ASIA: Accelerated Secure In-network Aggregation in Vehicular Sensing Networks

Xiao Wang and Patrick Tague
Carnegie Mellon University
{*xiaowang, tague*}*@cmu.edu*

***Abstract*—Vehicular Ad-Hoc Networks (VANETs) can potentially become a sensing platform. In-network aggregation, a fundamental primitive for querying sensory data, has been shown to reduce overall communication overhead at large. To secure data aggregation in VANETs, existing schemes mainly rely on digital signatures. However, generating and verifying such signatures can cause high computational overhead. More importantly, time-consuming verifications lead to the vulnerability to signature flooding attacks in which a receiver cannot timely verify all messages before their respective deadlines. In this paper, we propose ASIA as an Accelerated Secure In-network Aggregation strategy that can accelerate message verifications and significantly reduce computational overhead while retaining satisfactory security. We replace the most common tree graph with a directed acyclic graph as the aggregation structure. Resulting redundancy in information flow offers the opportunity for misbehavior detection. Meanwhile, by leveraging time asymmetry, upstream nodes in the structure can verify downstream messages through the modified light-weight TESLA scheme. We analyze the security properties of ASIA and provide evaluation results. We show that ASIA can largely accelerate message verifications and drastically reduce computational and communication overhead compared to existing schemes using the resource-consuming Elliptic Curve Digital Signature Algorithm.**

I. Introduction

With on-board sensors, a vehicle can constantly sense its surroundings. Information from sensor readings helps evaluate road conditions and will be fed to other promising applications which are not necessarily traffic- or driving-related. For example, vehicular networks can collect valuable data for environment monitoring. In this sense, the functionality and capability of VANETs are extended, which highlights its characteristics as a *vehicular sensing network* [1][2].

A VANET as a sensing network may accumulate a large quantity of data in a short period. One fundamental challenge is the resulting high transmission overhead in information collection. To reduce communication load and minimize related cost in querying information from VANETs, *data aggregation* has been proposed for use [3][4]. Vehicles can leverage an on-board unit to communicate data with neighbors so that a multihop wireless network is constructed to allow queries.

In aggregation, intermediate nodes, instead of simply forwarding data from downstream nodes, will combine all received data to create a single aggregate. For example, to perform the `SUM` aggregation, an intermediate node sums up all sensory readings passed by nodes directly downstream from it and sends a message containing the aggregation result to its upstream nodes. While bringing some benefits, the aggregation scheme offers opportunities for malicious intermediate nodes to subvert final aggregation results as they can easily manipulate the result when computing an aggregate.

The main goal of this work is to address the security concerns in data aggregation in VANETs. VANETs have several unique characteristics that contribute to the difficulties in designing secure aggregation strategies [5], including highly dynamic network topology, transitory nature of interactions, absence of centralized authority and low tolerance for errors.

Existing approaches regard digital signatures as the building block of secure data aggregation [6][7][8][9]. In IEEE 1609.2 standard [10], Elliptic Curve Digital Signature Algorithm (ECDSA) is used to provide authentication and non-repudiation. Generating and verifying signatures using ECDSA, however, leads to high computational overhead on the On-Board Unit (OBU) that validates messages. A typical OBU with a 400 MHz processor needs 20 milliseconds to verify one ECDSA signature. Given that beacon messages are broadcast every 100 milliseconds [10], a vehicle with more than 5 neighbors around cannot timely verify incoming messages. Therefore, even in a benign scenario, vehicles are likely to be overwhelmed, let alone the scenario with malicious neighbors. This potentially allows signature flooding attacks [11]. Also, signatures can causes excessive transmission overhead, especially in aggregation scenario where nodes may concatenate downstream signatures to create considerably long messages.

Secure aggregation was first studied in the context of stationary sensor networks. Due to the static topology, sensors can set up secret symmetric keys among them to facilitate security mechanisms. In the approach by Perrig et al. [14], broadcast authentication is guaranteed using TESLA [15] so that nodes can authenticate messages. To secure aggregation, each node generates a commitment to its aggregates which is verifiable to the querier. This scheme has multiple runs of information dissemination which require a relatively stable network topology. In the work by Chan et al. [16], authors assume preloaded symmetric keys at each sensor. We notice that in these schemes symmetric cryptography is preferred against signatures to reduce overhead.

To secure aggregation in VANETs, symmetric cryptography is also applied. Recent research has proposed several alternatives to the heavyweight signature strategy. In the scheme proposed by Dietzel et al. [12], nodes evaluate the trustworthiness of the aggregates using selective attestation and trust management. Han et al. [13] propose a probabilistic scheme

using the FM-sketch data function and symmetric cryptography to generate lightweight authentication codes. However, these schemes rely on pre-distribution of symmetric keys, which is considered non-realistic for VANET applications [5]. It incurs key management issues.

In this paper, we propose ASIA as an effective and efficient scheme for securing data aggregation in VANETs. Our approach can dramatically accelerate message verification because it mainly relies on hash operations which are several orders of magnitude faster than the digital signature scheme. It is able to largely reduce both communication and computational overhead compared to previous strategies.

ASIA consists of two basic security mechanisms: *Aggregate Consistency Check (ACC)* and *Generation-Skipping Verification (GSV)*. Our idea in designing ACC is providing security through introducing redundancy into the aggregation data flow. To this end, we use a *directed acyclic graph* (DAG) [17] as the aggregation structure instead of the commonly used tree graph. When performing aggregation in a DAG, one node sends its messages to multiple upstream nodes. Messages with identical content flow through network and will reach eventually a common node which can compare the received messages to detect potential misbehavior during the aggregation process. However, constructing the desired DAG in VANETs is non-trivial. In our proposed approach, vehicles will leverage location and speed information already used for safety applications to facilitate DAG construction.

GSV builds upon TESLA — a lightweight broadcast authentication scheme [15]. It allows upstream nodes in the aggregation structure to directly verify the integrity of messages from downstream nodes that are two hops away, bypassing the nodes residing between them. The philosophy of this approach is time-asymmetry authentication. The timing of message generation, packet transmission and secret key disclosure can offer authentication of source and message. In the scheme, expensive asymmetric cryptography can be avoided for most of the time.

