Probabilistic Validation of Aggregated Data in Vehicular Ad-hoc Networks *

Fabio Picconi †
Laboratory of Computer Science, Paris 6
Université Pierre et Marie Curie
8, rue du Cap. Scott
75015 Paris, France
fabio.picconi@lip6.fr

Nishkam Ravi
Department of Computer Science
Rutgers University
110 Frelinghuysen Road
Piscataway, NJ 08854
nravi@cs.rutgers.edu

Marco Gruteser
WINLAB, ECE Department
Rutgers University
110 Frelinghuysen Road
Piscataway, NJ 08854
gruteser@winlab.rutgers.edu

Liviu Iftode
Department of Computer Science
Rutgers University
110 Frelinghuysen Road
Piscataway, NJ 08854
iftode@cs.rutgers.edu

ABSTRACT

Vehicular ad-hoc networks present great opportunity for information exchange and equal opportunity for abuse. Validating traffic information without imposing significant communication overheads is a hard problem. In this paper, we propose a solution for validating aggregated data. The main idea is to use *random checks* to probabilistically catch the attacker, and thereby discourage attacks in the network. Our solution relies on PKI based authentication and assumes a tamper-proof service in each car to carry out certain secure operations such as signing and timestamping. We try to keep the set of secure operations as small as possible, in accordance with the principle of *economy of mechanism*. We show that our solution provides security without significant communication overheads.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and Protection*

General Terms

Security

*This material is based upon work supported in part by NSF under grants ANI-0121416, CNS-0520123 and CNS-0584475
†Visiting the Department of Computer Science, Rutgers University

VANET'06, September 29, 2006, Los Angeles, California, USA.

Keywords

Aggregation, data validation, vehicular networks, malicious nodes

1. INTRODUCTION

Vehicles equipped with GPS receivers and small computer screens have become a reality in the car industry. Existing products already assist the driver by displaying a map with the car's location along with a programmed route. In the near future, vehicular ad-hoc networks will provide real-time information to the driver or the car's safety systems. Using short-range radio communications such as DSRC [1], vehicles will be able to exchange messages describing the state of the road ahead (e.g traffic congestion or accident warnings).

Although significant work has been done on designing protocols for vehicular networks, ensuring the security of these systems remains a challenge. Conflicting goals, such as anonymity and authentication, must often be taken into account. Messages are exchanged using unreliable wireless communications [1] and across a highly dynamic ad-hoc network. In spite of this, critical data must be delivered quickly before it becomes outdated and useless. Attacks in vehicular networks vary according to the resources and motives of the attacker [14, 10]. In a *passive attack*, communication between cars is subject to eavesdropping. The motivation can be tracking a given car or profiling its driver. *Active attacks* involve injecting, altering, or blocking network packets. Examples include: Denial of Service(DoS), message omission, spoofing, and bogus information attacks.

In this paper, we focus on spoofing and bogus information attacks. In a spoof, a car creates fake car identities and broadcasts fake data as if it were received from those cars. A bogus information attack involves injecting information that does not correspond to real events or observations. For example, drivers in a hurry may try to divert traffic by falsely reporting a traffic jam on the road ahead; or a car might report false location information about itself and other cars, so as to deny involvement in an accident. Reputation-based

schemes are hard to devise given the ad-hoc nature of the network– a car would interact with another car for a few seconds, a few minutes at most, and would probably never hear from it again. Also, reputation schemes usually imply strong identities, which violate privacy and anonymity.

A mechanism for validating data on the fly must be employed. One approach (Golle et at [5]), is to correlate data from different cars and cross-validate it against a set of rules (e.g "cars located within wireless range at a given time must have observed each other"). If most cars are honest, data from a malicious car can be identified and discarded. In this approach, strong authentication is not required but messages must be signed to distinguish data generated by different cars. This approach is limited in the set of attacks it can counter, and fails if the number of malicious cars in a region is greater than the number of honest cars. In the early stages of vehicular networks, low density areas will be quite common, as few cars will be equipped with the necessary hardware/software. In low density areas, a few colluding aggregators could easily outnumber honest aggregators. Also, cross-validation works under the assumption that there are multiple sources of information (i.e multiple cars observing the same event), which is not always the case. This approach requires a set of rules against which the incoming information is cross-validated and incurs data overhead due to digital signatures.

Raya et al [14] suggest using stronger identities to avoid spoofs and outsider attacks, and to establish the identity of the message sender when required. They suggest that every car should sign messages with a key certified by a trusted authority. Using strong authentication has several advantages: (1) it prevents Sybil [3, 4] attacks and spoofs, (2) "outsiders" (i.e. cars using uncertified keys) can only cause limited harm (they can still launch a DoS attack), (3) a message can be linked to the sender's identity to be used as evidence when required. Privacy is ensured by providing multiple keys to each car and changing them periodically. A tamper-proof device is used to securely store the car's private keys and to carry out secure operations such as signing and timestamping. Digital signatures, however, are large, typically in the order of tens (using Elliptic Curve Cryptography) to hundreds of bytes (RSA). This adds a significant overhead, especially if records are small. In a wireless environment, bandwidth is a limited resource, so decreasing the record size increases the number of records that can be transmitted within the acceptable time frame, i.e., before the records become useless. In addition, the challenging radio propagation environment in vehicular networks motivates protocol designs with small message sizes.

One way to use available bandwidth more efficiently is to aggregate the information of several cars into a single record, as in the case of a V2V traffic information system [8] where cars share information about each other. Aggregation, however, aggravates the security problem. A malicious aggregator may send aggregated records that do not correspond to real data. For instance, it may falsely report a congested road by pretending to have aggregated more records than it has actually received from cars ahead of it.

In this paper, we present a way to probabilistically detect malicious cars that generate false aggregated information. In particular, we focus on validating speed and location information, which is common to most vehicular applications. We consider the case where cars lie about other cars in order to generate false traffic scenarios by launching spoofs and bogus information attacks. As part of problem definition, we seek to minimize data overhead (i.e bandwidth requirement) and maximize security.

