A Data-based Protocol for One-way Trust in Inter-vehicular Communication

Stephen Ly
California State University, Sacramento
Sacramento, CA, USA
stephenly@csus.edu

Yuan Cheng
California State University, Sacramento
Sacramento, CA, USA
yuan.cheng@csus.edu

ABSTRACT

As autonomous vehicles fill the roads and more manufacturers join the trend, the need for a unified communication protocol grows. Current paradigms in vehicle-to-vehicle communication are too slow to provide accurate and meaningful traffic data in a timely fashion, and it is difficult to trust that incoming data is correct without an authoritative server verifying the sender's identity. This paper introduces a protocol for peer-to-peer exchanges of positional data that determines the trust level of a particular message by comparing matching object data hashes. Similar in concept to non-interactive zero-knowledge proofs, the design retains the privacy and anonymity of senders and is relatively fast compared to certificate-based solutions under a reasonable traffic load. Our preliminary experiment shows promising results, with much faster runtimes compared to similar cryptographic solutions. Although the current implementation is still rough around the edges, the basic design can provide the groundwork for future paradigms in inter-vehicular communication without depending on expensive cryptographic operations performed on special or more powerful hardware. This opens doors for protocols that can be run on current vehicles without requiring the collective processing power of all vehicles to increase.

CCS CONCEPTS

• **Security and privacy** → **Privacy-preserving protocols**; **Trust frameworks**; • **Computer systems organization** → *Sensor networks.*

KEYWORDS

inter-vehicular communication, communication protocols, trust management, anonymity, autonomous vehicles

ACM Reference Format:
Stephen Ly and Yuan Cheng. 2021. A Data-based Protocol for One-way Trust in Inter-vehicular Communication. In *Proceedings of the 2021 ACM Workshop on Secure and Trustworthy Cyber-physical Systems (SAT-CPS '21), April 28, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3445969.3450430

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SAT-CPS '21, April 28, 2021, Virtual Event, USA
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8319-6/21/04…$15.00
https://doi.org/10.1145/3445969.3450430

1 INTRODUCTION

As advances in artificial intelligence (AI) have grown, their prevalence in our day-to-day life becomes more ubiquitous. One of the most promising uses of AI includes autonomous vehicles. Self-driving vehicles promise safer traffic conditions and reduced risks of human error. However, current vehicles are very short-sighted, relying only on what their own sensors can "see." Access to traffic data over a longer distance would allow the AI to make better decisions in the long run, but this would require a vehicle to obtain its data from an outside source. A continuous Internet connection can be difficult to maintain with today's infrastructure. And the lag incurred from sending and receiving packets over a network can render the data stale by the time it reaches another node, due to the volatile dynamics of moving traffic. Therefore, an ideal solution to sharing data between vehicles necessitates a direct peer-to-peer connection.

A direct connection between vehicles introduces its own problems. How do we trust that a data packet was sent by a trusted node and contains correct information? Traditional solutions usually rely on security infrastructures, such as a certificate authority (CA) or a key distribution center (KDC) [5]. These infrastructures provide varying levels of security in protecting communications but fail to address the specific requirements we want to target in our vehicular networks, namely *fast resolution*, *privacy protection*, and *independence from third-party servers.*

To address these requirements, we introduce a novel security protocol for determining the trust of a message using a method inspired by the concept of common reference strings from non-interactive zero-knowledge proofs. By utilizing the GPS and attribute data of vehicles as part of the verification process, we can ensure that the only nodes able to send an authentic message are ones with matching sensory hardware. Since we only intend to verify the accuracy of the message contents, it is unnecessary to identify the sender. This provides a level of anonymity and privacy not present in existing solutions. Moreover, as a vehicle can generate this message on its own, no third-party is required to facilitate the communication.

