Secure Cluster-based In-network Information Aggregation for Vehicular Networks

Stefan Dietzel*, Andreas Peter†, and Frank Kargl*†
*Institute of Distributed Systems, Ulm University, Germany
†Centre for Telematics and Information Technology, University of Twente, The Netherlands
E-mail: {stefan.dietzel, frank.kargl}@uni-ulm.de, a.peter@utwente.nl

***Abstract*—Vehicular ad-hoc networks are a promising research area. Besides improving safety, traffic efficiency enhancements are a major expected benefit. In this paper, we present a novel security mechanism for traffic efficiency applications that leverages on velocity-based vehicle clustering and uses *HyperLogLog* estimators to create bandwidth-efficient integrity proofs. Evaluation results show that our mechanism achieves high protection against plausible attacker models, and that it is more bandwidth efficient than a comparably secure security mechanism that does not employ clustering.**

I. Introduction

Vehicular ad-hoc networks (VANETs) aim to improve driving safety, efficiency, and comfort. Paramount to successful VANET deployment are the efficient use of communication bandwidth and the protection of information integrity in the presence of malicious entities. For safety applications, both of these factors have been successfully addressed and standardization is underway. For traffic efficiency applications, however, there is high demand for integrity protection mechanisms with low bandwidth usage. A key problem is the dissemination of navigational support information about large regions of roads to large amounts of interested vehicles. To ensure scalability, a promising approach is in-network data aggregation, surveyed in [1], that combines traffic or other reports from multiple vehicles and only disseminates semantic summaries, e. g., "there is a traffic jam in interval $[302, 408]$ m"). Because semantic summaries may combine reports from many vehicles and inform about potentially long road stretches, they are a lucrative target for insider attackers. A typical attacker goal is to create false traffic summaries to divert traffic from highways or otherwise alter navigation decisions. Such *insider* attacks can be detected through data consistency checks, for instance by correlating redundant reports about the same event to detect spurious information. Moreover, combining such checks with digital signatures protects the integrity of semantic summaries, which ultimately overcomes *outsider* attacks as well.

To avoid heavy bandwidth usage due to the transmission of many signatures in this combined approach, Picconi *et al.* [2] propose the use of interactive proofs: a sender first commits to atomic reports used to compute an aggregated summary, while a receiver randomly chooses one of these committed atomic reports for which the sender needs to present the original report including a cryptographic signature from its creator. Furthermore, to remove the inherent interactivity, it is proposed to replace the receiver's role by a trusted hardware module that runs locally in each vehicle, introducing substantial hardware and management costs. Therefore, Dietzel *et al.* [3] propose a secure aggregation mechanism that, instead of requiring trusted hardware, uses only a subset of all atomic reports as proof data. The mechanism's security overhead, however, can become large if information about long stretches of roads is aggregated. Aiming to improve bandwidth efficiency, a whole line of work [4]–[6] protects data integrity through so-called FM-sketches, which are data structures based on a bit array that allow to estimate the number of distinct items added to it. Integrity protection is implemented by having each vehicle add information to the sketch and sign the bits that change due to insertion. A downside is that the security overhead of these mechanisms is considerable if many signed 1 bits accumulate in a sketch. Instead of FM-sketches, Hsiao *et al.* [7] present a secure aggregation mechanism that uses z-smallest estimators, used to prove the number of contributors to aggregated information with little bandwidth overhead. However, their approach is limited to the verification of small-scale events, such as accidents.

In this paper, we design and analyze a novel cluster-based integrity protection mechanism suitable for in-network aggregation of traffic status and similar information. In contrast to existing approaches, we do not require expensive trusted hardware, avoid geographical dependence and the use of FM-sketches, and are not limited to small-scale events. Instead, we combine data-consistency checking within clusters with bandwidth-efficient proofs based on *HyperLogLog* estimators for inter-cluster communication. The HyperLogLog data structure further allows to improve information quality by providing duplicate detection capabilities. This way, we achieve low bandwidth usage, can deal with large-scale events (such as traffic jams) and hierarchical aggregation, which has not been considered in previous works. Finally, we substantially increase information accuracy by using HyperLogLog only to prove the number of vehicles that contribute to aggregate information rather than to represent the aggregated information itself.