The main contributions of this work are listed below:

- We propose two novel security mechanisms for data aggregation in VANETs, which are resource-conserving in terms of both computation and communication, and enable timely message verification.
- We describe a complete aggregation framework from the construction of aggregation structure to the actual data aggregation phase and provide security mechanisms throughout various stages.

This paper is organized as follows. In Section II, we describe our system model and adversary model. A overview of ASIA is given in Section III. We detail ASIA in Section IV. In Section V, we analyze the security properties of ASIA and present evaluation on its performance in terms of computation and communication efficiency. Finally, we conclude in Section VI.

II. System Model and Assumptions

In this section, we describe our system and adversary model and several basic assumptions.

A. System Model

In this paper, we consider a general VANET model. Vehicles are equipped with On-Board Unit (OBU) to broadcast outgoing messages and verify incoming messages. A vehicle can communicate with peers within its transmission range. To facilitate communication and security enforcement, vehicles are partitioned into groups according to certain clustering rules. Readers can refer to related work [9] for details about group management. Data aggregation is performed within each group. The network can thus be described as a general multihop wireless network consisting of a single querier and ordinary nodes that will respond to queries [18]. The logical aggregation structure will be a DAG rooted at the querier.

In the network, let A be a node. In the DAG structure, a node has n_p^A parents. We denote as $p_i(A)$ $(i = 1, \cdots, n_p^A)$ one of A's parents in DAG. All parent nodes constitute a set $\mathbb{P}(A) = \{p_i(A) \parallel i = 1, \cdots, n_p^A\}$. Similarly, we define $c_i(A)$ as child node of A, $\mathbb{C}(A)$ as the set of child nodes.

We expect the existence of PKI in VANETs for key management. Each vehicle uses only one public/private key pair at a time. We assume that public/private key pairs are under tamper-resistant protection and cannot be compromised. The authenticity can be verified by checking the digital signature on it and the corresponding certificate issued by a Certificate Authority (CA). Government agency like Department of Motor Vehicles or auto manufactures can assume the role of CA to issue certified key pairs.

We assume that each vehicle is equipped with Global Positioning System (GPS) device, which can provide meter-level positioning accuracy and nanosecond-level timing accuracy [19]. Readings from GPS are shared with neighbors without being modified or falsified. So each vehicle can continuously monitor the position and velocity of its neighbors. The IEEE 1609.2 standard instructs vehicles to broadcast `(location, speed)` beacon every $100ms$. The broadcast can be conducted efficiently and securely [11]. Also, we assume that information embedded in the beacon messages is authentic.

The precise timing provided by GPS can help achieve synchronization among vehicles. Synchronization is important to the realization of certain aspects of our security strategy, in which we exploit the 'asymmetry in time' property of TESLA to guarantee secure transmission.

It is shown that other aggregation functions can be performed using `SUM` function as the primitive [14][20]. So we focus on securing `SUM` aggregation. In the rest of the paper, the query function is `SUM` unless otherwise specified.

B. Adversary Model

Adversaries in VANETs can distort the final aggregation result in several ways. In this paper, we focus on the attack described in Definition 2.1.

Definition 2.1: Aggregate Manipulation Attack: To bias the final aggregation value, a malicious intermediate aggregator falsifies sub-aggregation result arbitrarily as long as the fabricated aggregate complies with the definition of the data type.

We do not consider denial-of-service (DoS) attacks which are not as hazardous as manipulation attacks that can fool vehicles to accept arbitrary reports. We do not consider the attack in which the attacker manipulates its own sensory readings. Without trusted computing or trusted execution technology [21], a misbehaving node can gain full control of its sensors and other on-board devices. It is able to tamper with local readings without being detected. However, there is normally a valid range for most sensory readings. For instance, the environment temperature takes the value from $0°C$ to $50°C$. Attackers can only modify the reading within this range or the reading will be discarded as an outlier. Given the relatively large scale of an aggregation group, the effect of falsified readings on the final result is largely limited. This observation is formalized in a work by Wagner [20].

We do not limit our study to a single adversary, as multiple malicious nodes may exist in a real network. Adversaries may collude to manipulate aggregates without being detected.

III. ASIA Verification Components

In this section, we present the key ideas of our approach and briefly describe the ACC and GSV mechanisms.

A. Aggregate Consistency Check

In previous work on aggregation security in VANETs, in-network aggregation is performed based on the tree structure. One benefit of this structure is its duplicate-free nature that can prevent double counting. In addition, there are many mature spanning tree construction algorithms.

In this paper, we propose to use the *directed acyclic graph* (DAG) as shown in Fig. 1 to replace the spanning tree as the aggregation structure. Aggregation in DAG differs in the sense that each node except the root, rather than sending its message to only one upstream node, will transmit it to two or more nodes. Receivers separately compute an aggregate and then send it with their own reading to a same upstream node which can thus check the integrity by comparing received messages.

We propose the *aggregate consistency check* (ACC), which verifies the aggregates of other nodes using the *atomic consistency block* (ACB). An ACB is defined by four nodes: A, $p_i(A)$ and $p_j(A)$ ($i, j = 1, \cdots, n_p^A$ and $i \neq j$) such that $\mathbb{P}(p_i(A)) \cap \mathbb{P}(p_j(A))$ is non-empty, and a verifier $v_{ij}(A) \in \mathbb{P}(p_i(A)) \cap \mathbb{P}(p_j(A))$. Then the block is defined as $ACB(A) = \{A, p_i(A), p_j(A), v_{ij}(A)\}$. An example of ACB is illustrated in Fig. 1. In a consistency check, $v_{ij}(A)$ compares two aggregates coming from $p_i(A)$ and $p_j(A)$.

For example, node A sends its reading 7 to $p_i(A)$ and $p_j(A)$ whose own readings are 3 and 8, respectively. Each node then computes the `SUM` aggregate which is sent to node D with its own reading. So $v_{ij}(A)$ will receive $(10, 3)$ from $p_i(A)$, and $(15, 8)$ from $p_j(A)$. By subtracting the reading from the aggregate, $v_{ij}(A)$ can obtain the actual data from A separately from these two messages. If $p_i(A)$ and/or $p_j(A)$ misbehave fabricating the aggregate, $v_{ij}(A)$ can detect any inconsistency after the subtraction operation.