We assume that every car has a tamper-proof service that carries out certain secure operations, such as signing of records, timestamp generation and random-number generation (construction of such a tamper-proof service is described in Section 3.2). We try to keep the number of secure operations to a minimum, in accordance with the principle of *economy of mechanism* [18].

The main idea is to challenge the aggregator to provide a proof that can be used to probabilistically validate the aggregated record. An aggregated record is created by combining and compressing information contained inside several individual records. To validate the aggregated record, the aggregator is asked to provide a randomly-chosen original signed record (whose information is contained in the aggregated record), after the aggregated record has been sent. If the corresponding record was made up, then it will not be possible for the aggregator to produce the original signed record, and he will be caught. The random check therefore acts as a deterrent.

Our validation mechanism does not require correlating data from different aggregators. The data from each aggregator is verified individually. Also, the overhead of our approach is minimal in terms of data and communication costs. Rest of the paper is organized as follows. Section 2 describes our model and assumptions. We present our solution in Section 3. Preliminary evaluation is presented in Section 4. Section 5 discusses related work. We conclude in Section 6 with a note on future work.

2. MODEL AND ASSUMPTIONS

We assume that each car is equipped with a tamper-proof service for carrying out data signing, timestamp generation and random-number generation (as suggested in [14]). Furthermore, we assume that this service signs all records with the same private key within a given time interval, after which it switches to a new key without any user intervention. This prevents a car from creating spoofs by signing messages with different keys. Changing keys periodically ensures privacy. Also, we assume that the tamper-proof service provides a *transmit-buffer*. Data once put in this buffer *will be* transmitted.

By having a few trusted operations in every car, security can be provided without compromising on flexibility. The applications themselves do not have to be trusted and can implement their own aggregation modules and communication protocols. Our design rationale is inspired by the principle of *economy of mechanism* which states that "the protection system's design should be as simple and small as possible" [18]. In Section 3.2, we describe how such a tamper-proof service can be practically implemented using currently existing technology.

We assume that each car periodically creates and transmits a small record containing some application-specific data. Records contain information observed by the car, although not necessarily about the car itself. This could also be information about the car's environment (other cars, road segment being repaired, accident, road-size facilities and stores, etc.). The scope of this paper is limited to information about cars. We expect records to be propagated through the ad-

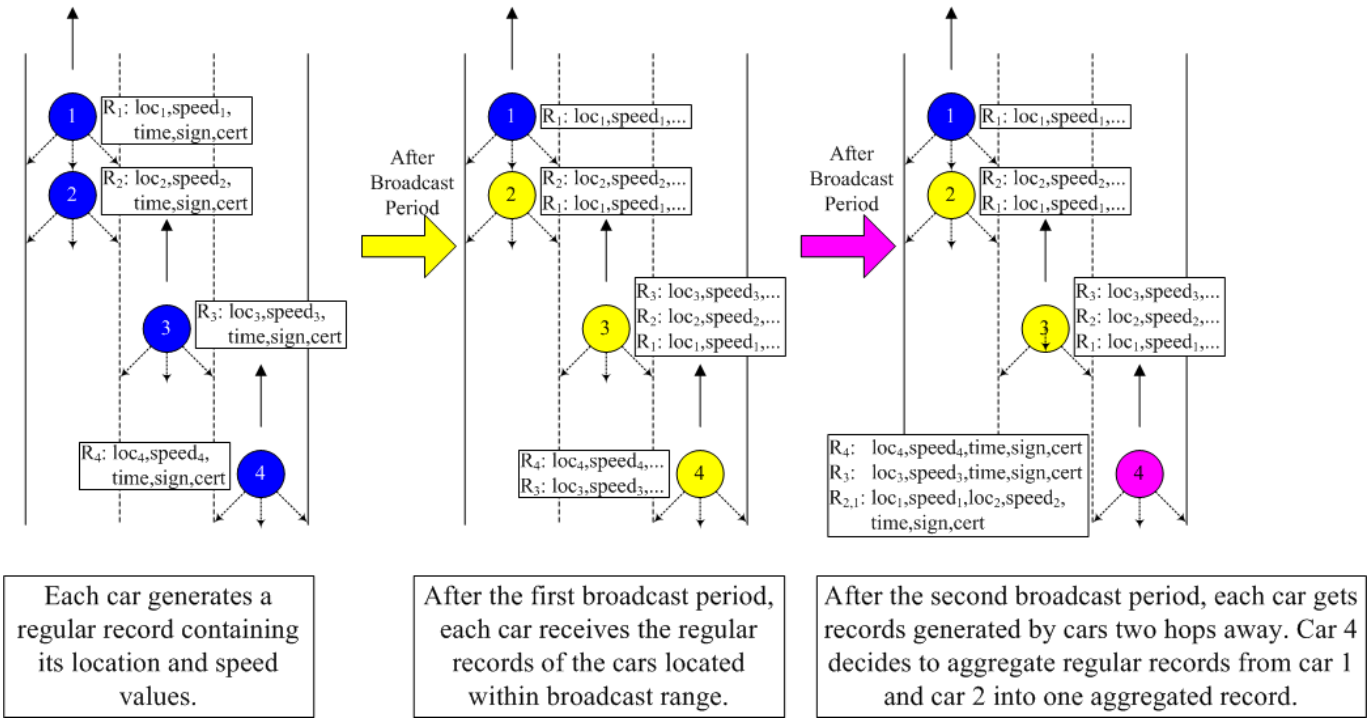


Figure 1: Speed and location information exchange in a V2V traffic information system

hoc network by a diffusion protocol, although this is not a requirement for our validation mechanism.