II. System and Adversary Model

We consider traffic information systems to be the prime application scenario for our security mechanism. Other possible applications include parking space finders or weather information systems. The goal of a traffic information system is to provide vehicles with up-to-date information about travel speeds in different parts of the road network. Exact informa-

978-1-4799-8088-8/15/$31.00 ©2015 IEEE

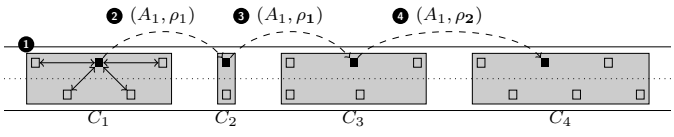

Figure 1. Overview of clustering security components.

tion is not required, but different traffic situations should be distinguishable, e. g., free-flowing traffic versus traffic jams. This is achieved on the basis of broadcasted atomic reports from each vehicle, which contain the current location, time, and velocity. To obtain scalability, we consider in-network aggregation that combines reports from multiple vehicles as they are forwarded through the network.

For information aggregation and dissemination, we take the clustering mechanism of Dietzel *et al.* [8] as a basis. The core idea is to cluster close-by vehicles according to their relative velocity to find stretches of similar speed. Using velocity, this clustering integrates well with aggregating information about stretches of homogeneous traffic. In brief, the clustering algorithm works as follows: Vehicles listen for cluster messages while they are in *undecided* state. If no messages are received within a pre-defined time period, a vehicle becomes *cluster head.* Other vehicles become *cluster members* when they receive a cluster message and their relative velocity is below a pre-defined threshold. Cluster members continuously re-evaluate their velocity threshold compliance and leave the cluster otherwise. When two clusters meet, they are joined if the cluster average velocity satisfies the velocity threshold criterion. This clustering protocol requires little additional communication overhead, yet results in stable clusters, because only vehicles with similar velocities form clusters. Since velocity information is already collected for cluster maintenance, it can easily be disseminated by cluster heads to implement a traffic information system. Cluster heads only disseminate their cluster's current average velocity. Using inter-cluster communication, cluster averages are disseminated further to inform distant vehicles.

For basic outsider attack protection, we assume a public key infrastructure is in place, which issues certificates to each registered vehicle, implementing protection against *Sybil* attacks. Each outgoing message is assumed to be signed with the sender's private key. In terms of data integrity, a vehicle's signature can be interpreted as a basic form of attestation to message correctness. When combined with aggregation, however, the integrity protection offered by signatures is limited. Each vehicle modifies messages during forwarding, invalidating the signatures.

In our *adversary model,* we assume an *insider attacker* who possesses at least one valid, certified key pair to create and sign arbitrary messages. By colluding with other attackers, multiple key pairs can be compromised and used in combination. However, to acquire a key pair, an attacker is required to have continuous physical access to the corresponding vehicle. Therefore, we assume the total number of key pairs in attacker control to be limited. In particular, the majority of vehicles is assumed to be honest in most situations, especially when the vehicle density is high. Under these assumptions, the attacker's goal is to alter the perceived traffic situation by creating messages with information that deviates from the actual traffic situation. Particularly in the cluster setting, an attacker can act as cluster head to disseminate cluster aggregates containing false traffic information. We assume our security mechanism to be *successful* if messages from attackers are correctly identified and ignored by honest vehicles.

III. Cluster-based Security Approach

Our cluster-based security mechanism is built on the assumption that clusters of vehicles serve as a trustworthy unit. That is, we assume that it is unlikely that attackers compromise the majority of vehicles within a cluster, which is a realistic assumption in our adversary model (cf. Section II). Let n be the number of vehicles that an attacker can compromise. Then clusters with at least $\tau > 2n$ members can be assumed trustworthy if majority decisions are used to agree on a cluster-wide aggregated view.

