free and ii) with errors.

A. Error-free Scenario

Whenever all signatures are valid, the computational cost for both verifications can be easily computed by considering the sum of the correspondent preparatory, aggregation, and verification phases. The cost for standard verification is the sum of Equations (1) and (4) while for batch verification is the sum of Equations (1) to (3) and (5) with $c_f = 1$. We provide the timing for different signature amounts in Figure 6.

In scenarios where batches contain only error-free messages, batch verification significantly outperforms standard single-message verification, processing nearly $\times 10$ as many messages, as illustrated in Figure 6. The figure also provides a detailed breakdown of the computational costs for different phases of batch verification highlighting how the batch preparatory phase is the most expensive because of all the scalar multiplication by b_i. The massive gain is the consequence of the usage of the naive double-and-add multiplication algorithm instead of the generic one because of the substantially lower timing $\mathcal{T}_{\mathbb{E}\times_{\mathsf{b}}} < \mathcal{T}_{\mathbb{E}\times}$. The limit scenario where the generic algorithm is used (i.e. $\mathcal{T}_{\mathbb{E}\times_{\mathsf{b}}} = \mathcal{T}_{\mathbb{E}\times}$), the gain is reduced to $\times 2$.

B. Error Scenario

We define the error-rate ϵ as the percentage of bad signatures in the batch of size L which will force a verification fail for

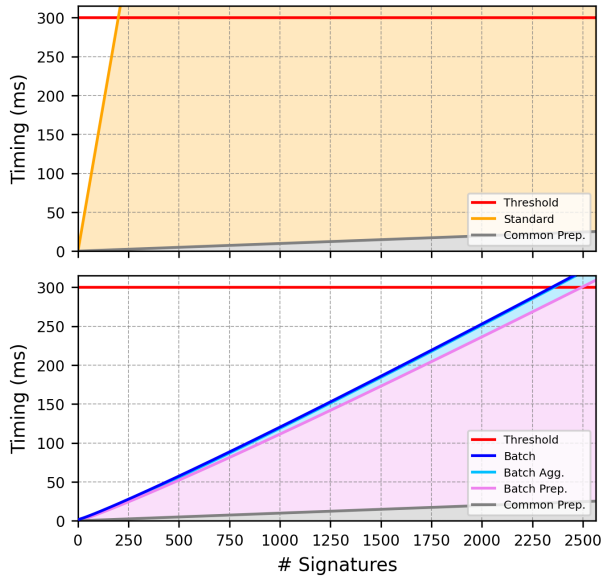

Fig. 6: Error-free timing of the single standard (first figure) and the batch verification (second figure). All the computational phases are highlighted.

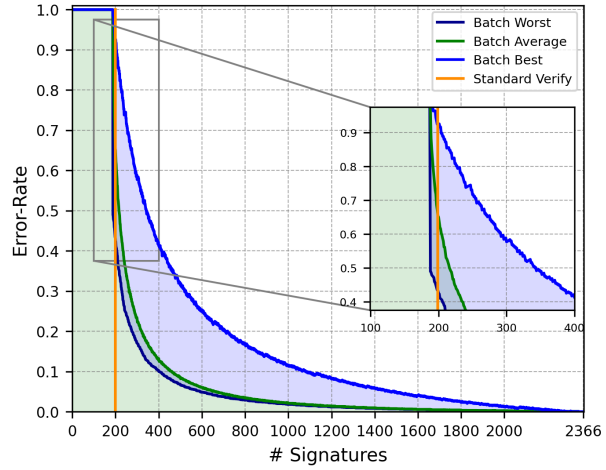

Fig. 7: Average error-rate achieving in 300ms of verification time for a specific number of signatures.

any pre-computed aggregation in which these bad signatures are present. This means that increasing the error-rate ϵ implies an increase in the verification's amount c_f which variates between the best and worst scenarios. We simulate different uniform random error-positions to obtain an average scenario representing the most likely scenario in practice. We plot in Figure 7 the error-rate achievable by the batch verification in the different scenarios for different batch sizes with a highlight on the intersection of such curves and the standard verification one.