As an example, we will consider a distributed traffic information system [8]. Each car maintains a database of records containing speed and/or location information about itself and other cars. The records in the database are periodically broadcasted. Every time a new record is received, it is merged with the database. When no aggregation is being carried out, the *merge* operation could result in either insertion of the received record (if it is about a car whose record is not in the database), or replacement of an old record about a car with the one just received. The records do not undergo any modification and therefore the original signatures and timestamps are retained. In the presence of aggregation, the *merge* operation is much more complicated. Since records need to be aggregated, information from records has to retrieved and put in a new aggregated record. Each aggregated record is constructed from a number of non-aggregated records and is supposed to contain information about multiple cars. The aggregated records are signed by the receiver before they are sent out. A malicious receiver can easily put bogus information in an aggregated record.

Figure 1 exemplifies communication and aggregation in a typical traffic information system. Since drivers are only interested in the state of the road ahead, a car only process messages sent by cars located ahead of it. This produces a flow of records in the direction opposite to traffic. Every traffic information system could have its own policy regarding when and by whom data is aggregated. For instance, one policy could be to aggregate all records corresponding to cars in a road segment. Another policy could be to perform aggregation when a certain bandwidth threshold is exceeded (e.g more than k bytes transmitted by a car). For more details on data aggregation policies, refer to [9].

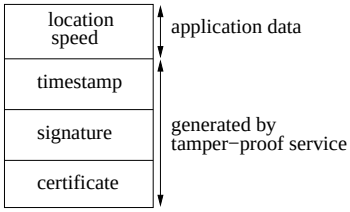


Figure 2: Structure of a regular record

2.1 Aggregation

Figure 2 shows the structure of a regular (non-aggregated) record containing speed and location information. To construct a regular record, the application sends data to the tamper-proof service, which adds a timestamp and signs the record with the car's current private key. The record also contains a certificate from the trusted authority which is used by other cars to verify the validity of the public key used for the signature. If the certificate is missing or invalid, the record is dropped.

Figure 3 shows the structure of an aggregated record constructed by extracting application data and car id from several *properly* signed regular records, putting it in a single record and getting it signed by the tamper-proof service. Note that the new aggregated record may not have the application data in the same format as in the original record. For example, the application could approximate the value of

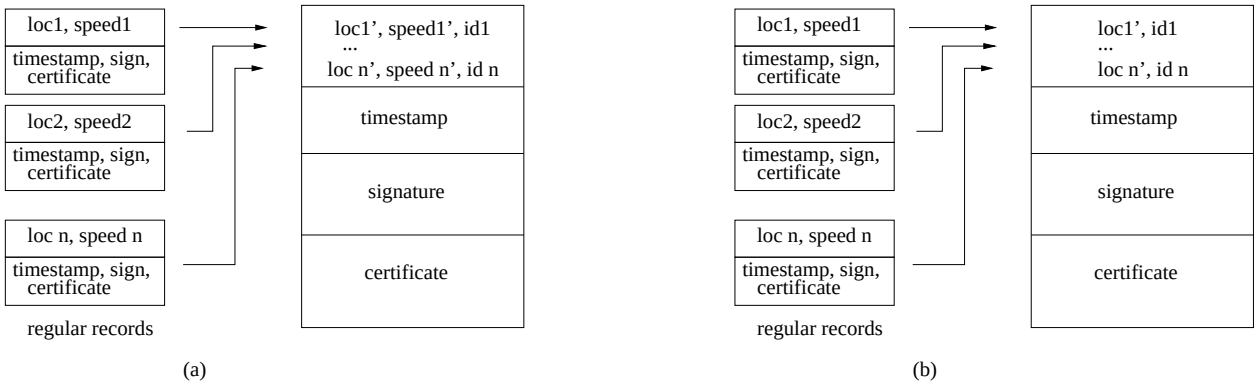


Figure 3: Construction of syntactically aggregated records: (a) loc1 → loc1' (e.g 23.0003 → 23), (b) only location information is retained

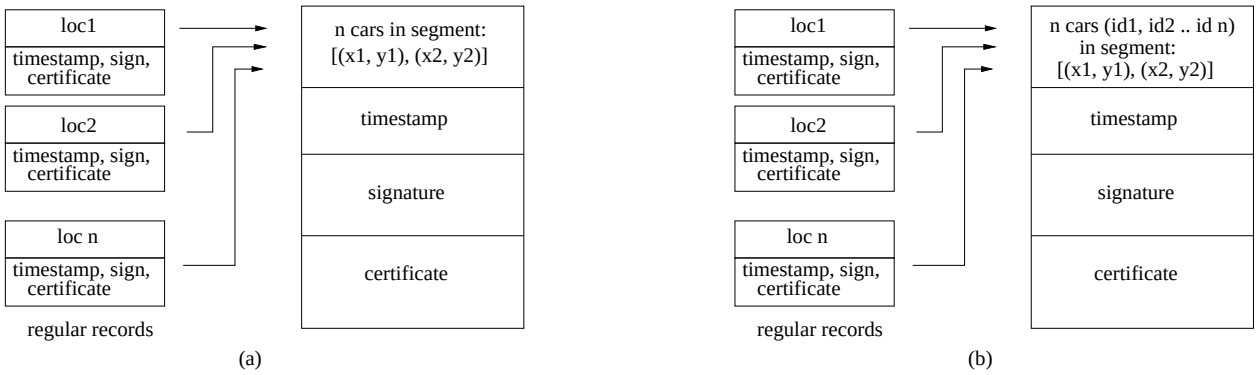


Figure 4: Construction of semantically aggregated records: more aggregation in (a) than (b) where car identities are retained

location or speed by rounding off real numbers to integers (e.g 65.00023 → 65), in order to save bandwidth. Similarly, the application could only extract and store a subset of the information present in the original records (as shown in Figure 3(b)). The car id is a small hash (e.g., 16 bits) of the certified key used for signing the record. Using 16-bit hashes keeps message overhead low while achieving an acceptable collision probability (0.07% when aggregating 10 records, and less than 2% for 50 records).

While *syntactic aggregation* compresses data to some extent, it mainly reduces header overhead. Reducing header overhead is valuable for low-bandwidth streams of sensor information, where message content is limited and therefore packet headers are dominant.