The goal of our integrity protection mechanism is to make the majority decision *verifiable within the cluster* and to create a *bandwidth-efficient proof* of the decision to be *disseminated to neighboring clusters.* Figure 1 shows an overview of the mechanism. C_1 created an aggregate A_1, which is disseminated to C_2 and then further downstream. At the same time, other clusters may create aggregates that may be merged with A_1; we omit these in the figure for clarity. The components are:

1) Vehicles in C_1 communicate their atomic observations to the cluster head who calculates an aggregated value A_1 and disseminates it within the cluster. Cluster members plausibilize the aggregated value based on overheard atomic observations from other cluster members.
2) C_1's members interactively create a proof ρ_1 that can be disseminated to neighboring vehicles along with A_1. Semantically, ρ_1 proves that at least τ vehicles have participated in a majority decision and are confident that A_1 is a correct representation of the traffic situation on C_1's covered area.
3) C_2, a cluster with size smaller than τ cannot be assumed to be trustworthy. Therefore, C_2 can only participate in forwarding information from C_1 to C_3. It does not create an own proof but forwards (A_1, ρ_1).
4) C_3, a cluster having $\geq \tau$ members, is assumed trustworthy. Hence, C_3 will not forward ρ_1 but instead create an own proof ρ_3 that replaces ρ_1. The semantic of ρ_3 is that the (honest) majority of C_3's members have verified ρ_1's correctness.

Intra-cluster agreement. Within a cluster, our security mechanism is based on data consistency mechanisms. Each cluster member i periodically broadcasts its current velocity (v_i), location (p_i), time (t_i), and a signature on these values:

$$o_i = ((p_i, t_i), v_i, \sigma_i(\cdot), PK_i, Cert_i). \tag{1}$$

The cluster head collects all observations from all n cluster members and calculates the cluster's aggregate:

$$A := (([\min_{1\leq i\leq n} p_i, \max_{1\leq i\leq n} p_i], \max_{1\leq i\leq n} t_i), \operatorname{avg}_{1\leq i\leq n} v_i). \tag{2}$$

The cluster head then broadcasts the aggregate in subsequent beacons. Each cluster member verifies whether the claimed average cluster speed deviates by at most the cluster

joining threshold from its own velocity. Also, each member verifies that the claimed aggregate time is plausible and that the own location is within the interval claimed by the aggregate. If all of these checks hold, the cluster member signs the cluster aggregate and attaches $\sigma_i(A)$ to subsequent beacons.

The cluster head collects all signatures $\sigma_i(A)$. If at least τ signatures can be collected, we can assume that A is a correct representation of the cluster situation. All cluster members can overhear the signed messages, as well. If a cluster head disseminates an aggregate A, and cluster members do not overhear at least τ signatures, they will change their state to cluster undecided (CU) and re-start the clustering process, excluding the malicious cluster head.

Inter-cluster dissemination. When a cluster aggregate A is disseminated to neighbor clusters, they cannot use data consistency alone to judge A's correctness. Instead, we create a proof that *at least* τ vehicles within the sending cluster checked its consistency. We use an estimator to prove the number of cluster members agreeing to A with a constant upper bound on proof size. Using estimators in this way was originally proposed by Hsiao *et al.* [7] to validate event reports. We extend their approach to the more dynamic setting of in-network aggregation, as well as to support hierarchical aggregation, that is, further aggregation of already aggregated information during inter-cluster dissemination.

We use the HyperLogLog data structure as estimator [9]. As an additional benefit, this estimator provides a means to detect duplicates when aggregates are merged hierarchically. The goal of applying estimators to integrity protection is to select a subset of size $s \ll n$ of all cluster member signatures to be disseminated to the neighbor cluster such that the probability that an attacker can successfully forge s signatures and still convey n total cluster members is negligible. To achieve that, we exploit that the output of a hash function that cannot be determined by the attacker is used to create the estimator data structure. And we use the hash function and the estimator data structure to select the subset s of signatures that are disseminated to the neighbor cluster.