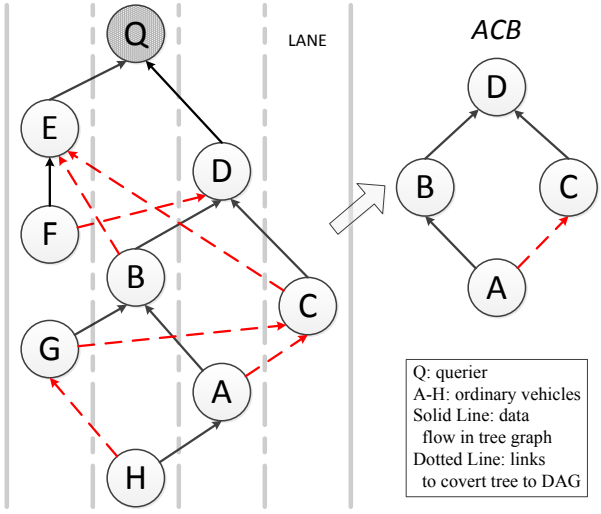

Fig. 1. DAG and Atomic Consistency Block: with two data flows originating from node A and sinking into node D, D can verify the integrity of aggregates.

One thing the attacker can do is falsify the aggregate and modify its own reading as well to ensure the subtraction remain the same. This attack is equivalent to modifying its own sensory readings. As explained in the adversary model, the effect of this attack is quite limited.

The DAG offers the possibility of locally detecting misbehavior, at the price of complexity in constructing the aggregation structure. We will detail the scheme in Section IV, describing the method to construct the DAG, the scheme to solve double-counting problem in DAG, and how to tackle with potential security threats during DAG construction.

B. Generation Skipping Verification

If $p_i(A)$ and $p_j(A)$ do not collude when they are trying to falsify their aggregates, there is strong probability that their data will result in inconsistency at node $v_{ij}(A)$. If $p_i(A)$ and $p_j(A)$ do collude and add a same value to their aggregates. Then the messages received by $v_{ij}(A)$ may look like $(90, 3)$, $(95, 8)$, which can bypass the current *'aggregate consistency check'* security scheme. Given the legitimate aggregates (10,3) and (15,8), collusive attack [1] can totally subvert the final result.

To detect aggregate manipulation attack even with collusion existing, it is favorable that node $v_{ij}(A)$ is able to directly verify the reading at A. A straightforward approach is to establish a shared secret between A and $v_{ij}(A)$. This can be done with Diffie-Hellman key exchange protocol or any appropriate method. This method, however, incurs high communication and computational overhead and may be infeasible in the VANETs scenario.

[1] In this paper, *collusion* happens among nodes at the middle tier of a ACB, in which they falsify aggregates to identical result. While for malicious nodes at different layers, they can *collaborate*. For example, nodes at the upper layer may ignore any inconsistency observed in aggregates from lower layer nodes.

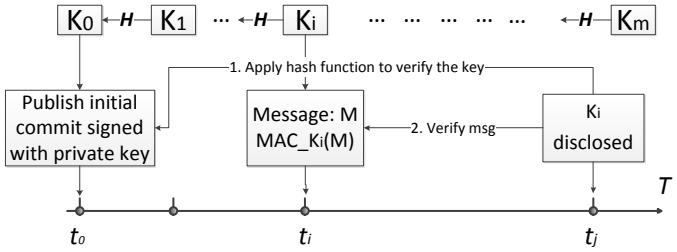

Fig. 2. TESLA Overview: hash chain construction and message verification.

We propose *Generation Skipping Verification* (GSV) which makes it possible that the verifier at the upper end in ACB can verify the integrity of readings from the source node at the lower end, even in the presence of possibly malicious nodes at the middle tier. Symmetric cryptography is used in our scheme for most time to accelerate message verification and reduce overhead. We first provide some background knowledge to help understand GSV.

GSV builds upon TESLA [15] to perform its functionality. TESLA is a packet loss tolerant broadcast authentication scheme. As shown in Fig. 2, the sender constructs a self-authenticating one-way hash chain and publishes the very first key K_0 as the *initial commit*. The sender transmits the message at time t_i and attaches it with a message authentication code (HMAC). The key K_i is disclosed at a future time point t_j. The receiver can first verify K_i by recursively applying the hash function until obtaining K_0. Then it can verify the message using the legal K_i. At the setup stage, K_0 is signed with the sender's private key and can thus act as the root of trust. The time-delayed key disclosure is based on a loose, but bounded clock synchronization between the two involved entities.

In our approach, node A first signs K_0 with its private key and then broadcasts it to $p_i(A)$ and $p_j(A)$ which are A's immediate neighbors. $p_i(A)$ and $p_j(A)$ are required to further broadcast K_0 to their neighbors so that $v_{ij}(A)$ can obtain the initial commit of A and verify subsequent keys from A.

During aggregation, when A sends a message to $p_i(A)$ and $p_j(A)$, it attaches a HMAC keyed with a key in its hash chain to help receivers verify the integrity of the message. This message also includes an expiration time, before which $p_i(A)$ and $p_j(A)$ are required to send $v_{ij}(A)$ their aggregates along with A's message. When the expiration time arrives, node A releases to $p_i(A)$ and $p_j(A)$ the key for that HMAC so that they can authenticate the message to detect any potential tampering during the transmission. However they are unable to forge the HMAC they have already presented to $v_{ij}(A)$. $p_i(A)$ and $p_j(A)$ are then required to forward the key to $v_{ij}(A)$. $v_{ij}(A)$ is thus able to verify the integrity of A's reading. Moreover, $v_{ij}(A)$ rejects messages from $p_i(A)$ and $p_j(A)$ that arrive after the corresponding K_i is disclosed.

With the GSV strategy, node $v_{ij}(A)$ can detect the manipulation attack even if $p_i(A)$ and $p_j(A)$ collude. In the following section, we describe in more details the procedure of the proposed aggregation technique.

Fig. 3. Real-World Traffic: aggregation tree spans in traffic moving direction.