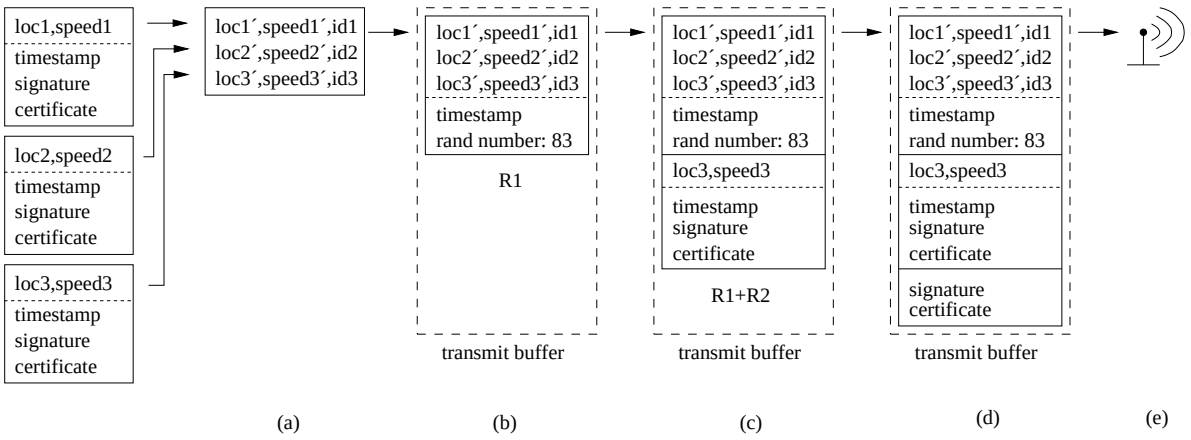
Figure 4 shows the structure of an aggregated record that contains only aggregated location information of multiple cars. Note that the information has been *semantically* aggregated. Instead of listing coordinates of individual cars, the number of cars on a road segment is reported. In Figure 4(b), the car identities are retained in the application data. Semantic aggregation saves more bandwidth than syntactic aggregation at the cost of loss of information (exact coordinates in this case). Also semantic aggregation is applicable to only certain information types (e.g location or speed) and is information specific.

In both syntactic and semantic aggregation, there is no authenticity of the data in the aggregated record. The reason is that the certificates corresponding to the individual records are not retained. A malicious aggregator could create the impression of a congested road by creating an aggregated record with car positions very close to each other, and low speed values. In this paper, we focus on syntactic aggregation which is more generally applicable. Our solution is also applicable to certain cases of semantic aggregation, in particular those where the resulting aggregate is the number of the elements in a set (as the example in Figure 4).

3. SECURE AGGREGATION

A naive solution is to implement a secure aggregation module inside the tamper-proof service and export it as an API to the applications. This implies standardization of aggregation and imposes restrictions on the data format, thereby compromising the flexibility of applications. Our design is motivated by the need to strike a balance between security, network overhead and application flexibility. In our design, applications can implement their own aggregation module, which creates the opportunity to save network bandwidth without compromising on flexibility.

In this section, we describe our solution for validating syntactically aggregated data. We also briefly outline how probabilistic validation can be applied to semantic aggregation



(a) the application extracts the data and car id from each regular record, and (b) puts it in the transmit buffer. The tamper−proof box appends a secure timestamp and the random number 83. (c) The application takes the regular record corresponding to entry i=(83 mod 3)=2 (i.e., the third entry), and appends it to the transmit buffer. (d) The tamper−proof box signs R1+R2 and (e) broadcasts the contents of the transmit buffer.

Figure 5: Secure aggregation

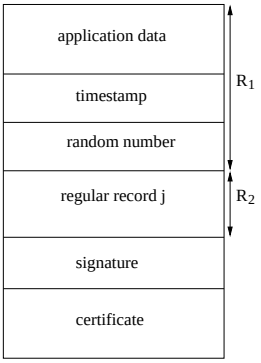


Figure 6: Structure of a secure aggregated record

of location information, if the identities of cars are retained in the application data (Figure 4(b)).

The basic idea behind our solution is to challenge the aggregator to provide a probabilistic proof of the integrity of the aggregated record, after the aggregated record has been sent. This proof consists of a randomly chosen subset of records among the ones that were aggregated, for which the aggregator must submit the original cerificates. Ideally, probabilistic aggregation validation should be trivial to implement as an end-to-end two-phase protocol over a reliable transport layer. However, reliable communication in extremely dynamic vehicular networks is hard if not impossible to achieve. Besides, a two-phase protocol would introduce additional forwarding latency in the network, as the receiver would have to wait for the proof to arrive before the aggregated record can be forwarded. In order to obviate the need for a two-phase protocol, we use the tamper-proof service in the car as a proxy for the receiver. As a proxy, the tamper-proof service plays two rolls. First, it provides a transmit buffer– data put on this buffer cannot be tampered with and *will be* transmitted. Second, it challenges the application to provide a randomly chosen original signed record to be sent with the aggregated data.

The aggregated record sent out using the tamper-proof service is required to contain two sub-records R_1 and R_2 (as shown in Figure 6). $R1$ contains aggregated application data, where the aggregation is carried out by the application, as described in the previous section. Figure 5 shows the different steps involved in secure aggregation. The application passes R_1 on to the tamper-proof service. The tamper-proof service appends a timestamp and a random number to R_1 and puts the resulting R_1 in the transmit buffer. The random number is used for the construction of R_2. The timestamp contains the current time value. Once R_1 is put in the transmit buffer, the application is committed to the aggregated record. (Recall that transmit buffer is a tamper-proof queue; data once put in this buffer *will be* transmitted).

R_2 is supposed to contain one of the original n records used for aggregation. This record is selected based on the random number r contained in R_1. The record with id $r \ mod \ n$ (where $r \in [0, 99]$) is selected and passed on to the tamper-proof service. The tamper-proof service attaches it to R_1 and signs $R_1 + R_2$. The resulting record is broadcasted.