A preliminary performance study based on a prototype with no optimization shows that the average-case runtime of message creation and validation using our protocol comes below the 50 ms maximum latency time defined by ETSI [1] for safe intelligent transport system applications. The current implementation has issues with scaling, but setting hard limits on the number of objects used in message creation can mitigate this flaw. After all, the main contribution of this work lies in the invention of a paradigm for future protocols in inter-vehicular communication that fulfills the need for a fast, privacy-preserving method untethered by third-party servers.

The remainder of the paper is organized as follows. In Section 2, we discuss some work relevant to this study. Section 3 describes the overall design of the protocol, focusing on the basic requirements, assumptions, and considerations in the design. Section 4 details the components of each protocol message while Section 5 describes the steps taken in verifying these messages. We identify potential threats and vulnerabilities in Section 6 and detail the results of our prototype tests in Section 7. Section 8 discusses the limitations of our proposed protocol and suggests some future work. Section 9 finally concludes the paper.

2 RELATED WORK

Using attribute data as a basis for inter-vehicular communication is not a new concept. Many schemes make use of it, as it provides a level of privacy not found in traditional vehicle-to-vehicle protocols [2, 4, 6, 7]. Although these existing schemes have demonstrated how useful attribute data is, they fall short in one way or another.

One such authentication protocol using attributes was proposed by Han et al. [6]. This protocol uses attributes as part of encryption and signature generation, both of which require more computational power. The authors themselves focus on data throughput in their performance analysis, in contrast to our emphasis on execution time.

Zhang et al. proposed a privacy-preserving sharing scheme that also uses attribute-based cryptography [7]. This system is closest to the purpose of our own protocol, focusing on sharing data between vehicles. The proposed system boasts a constant time complexity in decryption with respect to the number of attributes. However, the decryption time takes roughly 100 ms; and this does not count signing and encryption on the sender's side.

Neven et al. proposed a pseudonym-based credential system based on attributes [4]. Although it uses attributes like ours, it relies on a trusted certificate authority to work. Furthermore, even the authors refrain from prototyping their design, acknowledging that their implementation is not likely to meet any efficiency requirements of inter-vehicular communication.

3 PROTOCOL DESIGN

To implement our protocol, we first define some assumptions and requirements for ensuring the message security.

3.1 Basic Requirements

The first requirement is having access to an accurate and precise timestamp source. The main reason for this requirement is that we want all nodes to have a synchronized clock with a high degree of precision. The synchronized clock allows us to perform a more accurate position prediction at a particular time. The most accessible source comes from GPS satellites, which provide not only latitude and longitude coordinates but time data as well. In our design, the nodes do not need to constantly ping for time, as the internal clock can take over for a period of time after the clock is synchronized.

Accurate and precise GPS positions are the next requirement for the operation of our protocol. In this case, we want the GPS locations for both the sending vehicle and surrounding objects. Like timestamps, we do not necessarily have to ping the GPS satellites for coordinates for every message. Once a vehicle has obtained its precise coordinates, it can simulate an accurate GPS position based on the vehicle's speed and moving direction. Moreover, it is required to have GPS coordinates for objects in the sender's surrounding area, but we can calculate these coordinates based on their relative positions with respect to the sender.

Our third requirement forms the crux of our protocol. For our protocol, we require that the observable objects have measurable traits and attributes that would be reasonably difficult to measure or guess without the proper sensors or hardware. This attribute data forms a reference string that tells a receiving vehicle that the object in the message is the same object that it is aware of. Any number of attributes can be used, such as object type, color, and bounding box dimensions. However, for the protocol to work, a participating node must be able to populate these attribute fields for each object it trusts. This means that attributes such as vehicle identification numbers and license plate numbers are not admissible, since it might not be possible for a vehicle to be able to ascertain these attributes. In order to break the protocol, the entire set of attributes must be uncovered. Therefore, the more attributes used in the protocol, the harder it is for an adversary to brute-force.

3.2 Trusted Objects

To meet the third requirement of having observable objects with attribute data, our nodes need to maintain a list of objects whose data they can personally vouch for.