When considering the presence of errors, batch verification time is influenced by the error rate ϵ and how errors are distributed in the batch. Therefore, it's important to analyze achievable error rates for the 300ms threshold compared to

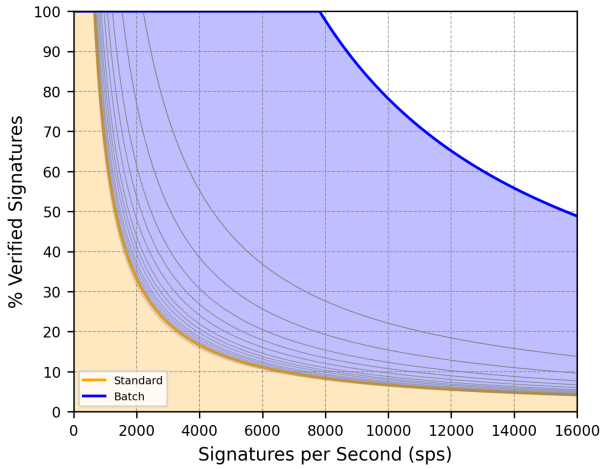

Fig. 8: Bandwidth of the verification algorithms. Gray lines represent batch verification with error-rates at steps of 5%.

the maximum signatures the standard algorithm can verify. As highlighted in Figure 7, the worst error-rate achievable is $\sim 42\%$ meaning that for lower error rates and the same batch size, the batch verification will **always** be faster than the standard one. The best error-rate achievable is $\sim 91\%$ when errors are contiguously located at the end of the batch. However, the best case represents a borderline case and, therefore, does not reflect a real application scenario. From the simulation, the average error-rate achieved for random positioning is $\sim 62\%$, at which point the average case is equivalent to the standard verification. The average case represents a real application scenario where errors are averagely distributed.

C. Bandwidth Analysis

As our last question, we are interested in analysing the verification bandwidth, i.e. the percentage of verifiable signatures for a stream of signatures. We plot in Figure 8 the achievable band for the standard verification and the batch one with additional highlights for increasing error rates.

As displayed in Figure 8, the batch verification with average error positioning allows a higher bandwidth than the standard algorithm. Increasing error-rates are displayed too highlighting how increasing error-rates reduce the verification bandwidth.

During our simulations, we observe a bursting effect appearing whenever increasing the bandwidth over the verification algorithm's limit. For a period of time, the scheme absorbs the higher workload and accumulates delay which increases until the 300ms verification threshold. At that point, the algorithm must discard expired signatures and, intuitively, recovers time from the delay and can verify again. This behaviour can be utilized in applications to control the signatures in input's bandwidth and anticipate when the device will drop expired signatures.

D. Further Improvements and Observations

While our strategy makes the batch verification process highly efficient, there are still opportunities to enhance compu-

tational cost and overall efficiency, pointing to potential future work.

- The $\{b_i\}$ values must be sampled and applied randomly to guarantee security, i.e. the malicious signer should be unaware of which values are used and in which order. For the VANET application, we suggest the vehicle to freshly sample such values when booting-up the system and securely store them until the preparatory phase is executed.
- The $\{b_i\}$ values of proportional to the batch size and the double-and-add scalar multiplication algorithm is used ensuring high efficiency. However, depending on the specific use case, $\{b_i\}$ can be set to a fixed size b and a more efficient b-long scalar-multiplication algorithm can be used, desiredably secure against side-channel attacks.
- Error-rate, input bandwidth and accumulated delay can be monitored and should be used to act accordingly. For example, there can be a fixed acceptable error-rate ϵ in VANET and, whenever a receiver notices a higher error-rate in the signatures of a sender, such behaviour is reported to the trust authorities, which can check the legitimacy of the sender. Also, whenever the receiver notices that the accumulated delay would push expire many signatures, the receiver can focus only on higher-priority messages until the bandwidth returns to achievable levels.

V. Conclusions

This study not only addresses a critical gap in understanding batch verification efficiency with faulty signatures in the existing literature, but also provides practical insights for enhancing batch verification efficiency of V2V communication systems. Our analysis highlights that in most likable scenario represented as the average case in the analysis that closely aligns with real-world conditions, batch verification consistently outperforms standard verification, maintaining high efficiency even with an error rate of up to approximately $\sim 62\%$. Even in the unlikely worst-case scenario, batch verification demonstrates robust performance, sustaining efficiency with error rates as high as $\sim 42\%$. The strategy applied for the high-efficiency gain in batch computation time also ensures protection against signature forging, ensuring the integrity of V2V communications. For future work, sampling of the coefficient b_i and its size can be pre-determined according to specific use case that will allow extending the application of the batch verification strategy to other domains.

References

[1] M. Arif, G. Wang, M. Z. A. Bhuiyan, T. Wang, and J. Chen, "A Survey on Security Attacks in VANETs: Communication, Applications and Challenges," *Vehicular Communications*, vol. 19, p. 100179, 2019.