An aggregated record is considered to be valid only if it contains a valid R_2. A valid R_2 must contain a valid signature, a timestamp value not older than a certain threshold value, and data in R_2 must match the data at index j in R_1 (where $j = r \ mod \ n$). The idea is that if the application has constructed R_1 with bogus values, it will, with a certain probability, be unable to provide a valid original record R_2. If the application does not send an R_2 to the tamper-proof service, the transmit buffer will eventually time out and R_1 will be signed and transmitted. An R_1 without R_2 would be considered invalid and the malicious car would be detected. Also, since R_1 has already been put on the tamper-proof transmit buffer, it is not possible for the application to modify it or prevent it from being sent out.

3.1 Probabilistic Validation

Figure 7 shows the different steps involved in probabilistic validation. When a car receives an aggregated record, it first verifies the signature and certificate and then checks for the presence of R_2. If R_2 is absent, it concludes that the sender is malicious. If R_2 is present, it validates R_2. For validating R_2, first the signature and certificate of R_2 are verified. Then, application data at index $i = r \ mod \ n$ in R_1 is compared with the application data in R_2. Optionally, the application data inside R_1 is cross-validated to check for the presence of anomalous values. Only if all the checks are successful, the aggregated record is accepted.

In the case of semantic aggregation of location information (Figure 4(b)), only the R_2 validation step differs. The car identity at index $i = r \ mod \ n$ in R_1 is compared with the car identity of R_2. If they match and if the location of the car in R_2 lies in the interval [(x1,y1), (x2, y2)], the aggregated record is accepted.

An aggregated record may have both real and bogus information. Since only a single original record is used to validate the aggregated record, if the random number corresponds to a record whose information is present in the aggregated record, the aggregator will be able to provide the corresponding record for R_2, and the attack will not be detected. Conversely, if the random number corresponds to a made up record or a record whose data has been distorted, the aggregator will be unable to provide a valid R_2, and the attack will be detected. Our validation method is therefore probabilistic in nature. The probability of a malicious aggregator getting caught is directly proportional to the amount of bogus information present in the aggregated record. The probability of catching the malicious car can be increased by generating multiple random numbers instead of just one, and asking the application to provide multiple original records in R_2. This would come at the cost of increased data overhead. We discuss this in Section 4.

Probabilistic validation of syntactically aggregated records could also be carried out by the tamper-proof service itself, before transmitting the aggregated record. This would further save network bandwidth. However, it would require applications to format application data in a specific way. In order to compare the application data in R_1 with that in R_2, the validation module needs to know the format, data type, organization and semantics of the application data (a bit-wise comparison is not possible as application data can be modified by the application during aggregation as shown in Figure 3). Only the application is assumed to have this knowledge, unless organization of application data is standardized. Also, cross-validation of application data in R_1 requires knowledge of the semantics of the data. The validation module should therefore be part of the application in accordance with the corresponding aggregation module. This is important for keeping the application design flexible.

For our solution to be effective, the penalty of getting caught should be high. Once an attack is discovered, cars should discard any further messages signed by the same attacker. They should also share the information about the attack with other cars, so as to isolate and paralyze the attacker. In order to be able to convince other cars that a particular aggregator is malicious, an honest car should also provide a proof of the attack. This is necessary in order to prevent cars from lying about other cars. In our scheme, the proof is the aggregated record itself. When a car announces

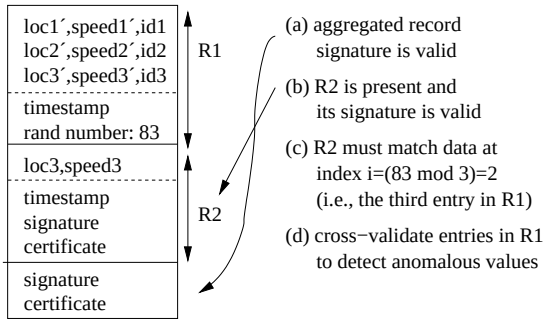


Figure 7: Probabilistic validation

an attack, it also forwards the aggregated record which was found to be bogus. In order to verify the announcement, other cars can carry out validation of the aggregated record themselves.

Since cars switch private keys at regular intervals of time to preserve privacy, an attacker who has been discovered may start sending false aggregated information as soon as its key changes. In order to counter this, we need to devise a scheme where a central server intermediates once a car has been identified as malicious. A brute-force solution would be to send a signal to the tamper-proof box so that it stops changing private keys. Note that the hardware in high-end cars is capable of receiving update/tuning signals from the car manufacturer over satellite. Another solution is to assume that the list of public keys built into the tamper-proof service of every car, is maintained by a trusted server (e.g the one that generated the public/private key pairs for all the cars). When a car is identified as malicious, the trusted server is contacted, and the list of all the public keys of that car are broadcasted to everyone, so that cars can ignore messages containing any of the *blacklisted* public keys.

3.2 Tamper-Proof Service

Signing, timestamp generation and random number generation are carried out by the tamper-proof service. It also provides a transmit buffer for the purpose of committing an application to an aggregated record. In this section, we describe how a tamper-proof service can be implemented using BIND [19], an enhancement of trusted computing.

Trusted Computing [2] is a hardware-based solution for security. The goal is to provide a mechanism for secure boot. At the heart of trusted computing is the idea of *attestation*. Attestation builds a certificate chain from a trusted firmware component all the way up to the operating system and the applications, in order to identify each component of the software stack. In the *active* version of trusted computing, the lower layer (e.g bootloader) allows the upper layer (e.g OS) to be loaded only if the hash of its binary matches the set of *known hashes* stored in a tamper-proof storage. In other words, trusted computing only allows well-known software (including applications) to execute. This leads to inflexibility not suitable for many computing scenarios such as the one being considered in the paper.