To consider an object a trusted object, a vehicle must be able to discern all of the object's positional and attribute data. Trusted objects and their relevant data will be added to a list, which becomes the base truth that incoming message objects are compared to.

3.3 Privacy and Anonymity Considerations

Privacy is one of the main issues we want to tackle here. Our design addresses it in several ways.

No personally identifiable information (PII) is included in messages. Attribute data from each object and the sender itself might be considered unique to each individual vehicle, but that data is hashed along with a timestamp and a nonce, creating a digest that is only unique to that particular message.

To preserve anonymity, we must ensure that the sender of a message cannot be identified by any protocol-specific leaks. Since our protocol is essentially a one-way single message broadcast, there are a few things we skip. For example, no reply is required, thus the source IP or MAC addresses are not needed to facilitate the communication. Destination addresses are not needed either, as the message is a broadcast. The most secure way to implement anonymity would be to have a separate channel dedicated to the sending and receiving of these messages over the radio, but we leave the implementation details to manufacturers.

Since the sending vehicle is considered a trusted object, its attributes are hashed and sent along with the message as well. Among a reasonably sized set of trusted objects, the sender's attribute data is indistinguishable from other trusted objects' attribute data. There are of course edge cases where anonymity is impossible (e.g., the sender is the only object in its field of view). However, the goal is to provide an accurate representation of current traffic, not complete secrecy.

4 MESSAGE STRUCTURE

Messages in the protocol are formed using a variety of data, each having its own role in the functionality of the protocol as a whole.

4.1 Timestamp

In our protocol, a timestamp is used to determine the age of an incoming message in the verification stage. If a message is older than our defined maximum lifetime, we simply discard it.

The timestamp is also used to verify positions of objects over time with the help of positional vectors (we will define them in Section 4.3). The position of each object is calculated from these positional vectors, creating a GPS prediction for the timestamp that is matched with the sender's trusted objects.

In our prototype, we use a value with 8 significant figures and millisecond precision denoting the number of seconds since the start of the day. A typical timestamp ranges from 00000.000 to 86399.999.

4.2 Nonce

A nonce is included in the message creation process. Its purpose is to introduce entropy into the attribute hashes we create. Without a nonce, different messages generated at the same time may have identical hashes for the same object.

The nonce is also used to ensure that a message is not replayed or reused. Our protocol expects each incoming message to have a unique nonce over a period of time.

Our prototype uses an 8-byte randomized string as a nonce. The security of the nonce is not vital as long as collisions are mitigated.

4.3 Positional Vectors

Positional vectors include GPS position, velocity, and acceleration. They are used in combination with timestamps to match object positions between the sender and receiver.

A position match tells the verifier that both the sender and receiver have eyes on the same object, and thus the receiver should compute a hash and verify that the computed hash matches the hash in the message.

For message creation, only the GPS coordinates need to be included. The receiver can use the timestamp and its own positional vectors to determine whether object positions match.

Our prototype uses up to 5 decimal places in our GPS positions, which roughly achieve a precision of 1 m. Ideally, a complete solution would utilize 6 decimal places to get to 10 cm precision.

4.4 Attribute Hash

Attribute hashes form the security of a message. To create a hash, the message creator goes through each of their trusted objects and creates a unique hash using each object's attribute data, the current timestamp, and a nonce. The reason we create separate hashes for each object is to be able to individually verify whether each object matches, creating a metric with which we calculate our trust value. This would not be possible with a single hash that encompasses all attributes since such a setup would require all objects to match, which is extremely unlikely.

Since the attribute data is not actually shared over the communication channel and we assume that attackers cannot generate or guess attribute data in time, the only nodes that can generate this hash are ones that can observe the object in question. If an object is in both the sender and receiver's trusted object lists, both of them can generate the hash given the same timestamp and nonce.