IV. ASIA Algorithmic Framework

In this section, we describe ASIA in detail. We provide the complete aggregation procedure from DAG construction to aggregate computation.

A. Group Formation and Management

Vehicles in the network are arranged into groups. After the aggregation is triggered, vehicles first determine the group they belong to. Grouping is location-based. The road is dissected into small sections which basically define different groups. A vehicle equipped with GPS can automatically know its group by comparing its GPS position to the preloaded road map with sections indicated [9][22].

B. Location-aware Aggregation Tree Construction

To construct the aggregation structure shown in Fig. 1, vehicles in each group first construct a spanning tree. They will face several challenges when trying to construct a preferable tree that can facilitate subsequent procedure. Classic distributed spanning tree algorithms might generate a random tree structure whose topology is unexpected. Recall that in our scheme, the aggregator A should connect to one of its parent's siblings C to generate the ACB. If the spanning tree is generated without any constraints, the random topology may result in communication failure between A and C. They might be physically far apart from each other and out of transmission range. A vehicle in the network can increase its transmission power in case that its desired parent is out of range. Maximum transmission radius is as high as 250 m [9]. However larger transmission radius incurs more communication overhead and more severe contention over the wireless medium. Therefore, it is not desirable to increase transmission power to reach a far away parent. The tree construction algorithm should ensure that logically close nodes in the aggregation DAG are also geographically close to each other.

Given the features of real world traffic as illustrated in Fig. 3, vehicles can construct the aggregation tree along the moving direction of traffic. The tree will not span in arbitrary directions so that physical locations of vehicles will basically match their logical positions in the aggregation structure shown in Fig. 1.

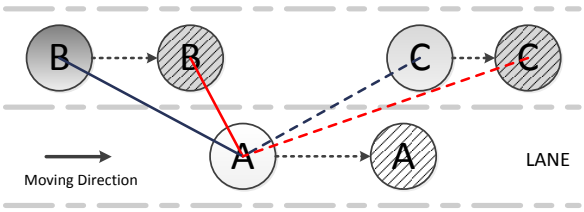

Fig. 4. Relative Position: determine relative position of neighbors using GPS information in periodic beacons.

1) Relative Position Detection: Each vehicle needs to determine the relative position between itself and neighbors. They can leverage the location information embedded in the periodically broadcast beacon, as illustrated in Fig. 4.

At time t_1, A receives beacon message from B and extracts its location $l_B^{t_1}$. The distance between A and B at t_1 can be calculated as $d_{AB}^{t_1t_1} = ||l_A^{t_1} - l_B^{t_1}||$. At time t_2, A obtains the distance between B's current position and A's position at t_1, $d_{AB}^{t_1t_2} = ||l_A^{t_1} - l_B^{t_2}||$. By comparing $d_{AB}^{t_1t_2}$ and $d_{AB}^{t_1t_1}$, A can determine B's relative position. If $d_{AB}^{t_1t_2} < d_{AB}^{t_1t_1}$, B is behind A. If $d_{AB}^{t_1t_2} > d_{AB}^{t_1t_1}$, B is in front of A. A maintains a record of its neighbors regarding the relative position. Each vehicle will choose the vehicles behind it as parent candidates.

Considered that maximum speed limit on free way is $65mph$, one vehicle can move at most $30m$ in one second. Beacon message is broadcast periodically every $100ms$. A vehicle can move $3m$ between consecutive beacons or $6m$ between every other beacons. Commodity GPS device can guarantee $1m$ level accuracy [19]. Hence, the scheme can tolerate the imprecision of location information. When traffic speed drops, vehicles can increase the time interval between t_1 and t_2 accordingly.

If B is very close to A or if B's speed is much faster so that B will pass A (assume B is behind A at t_1) during the interval between t_1 and t_2, it might be wrong to state that B is behind A when $d_{AB}^{t_1t_2} < d_{AB}^{t_1t_1}$. Vehicles that travel at much higher or lower speed will not be chosen as parent candidates. The threshold of the speed difference is correlated with distance, traffic density and transmission range. Another scenario as mentioned above is when two vehicles are too close to each other. Then it might be difficult for A to accurately determine B's relative position given the error in positioning. In this case, however, it is not necessary to infer the relative position. It has little, if any, negative effect on a successful construction of the aggregation tree since they are close enough.

2) Lane Identification: Knowing the lane on which neighbors are running, vehicles can construct a more stable tree structure that maintains better connectivity. In some time, the carpool lane has much faster traffic while the outermost lane has slow traffic. Lane number information will be considered when choosing parent node.

In this paper, we consider the 4-lane freeway shown in Fig. 5. Lanes are identified as lane number $1, 2, 3, 4$. Lane number 1 is the carpool lane.

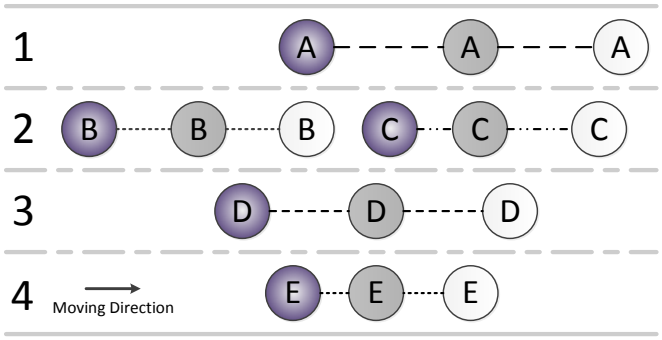

Fig. 5. Lane Identification: identify different lane through trajectories of neighbors

Procedure 1 Tree Construction: The Initiator

APPLY STAGE:
Input: neighbor_info
Output: parent_candidates
pick parent node candidates
assign preference
broadcast request to neighbors
wait for reply

ACCEPT STAGE:
Input: admission_decision
Output: parent_node
select node with highest preference score
broadcast decision

Vehicles can use GPS location information to infer the lane number [11]. As demonstrated in Fig. 5, with location of neighbors and itself at 3 time points, one vehicle can obtain the trajectories and thus infer the lane number. We omit details of the algorithm due to space limit.