Recently, Perrig et al proposed BIND [19]. BIND is a software service that runs on a TPM enabled machine and provides fine-grained attestation. BIND offers two key properties: (1) partial attestation, and (2) data isolation. Using

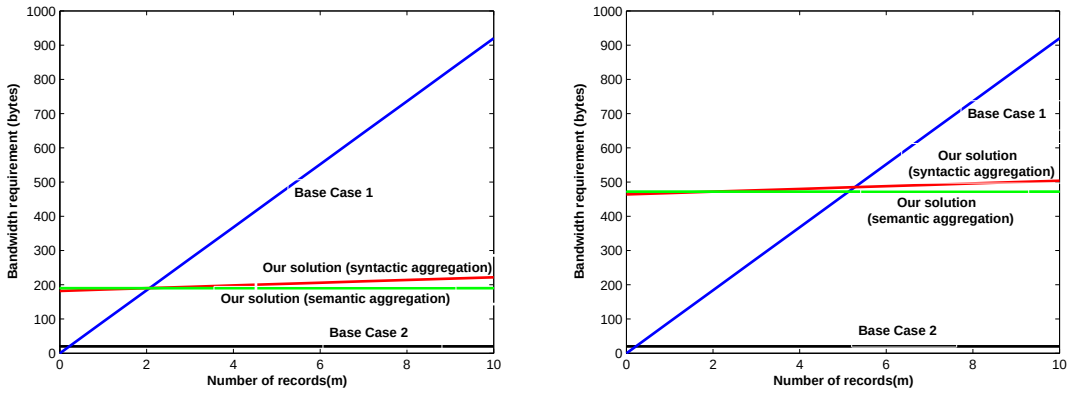


Figure 8: Bandwidth requirement of our solution compared with the two base cases (left: n = 1, d = 4 bytes, k = 16 bytes), (right: n = 4, d = 4 bytes, k = 16 bytes)

BIND, a piece of code can be selectively attested without having to attest all the applications. BIND attests the piece of code immediately before it is executed and uses a sandboxing mechanism to protect the execution of the attested code. This provides for tamper-proof execution of critical applications/services without affecting other applications. For more details, refer to [19].

BIND can be used for implementing the tamper-proof service as required by our solution. The tamper-proof service is responsible for carrying out the following operations: signing, time-stamping, generation of random numbers and providing a transmit buffer. These operations can be easily implemented in software as one single service and protected using BIND. The other applications do not have to be attested providing for greater flexibility.

The main component of the tamper-proof service is the transmit buffer. The transmit buffer is responsible for receiving and buffering a sub-record R_1 until one of two things occurs: the application provides the second sub-record R_2, or a timer T expires. In either case, a message is broadcasted to the network. In the first case, a signed record containing $R_1 + R_2$ is transmitted. If the timer expires and the application has not provided R_2, a record containing only R_1 is signed and broadcasted. The use of a secure transmit buffer is necessary to guarantee that a malicious application cannot modify or hold back R_1 once the random number is generated. Otherwise, it would chose to not broadcast R_1 if a valid R_2 cannot be constructed and will therefore never be caught.

The tamper-proof service exports an API to the applications, consisting of the following functions: *void sign(data)*, *int sendbuffer(data, timeout)*, *void appendbuffer(data, id)* and *int readbuffer(id)*. Function *sign(data)* accepts application data, timestamps it and signs it. It is used for generation of regular records. Function *sendbuffer(data,timer)* accepts application data, appends a timestamp and a random number to it, and securely buffers it until either the function *appendbuffer(data, id)* is called, or the timer expires. If the timer expires, the data is signed and broadcasted to the network. This function is used for buffering R_1 and returns an integer as a unique identity for the buffered data entity (i.e R_1). Function *appendbuffer(data, id)* appends data (i.e R_2) to the data entity (i.e R_1) identified by *id*, signs it, and broadcasts it. Function *readbuffer(id)* returns the random number associated with the buffered data entity identified by *id*. It allows applications to examine the transmit buffer contents to determine the random number and construct R_2 accordingly.

BIND provides secure storage to the tamper-proof service for storing keys and buffering data. Since the networking stack is part of the operating system, which is attested at boot-up time, *sendbuffer* and *appendbuffer* can safely use the networking stack for transmitting data.

4. PRELIMINARY EVALUATION

In this section, we define a metric for evaluating our solution and carry out an analytical evaluation.

4.1 Security vs. Bandwidth

One solution for security is to attach a timestamp, signature and certificate to every record and not carry out any aggregation. This is the highest security solution, which incurs high data overhead. The other extreme is to send unsigned semantically aggregated information (e.g average speed or number of cars in a segment). This saves bandwidth but provides no security. Our solution lies between these two extreme cases, both in terms of security guarantees and bandwidth requirements. A good security solution for vehicular ad-hoc networks should provide security with minimal data overhead. We identify *security/bandwidth*, or *sec/bw* (i.e security normalized by bandwidth requirement), as the correct metric for evaluating our solution.

We define the bandwidth requirement of our solution as the size of one secure aggregated record. Let d be the size of application data of a regular record. In the case of location and speed, d is equal to 4 bytes (2 bytes each for location and speed). The size of an unsigned regular record is therefore d bytes. We assume that timestamp size is 4 bytes, random number is 2 bytes, the signature is 28 bytes (using ECDSA [14]), and the certificate is 56 bytes (28 for the car's public key and 28 for the authority's signature). The size of a signed regular record is $d + 88$ bytes. A secure aggregated record is composed of R_1, R_2, signature and certificate (Figure 6). If R_1 contains information about

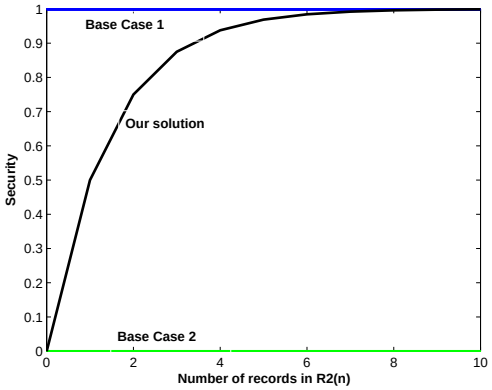


Figure 9: Security of our solution compared with the two base cases (f/m = 0.5)