The HyperLogLog data structure consists of $m = 2^b, b \in \mathbb{Z}_{>0}$ registers. Each register $M[i], 1 \leq i \leq m$ is initially set to 0. To add a value v, a single 32-bit hash function h can be used. The first b bits of $h(v)$ are used to determine the register $M[i]$ to use. Of the remaining bits, the length l of the initial uninterrupted sequence of 0 bits is counted, and $M[i]$ is set to $M[i] = \max(M[i], l)$. The estimated total number of distinct values is equal to the harmonic mean of each register's value:

$$E := \alpha_m m^2 \left(\sum_{1 \leq j \leq m} 2^{-M[j]} \right)^{-1}, \tag{3}$$

where α_m is a correction factor that depends on the number of registers m (cf. [9]).

To build the HyperLogLog estimator for the number of cluster participants, we add the tuple (A, PK_i) for each cluster member i to the estimator by using it as input for h. Let $\{\sigma_{\pi(1)}(A), \ldots, \sigma_{\pi(m)}(A)\}$ be the m signatures for which adding their corresponding tuple contributed the maximum register values $M[i]$. Then

$$\rho = (\sigma_{\pi(1)}(A), \ldots, \sigma_{\pi(m)}(A)) \tag{4}$$

is the proof that is communicated to the neighbor cluster. Note that the proof size is at most m signatures, independent of the cluster size. For $m = 16$ and assuming an extended version of [10] that uses aggregate signatures [11], the proof is, therefore, sufficiently small to fit into a single IEEE 802.11p transmission unit (MTU). To further reduce bandwidth use, only a subset of all m signatures can be added. Because the attacker cannot influence which signatures are selected, using only a subset does not increase the attacker's chance of success significantly (cf. Section IV).

The receiving cluster verifies that all proof signatures are valid and recreates the estimator data structure using the public keys and A. It then calculates the estimate E and verifies that $E \geq \tau$. Because the HyperLogLog data structure is determined by adding the aggregate value A and the public key corresponding to a correct signature on A, an attacker cannot inflate A. Even if the attacker would be able to choose public keys at will, she would still need to brute force the hash function to find public keys that result in desired hashes with long runs of 0 bits [7].

Hierarchical inter-cluster aggregation. So far, we discussed the exchange of aggregates between directly adjacent clusters. However, the goal of aggregation is to disseminate information farther than that. As shown in Figure 1, other clusters will disseminate not only their own aggregates but also aggregates received from other clusters. Moreover, clusters may merge their own aggregates with aggregates from other clusters and only forward the merged result. Such merging will happen if multiple clusters are part of the same traffic situation, such as a long traffic jam. To merge two aggregates A_1 and A_2, the estimator for the number of participants is updated as follows. Let M_1 be the estimator for A_1's participants and M_2 be the estimator for A_2's participants. Then

$$\forall 1 \leq i \leq m : M[i] = \max(M_1[i], M_2[i]). \tag{5}$$

The result is an estimator for the combined number of participants, eliminating duplicates [9].

The original proof information, however, becomes invalid during the merging operation, because the maximum sketch entries are changed. To create the new proof, we again exploit that a cluster with at least τ members is considered trustworthy. Suppose a cluster C_2 has merged aggregates A_1 and A_2 and wants to communicate $A_{1,2} = (A_1 \circ A_2)$ to a third cluster C_3. First, C_2's cluster head disseminates the merged aggregate $A_{1,2}$, as well as A_1 and A_2 to its cluster members. All cluster members verify that $A_{1,2}$ is a correct aggregation of A_1 and A_2 and that A_2 corresponds to the local traffic situation. Finally, the cluster members verify that A_1 was received before and that it was accompanied by a proof ρ_1. Cluster member attest to these checks by broadcasting $\sigma_i(A_{1,2})$.