Lane identification is helpful when there is 'significant' persistent speed difference between traffic on different lanes. Through velocity information in beacons, each vehicle can evaluate the average speed of traffic, traffic density and speed difference. Those factors influence tree structure construction.

3) Tree Construction Protocol: Under the circumstance that traffic on different lanes move at 'significant' different speeds, to construct an aggregation tree with better connectivity, vehicles in lane number 2 and 3 (Type I) will form the backbone of the tree. The construction procedure will experience three stages from *apply* to *admit* to *accept*.

A Type I vehicle sends request to Type I vehicles behind it to ask them to become its parent candidates. The number of candidates is decided in accordance with current traffic density. Each vehicle maintains an adequately large group of neighbors by adjusting its transmission range so that it has enough parent candidates. In the request, the sender indicates its preference over all candidates. Taking into account lane number, relative distance and speed, a vehicle can determine the preference value assigned to each candidate. They would generally give

Procedure 2 Tree Construction: The Responder

ADMIT STAGE:
Input: request_list
Output: admission_list
wait for incoming requests
select potential child nodes
broadcast admission decision

higher preference to vehicles with same lane number, smaller distance and similar speed.

Vehicles receiving requests reply acknowledgement to senders. Whether to accept the request is probabilistic with the possibility proportional to preference value. The number of child nodes to admit is also based on traffic density. Last, the vehicle initiating the request chooses one vehicle as its parent from those who accept its request.

We suggest that vehicles in lane number 1 and 4 (Type II) prefer to connect to Type I vehicles as leaf nodes since they are more likely to create disconnectivity in the aggregation structure. In extreme situation, vehicles in the carpool lane move fast while those in other lanes get stuck. Then vehicles divide into two groups and perform aggregation separately.

Procedures are listed in Procedure 1 and 2.

C. From Aggregation Tree to DAG

Vehicles in each group construct a DAG to perform aggregation. The DAG is built upon the tree graph. Vehicles construct a spanning tree first and then transform it into a DAG.

A node X can be aware of its siblings' presence. This is because when its parent node P communicates with other nodes to establish parent-child relation, messages from P is clear to X so that X can overhear the communication and figure out who else became P's child node. With its own neighbor list, X can know the siblings who are within its transmission range. X puts them on its `sibling_list`. Node X will randomly choose one partner-node from this list and send out the request.

All child nodes of P are guaranteed to be geographically close and within the same collision domain in their communication. This is achieved by our tree construction method. So at each time, only one node can send or respond to pairing requests. Any node that received a request and is still available for pairing will accept the request by default. Those who already paired up with others would normally reject any subsequent requests. One node will have no partner-node if there are odd number of nodes involved in the paring procedure. So each node sets up a flag `rej` indicating the times having been rejected. It is guaranteed to be accepted in its third attempt. Also, before sending out the request, each node can make a fresh choice based on the knowledge it obtains via overhearing. If the candidate on its list is no longer available, it picks randomly from the rest in the `sibling_list`. The overall flowchart is presented in Fig. 6.

Another approach is to let an individual node act as the central controller in the pairing task, and broadcast the decision

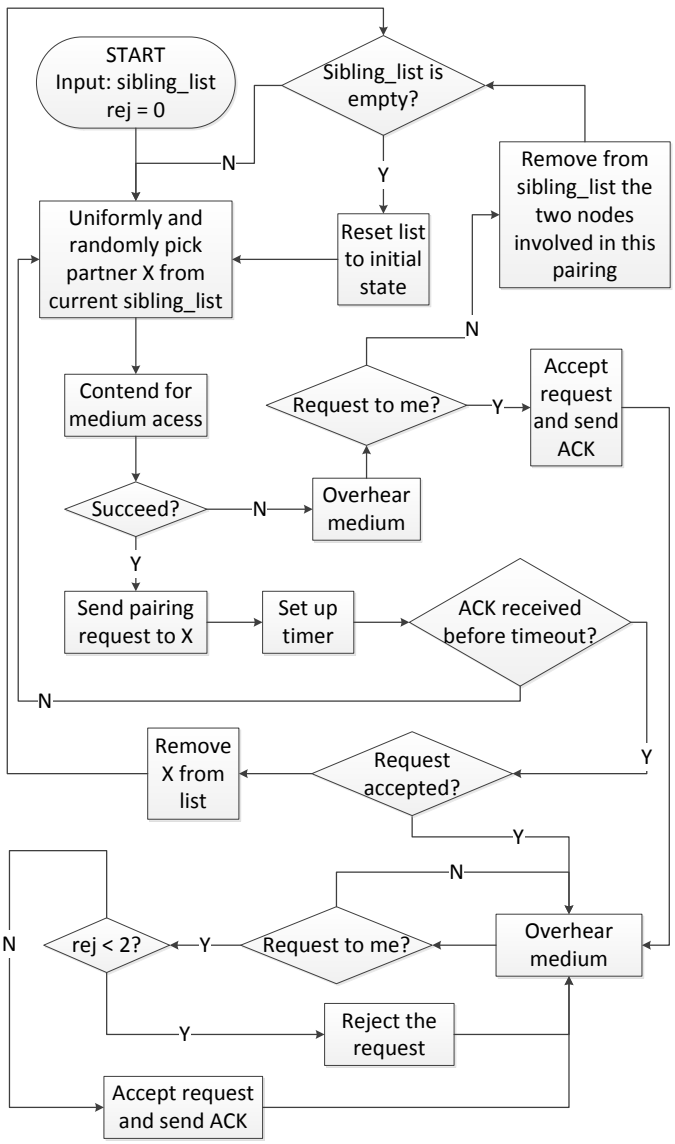

Fig. 6. DAG Formation Flowchart

to its child nodes which forward the message one level below to their children. The centralize approach is more efficient in terms of communication overhead. The drawback is controller might manipulate the pairing result to facilitate collusion.

D. Verification through Time Asymmetry

We give an example to explain the GSV. Consider the atomic checking structure in Fig. 1. At time t_i, A sends message $\{A, t_i, r_A, Agg_A, t_j^E, MAC_{K_i}(A||t_i||r_A||Agg_A||\ t_{i+1})\}$ to B and C, where r_A is A's own reading, Agg_A is aggregate computed by A and $t_j^E(>\ t_i)$ is the expiration time for this message before which it should be received by D. All information is concatenated and the sender creates an HMAC over it. B and C must include into their messages to D the content from A. They cannot forge the HMAC because they do not have the key K_i. A sends K_i to B and C at time

$t_k (> t_j^E)$ which is then forwarded to D so that D can verify the integrity of the aggregate from A.