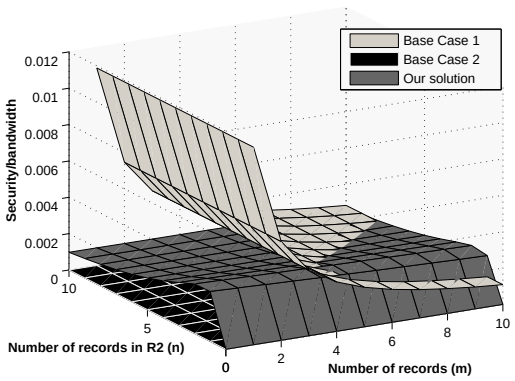


Figure 10: Security/bandwidth of our solution compared with the two base cases (d = 4 bytes, f/m = 0.5)

m cars, and we assume syntactic aggregation (which is the more general case) size of R_1 is given by $m*d+6$ bytes. Size of R_2 is $d+88$ bytes. Signature and certificate together are 84 bytes. The size of a secure aggregated record is therefore $(m*d+d+178)$ bytes. The probability of catching the malicious car can be increased by generating multiple random numbers instead of one, and including multiple original records in R_2. If we assume that there are n records in R_2 (and hence n random numbers in R_1), the size of a secure aggregated record would be $(m*d+n*(d+90)+88)$.

As mentioned before, one extreme case is to sign and certify every record and not carry out any aggregation. This provides high security at the cost of high bandwidth. We call this "Base Case 1". The bandwidth requirement of Base Case 1 is given by $m*(d+88)$. The other extreme case is to send unsigned semantically aggregated data. We call this "Base Case 2". The bandwidth requirement of Base Case 2 is $d+k$ where k is a small constant (e.g 10 bytes). The bandwidth requirement of our solution is $(m*d+n*(d+90)+88)$ in the case of syntactic aggregation which is the more general case. In cases where our solution could be applied to semantic aggregation, the bandwidth requirement would fall to $(d+k+n*(d+90)+88)$. Figures 8 shows how our solution compares with the two base cases in terms of bandwidth requirement. It is clear from the graphs that, for higher values of m, the bandwidth requirement of our solution is way less than that of Base Case 1 (even when the number of records in R_2 is increased to 4). Note that required bandwidth for syntactic and semantic aggregation is comparable.

We assume that security ranges between 0 and 1, with 0 standing for no security and 1 for the highest security. The security of Base Case 1 is 1, and that of Base Case 2 is 0. The security of our solution is defined as the probability of detecting a malicious car, and is given by $1-(1-f/m)^n$, where f is the number of false or made up values in R_1. Figure 9 compares the security of our solution with that of the two base cases for $f/m = 0.5$. When $n > 4$ the probability of detecting a malicious car is more than 90%. However, as evident from Figure 8(right), the bandwidth requirement of our solution when $n = 4$ is less than that of Base Case 1 for $m > 5$.

The value of sec/bw for Base Case 1 is $1/m*(d+88)$ and for Base Case 2 is 0. The value of sec/bw for our solution is given by $(1-(1-f/m)^n)/(m*d+n*(d+90)+88)$. Figure 10 compares sec/bw for the three solutions (with $f/m = 0.5$). For $m > 5$ a certain value of n always yields higher sec/bw value for our solution than the two base cases.

4.2 Computation time

Our aggregation mechanism also helps reduce the computation time required to verify received data. When no aggregation is used, each car must verify the signature of every individual record it receives from the cars ahead. For instance, if n records are forwarded across a road segment containing m cars, then the total number of signature verifications performed in that road segment would be $n*m$. If aggregation is used, the total number of signature verifications would be much smaller. Assuming that no new records are generated in that segment, only the aggregator must verify the signatures of the n records received from the segment ahead of it. The remaining $m-1$ cars in the segment will only carry out two verifications each, one for the aggregated record, and one for the randomly chosen verifier record contained in it. This implies that only $n+(m-1)*2$ verifications, instead of $n*m$, must be carried out. For example, when $n = 10$ and $m = 10$, the number of verifications drops from 100 to 28. Reduction of record verification time implies lower message forwarding latency and increased processor availability for other applications.

4.3 Limitations and Extensions

Currently our solution can be used for validating information about cars (e.g speed, location, etc) but not about road-side events (e.g accidents). Our solution handles modification of records or inclusion of fake ones; it does not handle omission of records (e.g. a malicious car aggregates or forwards only a subset of the records it receives). We rely on path redundancy and multiple aggregators to detect records dropped by an attacker. Also, reaggregation is not addressed. We believe that the idea of *random checks* can be further extended to handle reaggregation. This is left for future work. It is worth mentioning though, that the bandwidth savings due to reaggregation are not very significant.

Another limitation of our solution is that it does not provide a generic scheme to validate semantically aggregated data. Although Figure 8 shows that the required bandwidth for syntactic and semantic aggregation is comparable, it would be interesting to come up with a generic scheme for validating semantically aggregated data. Przydatek et al. [12] suggest using Probabilistically Checkable Proofs (PCPs) to validate semantic aggregation in sensor networks. However, their solution requires an interactive protocol between the aggregator and the verifier. This full-duplex communication does not seem feasible in a vehicular ad-hoc network. That said, it would be interesting to explore the use of simple random sampling [13], performed at the aggregator using secure random numbers, to validate the result of common aggregation functions, such as min, max, average, count, etc.

Finally, our solution relies on the presence of a tamper-proof service in each car. While this does not compromise flexibility significantly, it implies additional hardware cost.

5. RELATED WORK

Parno et al [10] suggest a set of security primitives for vehicular networks, such as message origin authentication, anonymization, and secure aggregation. Though insightful, the paper does not lay out a solution for securing aggregated data in vehicular networks.