[2] A. S. Kittur and A. R. Pais, "A New Batch Verification Scheme for ECDSA* Signatures," *Sādhanā*, vol. 44, no. 7, p. 157, 2019.

[3] S.-F. Tzeng, S.-J. Horng, T. Li, X. Wang, P.-H. Huang, and M. K. Khan, "Enhancing security and privacy for identity-based batch verification scheme in vanets," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 4, pp. 3235–3248, 2015.

[4] P. Bagga, A. K. Sutrala, A. K. Das, and P. Vijayakumar, "Blockchain-based batch authentication protocol for internet of vehicles," *Journal of Systems Architecture*, vol. 113, p. 101877, 2021.

[5] C. Maurya, V. Chaurasiya, and Kumar, "Efficient anonymous batch authentication scheme with conditional privacy in the internet of vehicles (iov) applications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 9, pp. 9670–9683, 2023.

[6] J. Liu, C. Peng, R. Sun, L. Liu, N. Zhang, S. Dustdar, and V. C. Leung, "Cpahp: Conditional privacy-preserving authentication scheme with hierarchical pseudonym for 5g-enabled iov," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 7, pp. 8929–8940, 2023.

[7] X. Feng, Q. Shi, Q. Xie, and L. Wang, "P2BA: A Privacy-Preserving Protocol with Batch Authentication Against Semi-Trusted RSUs in Vehicular Ad Hoc Networks," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 3888–3899, 2021.

[8] S. Naskar, C. Brunetta, G. Hancke, T. Zhang, and M. Gidlund, "A Scheme for Distributed Vehicle Authentication and Revocation in Decentralized VANETs," *IEEE Access*, 2024.

[9] C. Yan, C. Wang, J. Shen, K. Dev, M. Guizani, and W. Wang, "Edge-Assisted Hierarchical Batch Authentication Scheme for VANETs," *IEEE Transactions on Vehicular Technology*, 2023.

[10] C. Lin, D. He, X. Huang, N. Kumar, and K.-K. R. Choo, "Bcppa: A blockchain-based conditional privacy-preserving authentication protocol for vehicular ad hoc networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 12, pp. 7408–7420, 2020.

[11] M. Zhang, J. Zhou, G. Zhang, M. Zou, and M. Chen, "Ec-baas: Elliptic curve-based batch anonymous authentication scheme for internet of vehicles," *Journal of Systems Architecture*, vol. 117, p. 102161, 2021.

[12] S. K. Dwivedi, R. Amin, S. Vollala, and A. K. Das, "Design of blockchain and ecc-based robust and efficient batch authentication protocol for vehicular ad-hoc networks," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[13] A. Antipa, D. Brown, R. Gallant, R. Lambert, R. Struik, and S. Vanstone, "Accelerated Verification of ECDSA Signatures," in *International Workshop on Selected Areas in Cryptography*. Springer, 2005, pp. 307–318.

[14] A. Yang, J. Weng, K. Yang, C. Huang, and X. Shen, "Delegating Authentication to Edge: A Decentralized Authentication Architecture for Vehicular Networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 2, pp. 1284–1298, 2020.

[15] S. Chen, Y. Liu, J. Ning, and X. Zhu, "BASRAC: An Efficient Batch Authentication Scheme with Rule-based Access Control for VANETs," *Vehicular Communications*, vol. 40, p. 100575, 2023.

[16] J. Shen, D. Liu, X. Chen, J. Li, N. Kumar, and P. Vijayakumar, "Secure Real-Time Traffic Data Aggregation with Batch Verification for Vehicular Cloud in VANETs," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 1, pp. 807–817, 2019.

[17] L. Wei, J. Cui, H. Zhong, I. Bolodurina, and L. Liu, "A Lightweight and Conditional Privacy-Preserving Authenticated Key Agreement Scheme with Multi-TA Model for Fog-based VANETs," *IEEE Transactions on Dependable and Secure Computing*, 2021.

[18] E. Käsper, "Fast Elliptic Curve Cryptography in OpenSSL," in *Financial Cryptography and Data Security: FC 2011 Workshops, RLCPS and WECSR 2011, Rodney Bay, St. Lucia, February 28-March 4, 2011, Revised Selected Papers 15*. Springer, 2012, pp. 27–39.

[19] J. W. Bos, J. A. Halderman, N. Heninger, J. Moore, M. Naehrig, and E. Wustrow, "Elliptic Curve Cryptography in Practice," in *Financial Cryptography and Data Security: 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers 18*. Springer, 2014, pp. 157–175.