E. Monitoring by Overhearing

Overhearing happens all through the process from aggregation structure construction to the completion of an aggregation. During tree and DAG construction phases, overhearing can prevent malicious nodes from manipulating the parent-child pairing process. This advantage increases the difficulty for colluding nodes to connect to each other and enhances security.

Overhearing also facilitates information sharing among neighbors, which can benefit vehicles in DAG construction. For instance, one node can identify its siblings by overhearing the conversation between its parent node and others.

F. Aggregates Computation

In aggregation through a DAG structure, nodes in the network first determine whether the aggregation function is duplicate-sensitive or not. For duplicate-insensitive functions, data are aggregated as in the tree structure. For duplicate-sensitive functions, the node at upper end in the ACB should remove any duplicates when computing the aggregate.

In an alternative approach, suppose one node with data x has k parents. In aggregation, this node sends x/k to each of its parents. For aggregation functions such as `SUM`, the aggregator does not need to take extra actions. Another advantage of this scheme is the reduced error caused by single packet loss [17].

V. Analysis

In this section, we analyze the security and performance of ASIA, both through logical arguments and simulation.

A. Security Evaluation of ASIA

We begin with the a logical evaluation of the security properties of ASIA.

Under our framework, TESLA provides source authentication in the broadcast environment without incurring considerable overhead. Message authentication can be guaranteed by HMAC though there is a delay in verification because corresponding key is disclosed later. Non-repudiation is not provided in our scheme. However with the overhearing tactic, neighbors can monitor conversations and keep a copy of messages allowing them to reach a local consensus over possible misbehavior. Alternatively, upon detecting an inconsistency, the detector can ask the suspect to retransmit the message signed with its private key. This can help either to correct the inconsistency or to offer evidence of tampering.

In the scenario where attackers act independently without colluding, inconsistencies will be detected at benign nodes or the querier with high probability, regardless of the number of malicious nodes. There probability that two independent attackers modify their aggregates to the same value is negligible. The more attackers there are, the lower probability that they can avoid detection because every pair of adversaries residing at the middle tier of the ACB (namely $p_i(A)$ and $p_j(A)$) need to generate identical fabricated aggregates.

In the case that some attackers collude, their parent nodes can observe misbehavior by verifying the downstream messages from nodes that are two-hop away. However, if all parent nodes are also malicious, the attack cannot be detected.

We define ***effective attack set*** $\mathcal{U}$ as the set with minimized number of colluding/collaborating attackers who can mount successful attacks without being detected. In the DAG, node A is said to be below node B if the distance between A and the the querier Q counted by hops is longer than the distance between B and Q. A is said to be below set $\mathcal{U}$ if A is below all nodes in $\mathcal{U}$. The set $\mathcal{U}$ has the following two properties:

- Any nodes not belonging to $\mathcal{U}$ cannot directly verify messages from nodes below $\mathcal{U}$;
- There exists at least one node that is one hop below $\mathcal{U}$ whose aggregates flow into nowhere other than $\mathcal{U}$.

With these two properties, attackers in $\mathcal{U}$ can gain full control of at least one piece of aggregate. Successful attacks thus follow. The number of attackers in $\mathcal{U}$ depends on the local DAG topology. To increase this number, GSV can be upgraded so that more upper level nodes can directly verify downstream messages. Node D in the ACB is supposed to further send its parent nodes the initial commit of A.

To mount a successful attack, colluding attackers must ensure that their logical positions in the aggregation structure satisfy the properties of $\mathcal{U}$. This cannot be completely determined on their own. They would try their best to stay close to enjoy higher probability of mutual connection during the DAG construction and they definitely accept the requests from associates. However, benign neighbors may disrupt their plan. Benign nodes may win the contention to gain priority claiming a desirable parent node and/or sibling node. Corresponding attackers receiving the request cannot simply reject it because other nodes around keep overhearing the medium and can identify policy-incompliant actions.

To see how ACC and GSV combine to thwart attacks, consider the instantiation of DAG in Fig. 1. If A and G collude, both B and C can verify the aggregate from H and thus detect the attack. If B and G collaborate, C can detect misbehavior by comparing the data from G and A. If A, G and B misbehave, C, D and E can notice the attack. The only way to bypass the security scheme is to let all of A, B, G and H collude/collaborate. They constitute the effective attack set in this instantiation.

We next provide a simulation study to validate our claims. In our setting, attackers are assumed to be distributed uniformly and randomly throughout the group. An agreement about manipulating the aggregate exists and is known to all attackers so that every attacker adds the same value to the aggregate when they are tampering with the result. Therefore, their independent manipulations actually constitute collusion if their logical locations in DAG happen to satisfy the requirements of the effective attack set $\mathcal{U}$.

Simulation parameters and corresponding sample values are listed in Table I. The network and security parameters correspond to parameters used in previous VANET simulation studies [23][24], and the lane width is standard in the USA.

TABLE I
SIMULATION PARAMETERS

Parameter	Value
Length of Road Segment	1000 m
Lane Width	3.7 m
Max Speed Limit	120 km/h or 75 mph
Number of Vehicles	50 – 150
MAC/PHY	802.11
Radio Propagation	shadowing model
Public Key Size	256 bits
Signature Size	512 bits
SHA-1 Size	160 bits
ECDSA Generation Time	7 ms
ECDSA Verification Time	22 ms
Hash Operation Time	1 μs

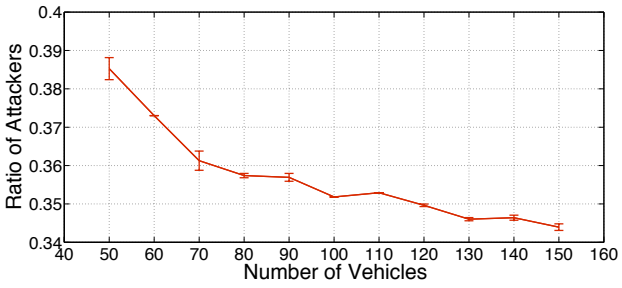

Fig. 7. We evaluate the ratio of attackers as the minimum fraction of the group that must collude in order to achieve a 5% probability of success in manipulating the aggregate.