Kuhn [11] suggests using non-uniform hash functions to probabilistically prove a lower bound on the number of unique valid signatures of a record. The scheme, however, provides only order-of-magnitude resolution, making it unsuitable when the number of signatures is small.

Capkun et al. [15] present a mechanism based on distance bounding [16] to securely determine the position of a node. Since it requires well-known reference points, the mechanism is useful for sensor networks, but difficult to apply in VANETs.

Golle et al [5] describe how cross-validating data from different cars can be used to detect false information, provided only a small number of cars are malicious. If the number of malicious cars in a region outnumber the number of honest cars, their solution would be insufficient. Also, the paper does not address validation of aggregated data.

Raya et al [14] argue that unauthenticated communication is too weak for vehicular networks. They suggest that every car should sign messages with a key certified by a trusted authority. They use a tamper-proof device to securely store the car's private keys, and sign messages. While this approach is generically applicable, if applied as such, it would incur high data overhead.

Several aggregation protocols have been proposed for sensor and vehicular ad-hoc networks. Aggregation is usually associated with functions that compute the min, max, average, median, and other basic functions on a set. In these protocols, the aggregated data is just the result of one or more of these functions. Most of these protocols assume that nodes are honest [8, 7] and do not employ any security mechanisms. Hu and Evans [6] propose a secure aggregation mechanism for sensor networks. However, being a sensor network design, it assumes a tree topology with a base station at the root. Also, several rounds of communication between nodes are required, making this protocol difficult to apply to vehicular networks. Przydatek et al [12] suggest using PCPs between a base station (the verifier) and an aggregator (the prover) in a sensor network environment. Although their protocol supports aggregation primitives such as min, max, median, and count-elements, it also requires several rounds of communication between the aggregator and the verifier.

In contrast, our approach minimizes data overhead while providing reasonable security guarantees, which was possible by the understanding of the communication protocol used in V2V traffic information systems.

6. CONCLUSIONS

In this paper, we have looked at the problem of validating aggregated data in V2V traffic information systems. We proposed a solution for this problem that incurs minimal data overhead and does not impose any additional communication steps between the sender and the receiver. The main idea is to use *random checks* to probabilistically catch the attacker. By making the penalty of getting caught high, our solution discourages spoofing and bogus information attacks. Our solution relies on PKI based authentication and assumes a tamper-proof service in every car to carry out certain secure operations, such as signing, timestamping and random number generation. We showed how such a tamper-proof service can be constructed using BIND [19] which is an enhancement of trusted computing. The applications and aggregation modules themselves do not have to be trusted providing for greater flexibility. We defined a new metric for evaluating our solution: security normalized by bandwidth, and showed that our solution scores high on the metric.

We are in the process of implementing this solution and integrating it with our traffic information system (TrafficView [8]). As part of future work, we plan to carry out an empirical and theoretical analysis of this solution to identify the equilibrium state.

7. REFERENCES

[1] Dedicated Short Range Communications(DSRC) http://grouper.ieee.org/groups/scc32/dsrc/

[2] Trusted computing platform alliance. http://www.trustedcomputing.org.

[3] John R. Douceur. The Sybil attack. In *First International Workshop on Peer-to-Peer Systems (IPTPS)*, March 2002.

[4] J. Newsome, E. Shi, D. Song and A. Perrig. The Sybil Attack in Sensor Networks: Analysis and Defenses. In *Proceedings of the Third International Symposium on Information Processing in Sensor Networks (IPSN 2004)*.

[5] P. Golle, D. Greene, and J. Staddon. Detecting and correcting malicious data in vanets. In *Proceedings of the first ACM workshop on Vehicular ad hoc networks*, 2004.

[6] L. Hu and D. Evans. Secure aggregation for wireless networks. In *Workshop on Security and Assurance in Ad hoc Networks*, 2003.

[7] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. In *Proceedings of the Fith Annual Symposium on Operating Systems Design and Implementation (OSDI)*, 2002.

[8] T. Nadeem, S. Dashtinezhad, C. Liao, and L. Iftode. Trafficview: Traffic data dissemination using

car-to-car communication. In *ACM Sigmobile Mobile Computing and Communications Review*, 2004.

[9] T. Nadeem, S. Dashtinezhad, C. Liao, and L. Iftode. Trafficview: Traffic data dissemination using car-to-car communication. In *IEEE International Conference on Mobile Data Management (MDM)*, 2004.

[10] B. Parno and A. Perrig. Challenges in securing vehicular networks. In *Workshop on Hot Topics in Networks (HotNets-IV)*, 2005.

[11] Markus Kuhn. Probabilistic counting of large digital signature collections. In *Proceedings of USENIX Security Symposium*, 2000.

[12] B. Przydatek, D. Song, and A. Perrig. Sia: Secure information aggregation in sensor networks. In *Proceedings of ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2003.

[13] Ziv Bar-Yossef, S. Ravi Kumar, and D. Sivakumar. Sampling algorithms: lower bounds and applications. In *Proceedings of 33rd Annual Symposium on Theory of Computing (STOC)*, 2001.

[14] M. Raya and J.-P. Hubaux. The security of vehicular ad hoc networks. In *Proceedings of ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, 2005.

[15] S. Capkun and J.P. Hubaux. Secure positioning of wireless devices with application to sensor networks. In *Proceedings of INFOCOM*, March 2005.

[16] S. Brands and D. Chaum. Distance-bounding protocols. In *Workshop on the theory and application of cryptographic techniques on Advances in cryptology*, 1994.

[17] Y. Hu, A. Perrig and D. Johnson. Packet Leashes: A Defense against Wormhole Attacks in Wireless Ad Hoc Networks. In *Proceedings of INFOCOM 2003*.

[18] J. Saltzer and M. Schroeder. The protection of information in computing systems. In *Proceedings of the IEEE*, 1975.

[19] E. Shi, A. Perrig, and L. van Doorn. Bind: A time-of-use attestation service for secure distributed system. In *Proceedings of IEEE Symposium on Security and Privacy*, 2005.