The cluster head creates an estimator M using the received signatures, which attests to the number of cluster members in C_2. In addition, it creates an estimator M' by merging the estimators of A_1 and A_2 as explained above. The cluster head then forwards a proof ρ_2, as well as a difference-encoded

estimator E' for $A_{1,2}$ to C_3, where

$$\rho_2 = (\sigma_{\pi(1)}(A), \ldots, \sigma_{\pi(m)}(A)), \quad (6)$$

$$E' = (M[1] - M'[1], \ldots, M[m] - M'[m]). \quad (7)$$

Note that C_2 only proves the number of its *own* cluster members, but it additionally creates an estimator for the merged aggregate. We use the proof for verification of trustworthiness, and we use the combined estimator to detect duplicates during further hierarchical aggregation. We only use C_2's proof, because we assume that the cluster as a unit is trustworthy if it has at least τ members. Therefore, we assume that C_2's members can attest that they checked the previous cluster's proof. C_2 will only keep the old proof information if does not merge the received aggregate.

An additional benefit of the HyperLogLog estimator during hierarchical aggregation is that it helps to prevent duplicates, which – potentially biasing information – are a fundamental problem of hierarchical aggregation. Let $E(A_x \circ A_y)$ be the estimate of two aggregates' union, calculated as explained in Equation (5). When a cluster receives two aggregates A_x and A_y, which are the result of merging several cluster aggregates, it estimates the number of their intersecting vehicles by

$$I = E(A_x) + E(A_y) - E(A_x \circ A_y). \quad (8)$$

If $I > 0$, it is likely that merging A_x and A_y would introduce a bias; hence, we disseminate the aggregates separately.

Dealing with small clusters. A cluster with $< \tau$ members cannot participate in any aggregation process, since the majority of members could be controlled by an attacker. Such small clusters, hence, only forward received aggregates unmodified, including the original sending clusters' proofs. This approach may incur additional bandwidth overhead in favor of higher security. We argue, however, that in situations where clusters are small the overall vehicle density is low and additional overhead is negligible. We investigate the interrelation of vehicle density and cluster size further in Section IV.

IV. Evaluation

Our security mechanism builds on the assumption that clusters of a minimum size τ form a trustworthy unit. Once τ is reached, majority votes assure correct aggregation within a cluster, and an integrity protected HyperLogLog sketch assures the reaching of the threshold τ to other clusters (cf. Section III). The security primitives used are based on established cryptography, such as ECDSA signatures, which we assume to be secure against attacks. Rather than verifying the underlying cryptographic primitives, we therefore focus our security analysis on clustering behavior of vehicles, as well as the behavior of the sketch integrity protection.

Average cluster size. If many clusters have $< \tau$ members, our security mechanism will have a considerably higher bandwidth overhead, because those clusters do not participate in hierarchical aggregation. To assess expected cluster sizes, we implemented the clustering and security mechanisms in the JiST/SWANS network simulator, which implements IEEE 802.11p medium access and a physical layer with two-ray ground pathloss and Rayleigh fading. Figure 2 shows the average cluster size for different vehicle densities on a $5\,\mathrm{km}$ highway stretch with 3 lanes. We plot standard deviations for 10 simulation runs with randomized vehicle movement.

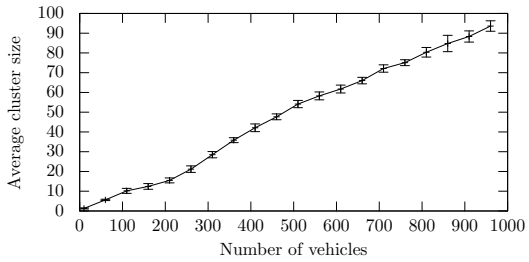

Figure 2. Average cluster sizes for different vehicle densities.

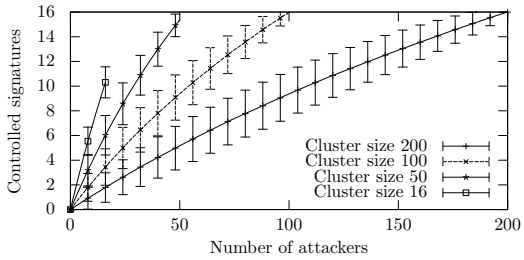

Figure 3. Number of attacker-controlled signatures in HyperLogLog sketch for different cluster sizes and numbers of attackers.