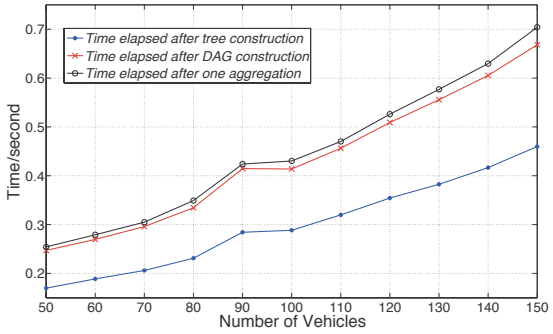

Fig. 8. We evaluate the latency of various stages of the ASIA scheme, indicated as the total time to completion.

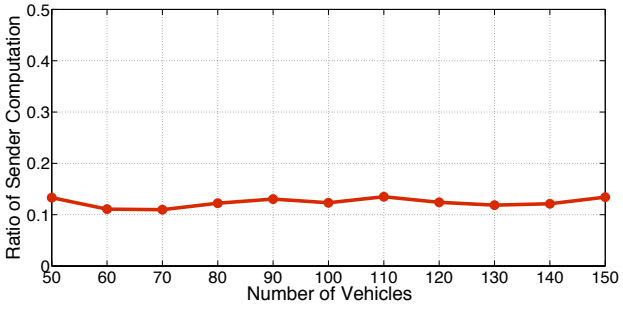

Fig. 9. We evaluate the sender's computational overhead using ASIA relative to that of the approach using ECDSA.

The road segment which defines the aggregation group is 1000 m long. The number of vehicles in one segment varies from 50 to 150, and this number can also indicate the traffic density. Vehicles move according to random traffic pattern. The maximum speed limit is 34 m/s. To simulate realistic road traffic, we use SUMO [25] to generate trajectories for each vehicle. In this simulation, we have 20 instantiations of traffic for each number of vehicles.

To evaluate the robustness of our scheme against the aggregate manipulation attack, we define the minimum *ratio of attackers* that is required to meet a 5% probability of the existence of an effective attacker set $\mathcal{U}$ in the DAG. Fig. 7 illustrates the simulated ratio of attackers with error bars representing the standard deviation over the 20 trials. The ratio of attackers required decreases from 38% to 34% as traffic density goes up. Two factors contribute to the difficulty of mounting a successful attack. First, malicious nodes need to be clustered nearby, which is not easy to achieve in a random distribution. Second, any geographically close nodes would have a difficult task in forming the desired logical connections.

B. Performance Evaluation

In performance evaluation, we assess the latency, computational overhead and communication overhead of ASIA. We use the same parameters as previous including those in Table I.

In VANETs with highly dynamic topology, latency is an important figure to reflect the efficiency of aggregation structure construction. We evaluate the latency for tree construction, tree to DAG conversion, and one run of aggregation. As shown in Fig. 8, latency rises as the number of vehicles increases. More traffic in the segment implies more contention for wireless medium access, and longer delay naturally follows. There are several factors leading to the latency including contention and retransmission in wireless communication; radio propagation; and message generation and verification. In our scheme, we use HMAC instead of a digital signature to provide source and message authentication, which reduces the latency of packet generation and verification phases. Meanwhile, HMAC is smaller in size and thus consumes less time to transmit. This two benefits will be clear as we present the improvement on computational and communication efficiency. After construction, the DAG structure can support multiple runs of aggregation operation. We can see from Fig. 8 that it takes relatively less time to complete an aggregation procedure, so additional runs do not significantly alter the overall latency.

Vehicles have constrained computational resource which may be exhausted by extensive computation, especially in the case of signature generation and verification. A typical OBU with 400MHz processor requires 20 ms to verify one ECDSA digital signature, the basic security primitive employed in the IEEE 1609.2 standard. We compare our scheme with the approach in which ECDSA is used to provide authentication and message integrity, and the aggregation is performed through a tree structure [9]. We refer to this scheme as '*All-Sig*'.

We evaluate the computational overhead of aggregation

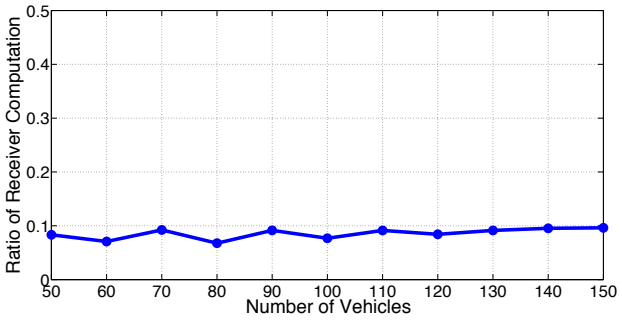

Fig. 10. We evaluate the receiver's computational overhead using ASIA relative to that of the approach using ECDSA.

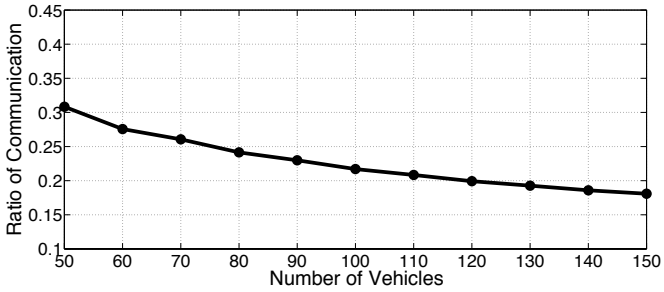

Fig. 11. We evaluate the communication overhead using ASIA relative to that of the approach using ECDSA.

structure construction to the complete of 10 runs of the aggregation operation. Fig. 9 shows the sender's computational overhead in ASIA to that of the All-Sig approach. We can observe a drastic decrease in computational overhead, less than 15% of that required for ECDSA.