As expected, the average cluster size increases as the number of vehicles increases. In particular, the average cluster size is at least 10 vehicles for 100 or more vehicles on the $5\,\mathrm{km}$ highway. Because we assume that an attacker needs to physically compromise a vehicle to obtain key material, a threshold cluster size of $\tau = 10$ is likely to offer sufficient integrity protection. Therefore, simulation results show that the security mechanism applied to velocity-based clustering will reach the required cluster size threshold in most traffic situations. In particular, the mechanism can benefit from large cluster sizes in high traffic density settings. Note that in low traffic density scenarios both the bandwidth overhead and integrity protection are at least as good as in a setting where no clustering and aggregation are applied. The reason is that essentially all vehicles will attach signatures to messages, and signatures are kept during message forwarding in low traffic density scenarios.

Sketch manipulation. We use a HyperLogLog sketch to represent a proof of cluster size using significantly fewer total cryptographic signatures than there are cluster members. The number of signatures used depends on the parameter m of the sketch. The smallest possible value is $m = 2^4 = 16$ [9].

In Figure 3, we analyze how likely it is that an attacker can create a false proof of cluster size. Recall that for each register, the signature of the public key that contributed the maximum register value is kept. We randomly select a subset of all vehicles to be controlled by the attacker and plot the average absolute number of controlled sketch signatures. An attacker is considered successful if all signatures attached to the HyperLogLog data structure are created by vehicles under attacker control. Figure 3 shows average number of attacker-controlled signatures for 10 000 random clusters with different sizes. The graph shows that an attacker needs to control –

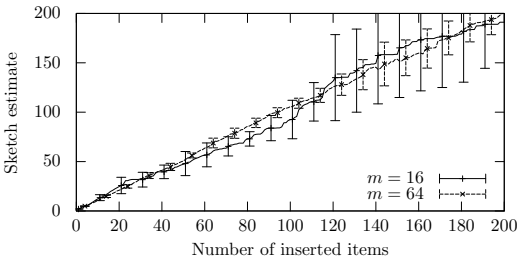

Figure 4. Average sketch estimate for different numbers of inserted items.

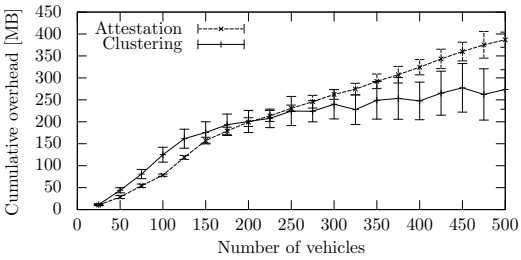

Figure 5. Overhead of our security mechanism compared to [3].

independent of the cluster size – almost all cluster members to manipulate all 16 signatures. Even if only a subset of the signatures is used for proofs, the attacker needs to control a significant portion of the cluster members; for instance, on average 50 percent to be able to generate 8 signatures.

Sketch accuracy. The HyperLogLog data structure we use as proof of cluster size is a probabilistic mechanism. Therefore, certain error margins in the estimated cluster size are to be expected. Existing works [9] analyze the behavior of HyperLogLog sketches for large counts and large values for m. Therefore, we conduct a separate study to analyze the probabilistic nature of HyperLogLog sketches in our setting.

Figure 4 shows the cluster size as estimated by the sketch (y-axis) for cluster sizes between 0 and 200 vehicles, as shown on the x-axis. The standard deviation is shown for 10 different random clusters per cluster size. For $m = 16$, the standard deviation is high especially for larger clusters. Setting $m = 64$ decreases the standard deviation significantly but results in up to 64 attached signatures. To achieve a compromise between accuracy and bandwidth overhead, we propose to combine $m = 64$ with attaching only 16 randomly selected signatures. As shown in Section IV-B, attaching only a fraction of the signatures does not harm the security properties significantly. The remaining error margin can be compensated by increasing the cluster size threshold τ accordingly. For instance, if $\tau = 10$ is the desired value and the sketch error is 5, setting $\tau = 15$ will compensate the error.

Bandwidth overhead. To assess our mechanism's bandwidth overhead, we compare its cumulative overhead in comparison with the mechanism presented in [3] (cf. Section I). Figure 5 shows the simulation results. The x-axis shows different vehicle densities; the y-axis shows the total security overhead cumulated over the whole simulation. Standard deviations are shown for 10 random vehicle distributions. For vehicle densities lower than 200 vehicles, both mechanisms perform almost identically. Our clustering-based mechanism incurs slightly higher overhead because more clusters fail to reach the required threshold of members. For higher vehicle densities, however, the clustering-based mechanism uses significantly less bandwidth. In particular, the other mechanism's continues to grow linearly in the number of vehicles, whereas the clustering mechanism's overhead stays almost constant.

V. Conclusion

We have presented a bandwidth-efficient integrity protection mechanism for in-network information aggregation in VANETs. As underlying communication primitive, we use a clustering mechanism with the distinguishing feature that it uses velocity rather than geographic position to achieve stable clusters.

Regarding clusters as a trustworthy unit, we achieve integrity protection using constant security overhead. In particular, we use HyperLogLog estimators to which we add integrity protection for bandwidth-efficient proofs of information correctness. As an additional benefit these estimators help to detect duplicate information during hierarchical aggregation. Extensive simulation results show that our mechanism is efficiently applicable to a wide range of traffic scenarios. Moreover, we show that the HyperLogLog estimator provides sufficient security and accuracy with limited bandwidth overhead.

Acknowledgements

Andreas Peter is supported by the Netherlands Organisation for Scientific Research (NWO) in the context of the CRIPTIM project.

References

[1] S. Dietzel, J. Petit, F. Kargl, and B. Scheuermann, "In-network aggregation for vehicular ad hoc networks," *IEEE Communications Surveys Tutorials*, 2014. DOI: 10.1109/COMST.2014.2320091.

[2] F. Picconi, N. Ravi, M. Gruteser, and L. Iftode, "Probabilistic validation of aggregated data in vehicular ad-hoc networks," in *Proc. ACM VANET*, 2006, p. 76. DOI: 10.1145/1161064.1161077.

[3] S. Dietzel, E. Schoch, B. Könings, M. Weber, and F. Kargl, "Resilient secure aggregation for vehicular networks," *IEEE Network*, vol. 24, no. 1, pp. 26–31, 2010. DOI: 10.1109/MNET.2010.5395780.

[4] M. Garofalakis, J. M. Hellerstein, and P. Maniatis, "Proof sketches: verifiable in-network aggregation," in *Proc. IEEE Intl. Conf. on Data Engineering*, 2007, pp. 996–1005. DOI: 10.1109/ICDE.2007.368958.

[5] Y.-C. Fan and A. Chen, "Efficient and robust sensor data aggregation using linear counting sketches," in *Proc. IEEE Intl. Symposium on Parallel and Distributed Processing*, 2008, pp. 1–12. DOI: 10.1109/IPDPS.2008.4536265.

[6] R. W. van der Heijden, S. Dietzel, and F. Kargl, "Sedya: secure dynamic aggregation in vanets," in *Proc. ACM WiSec*, 2013, pp. 131–142. DOI: 10.1145/2462096.2462119.

[7] H.-C. Hsiao, A. Studer, R. Dubey, E. Shi, and A. Perrig, "Efficient and secure threshold-based event validation for vanets," in *Proc. ACM WiSec*, 2011. DOI: 10.1145/1998412.1998440.

[8] S. Dietzel, M. Balanici, and F. Kargl, "Short paper: towards data-similarity-based clustering for inter-vehicle communication," in *IEEE VNC*, 2013, pp. 238–241. DOI: 10.1109/VNC.2013.6737622.

[9] S. Heule, M. Nunkesser, and A. Hall, "Hyperloglog in practice: algorithmic engineering of a state of the art cardinality estimation algorithm," in *Proc. ACM Intl. Conference on Extending Database Technology*, 2013, pp. 683–692. DOI: 10.1145/2452376.2452456.

[10] ETSI, "ITS security header and certificate formats," TS 103 097, 2013.

[11] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *EUROCRYPT*, ser. LNCS 2656. Springer Berlin Heidelberg, 2003, pp. 416–432. DOI: 10.1007/3-540-39200-9_26.