Fig. 10 similarly shows the receiver's computational overhead. The ratio of computation against the All-Sig strategy is less than 10%. The additional savings at eh receiver side is because signature verification is more time consuming than generation. In DAG structure, one sender sends its message to multiple receivers, increasing the verification task load at receiver, but, the net effect still favors the receiver.

Fig. 11 compares the communication overhead between ASIA and the All-Sig approach. The total communication overhead of ASIA is only 18% – 31% of that of the All-Sig approach, and it drops with the growth in number of vehicles.

VI. Conclusion

In this paper, we investigate secure and efficient data aggregation in VANETs as a sensing platform. We propose ASIA as an aggregation strategy using a directed acyclic graph instead of a spanning tree as the aggregation structure. We propose two security mechanisms, aggregate consistency check and generation-skipping verification, both leveraging the DAG structure. Our mechanisms rely primarily on symmetric cryptography, enabling timely message authentication and providing significant reduction in both computational and communication overhead compared to signature-based schemes. Our evaluation demonstrates that the more robust aggregation structure provides strong protection against the aggregate manipulation attack, even with a significant number of colluding attackers.

References

[1] T. Willke, P. Tientrakool, N. Maxemchuk, "*A Survey of Inter-Vehicle Communication Protocols and Their Applications*," Communications Surveys and Tutorials, IEEE , vol.11, no.2, pp.3-20, Second Quarter 2009.

[2] U. Lee, M. Gerla, "*A Survey of Urban Vehicular Sensing Platforms*," Computer Networks, Volume 54, Issue 4, Pages 527-544, ISSN 1389-1286, March 2010.

[3] S. Dietzel, F. Kargl, G. Heijenk, and F. Schaub, "*Modeling in-network aggregation in VANETs*," Communications Magazine, IEEE , vol.49, no.11, pp.142-148, November 2011.

[4] C. Lochert, C. Wewetzer, A. Luebke, and M. Mauve, "*Data Aggregation and Roadside Unit Placement for a VANET Traffic Information System*," in Proceedings of the fifth ACM international workshop on VehiculAr Inter-NETworking (VANET '08). ACM, New York, NY, USA, 2008.

[5] B. Parno, and A. Perrig, "*Challenges in Securing Vehicular Networks*," in Proc. of the Workshop on Hot Topics in Networks (HOTNETS-IV), 2005.

[6] J. Molina-Gil, P. Caballero-Gil, C. Herna, and C. Caballero-Gil, "*Data Aggregation for Information Authentication in VANETs*," in 6th International Conference on Information Assurance and Security, Aug. 2010.

[7] M. Raya, and J. Hubaux, "*Securing Vehicular Ad Hoc Networks*," Journal on Computer Security, 15(1): 39-68, Jan. 2007.

[8] N. Ristanovic, P. Papadimitratos, G. Theodorakopoulos, J.-P. Hubaux, and J.-Y. Leboudec, "*Adaptive Message Authentication for Vehicular Networks*," in Proc. of ACM VANET '09, 2009.

[9] M. Raya, A. Aziz, and J. Hubaux, "*Efficient Secure Aggregation in VANETs*," in ACM VANET '06, New York, NY, USA, 2006.

[10] IEEE, "*1609.2-2006: IEEE Trial-Use Standard for Wireless Access in Vehicular Environments-Security Services for Applications and Management Messages*," IEEE Standard, 2006.

[11] H.-C Hsiao, A. Studer, C. Chen, A. Perrig, F. Bai, B. Bellur, and A. Iyer, "*Flooding-Resilient Broadcast Authentication for VANETs*," in Proc. ACM MobiCom 11, 2011.

[12] S. Dietzel, E. Schoch, B. Konings, M. Weber and F. Kargl, "*Resilient Secure Aggregation for Vehicular Networks*," IEEE Journal on Network Management of Global Internetworking, Jan. 2010.

[13] Q. Han, S. Du, D. Ren, and H. Zhu, "*SAS: A Secure Data Aggregation Scheme in Vehicular Sensing Networks*," in Proc. IEEE ICC, 2010.

[14] H. Chan, A. Perrig, and D. Song, "*Secure Hierarchical In-Network Aggregation in Sensor Networks*," in Proc. of the 13th ACM Conference on Computer and Communications Security, CCS '06, 2006.

[15] A. Perrig, R. Canetti, J. D. Tygar, and D. Song, "*The TESLA Broadcast Authentication Protocol*," in RSA CryptoBytes, 2002.

[16] H. Chan, A. Perrig, B. Przydatek, and D. Song, "*SIA: Secure Information Aggregation in Sensor Networks*," Journal on Computer Security, 15(1): 69-102, Jan. 2007.

[17] S. Nath, P. B. Gibbons, S. Seshan, and Z. R. Anderson, "*Synopsis Diffusion for Robust Aggregation in Sensor Networks*," in Proc. ACM SenSys '04, New York, NY, USA, 2004.

[18] S. Eichler, "*A Security Architecture Concept for Vehiclular Network Nodes*," 2009.

[19] LEA-6 U-Blox 6 GPS Modules Data Sheet, 2010.

[20] D. Wagner, "*Resilient Aggregation in Sensor Networks*," in Proc. ACM SASN '04, New York, NY, USA, 2004.

[21] E. Gallery, "*An overview of trusted computing technology*," in C. Mitchell, editor, Trusted Computing, chapter 3. IEE, 2005.

[22] K. Sampigethaya, M. Li, L. Huang, and R. Poovendran, "*AMOEBA: Robust Location Privacy Scheme for VANET*," IEEE Journal on Selected Areas in Communications, 2007.

[23] J. J. Haas, and Y.-C. Hu, "*Communication Requirements for Crash Avoidance*," in Proc. IEEE VANET '10, 2010.

[24] A. Studer, F. Bai, B. Bellur, and A. Perrig, "*Flexible, Extensible, and Efficient VANET Authentication*," Journal of Communications and Networks, 11(6): 574-588, Dec. 2009.

[25] M. Behrisch and L. Bieker and J. Erdmann and D. Krajzewicz, "*SUMO - Simulation of Urban MObility: An Overview*," in Proc. of the Third International Conference on Advances in System Simulation, SIMUL '11, Barcelona, Spain, 2011.