Any set of attributes will work for this as long as all participating nodes use the same set. For our prototype, we elect to use two attributes, bounding box width and bounding box length.

4.5 Combined Structure

We concatenate all the parts mentioned above into a single message string consisting of a timestamp, a nonce, and a list of positions and hashes for each object. Equation 1 shows the full structure of a protocol message. Each additional trusted object appends its position and hash to the end of the message.

$$\begin{aligned} &Timestamp||Nonce||Position_{Obj_1}||Position_{Obj_2}||...||Position_{Obj_N} \\ &||HASH(Attributes_{Obj_1}||Timestamp||Nonce) \\ &||HASH(Attributes_{Obj_2}||Timestamp||Nonce) \\ &... \\ &||HASH(Attributes_{Obj_N}||Timestamp||Nonce) \end{aligned} \tag{1}$$

5 MESSAGE VERIFICATION

Since we are more concerned with the evaluation of the protocol on its own, we will not go into the details on how a message is actually sent from a sender to a receiver. Instead, we will take a look at how a message is verified and trust is assigned once the message reaches a participating node.

When an incoming message is received, the first thing that happens after parsing is to check the validity of the timestamp. This is to ensure that the message does not contain an invalid timestamp or stale data. If the timestamp is not within the valid range, we simply drop the message at that stage. For our prototype, we set the maximum lifetime of a message to be 200 ms.

After confirming the timestamp is valid, we begin iterating through each of the GPS positions in the message. For each object position in the message, we go through the receiver's trusted object list looking for a match, using our velocity and acceleration vectors to adjust to the timestamp of the message.

If we find a GPS position match, we generate a hash for that object in the receiver's trusted object list using the message timestamp, nonce, and the attribute data. If the hash in the message matches the hash generated by the receiver for that object, we increment a counter to keep track of how many matching objects the sender and receiver share. If the hashes do not match, we abort the verification process for that message immediately. Although a genuine error may have occurred to result in a mismatch between the attribute data, it is safer to simply drop the message.

If the verifier finishes processing the message, it will end up with a value representing the number of hash matches.

How we determine the trust level of the message contents using this number is still a rough work in progress. For our prototype, we go with a very simple equation that grades the trust level on a scale from 0 to 100.

$$Trust\ level = \frac{\#\ of\ hash\ matches}{\#\ of\ trusted\ objects} * \frac{(0.2 - \Delta Timestamp)}{2} \quad (2)$$

The maximum trust value attained with Equation 2 is 100, but the probability of this happening is very small, as it requires not only a 100% object match ratio, but also a time difference of 0 ms between the message and the verifier's timestamp.

What we do with this trust value is not covered in the scope of our prototype, but what we can do is to add all object positions from the message to a separate untrusted object list, and assign these objects an additional numerical field corresponding to the trust level we granted to the message. Multiple messages that corroborate the presence of an object at a specific location would further cement the algorithm's confidence in the existence of the said object.

6 THREAT MODEL

Using a list of potential inter-vehicular communication adversaries and threats from Jamthe et al.'s work [3], we evaluate how well our own protocol fares.

6.1 Adversaries

We first identify possible adversaries to our protocol and determine what level of risk these agents pose to our protocol.

6.1.1 Greedy Drivers. Greedy drivers are essentially nodes that are able to participate in the protocol, but do so selfishly, manipulating data for their own benefit. We have established that in order to create and verify messages, a node needs the ability to generate the appropriate attributes with sensors and GPS. If a greedy driver is itself a vehicle with the same capabilities, it will already have full access to the data behind each message. However, the security impact on our protocol due to this is negligible .

Because messages are sent one-way and broadcast anonymously, attacks on messages sent from other nodes are limited to data mining and static analysis attacks. The only data sent in plaintext are timestamp, nonce, and object positions, each of which a greedy driver already has access to. Cracking the hashes might have some merit. However, since all the attribute data should be publicly observable by all nodes, a greedy driver does not really gain anything from learning this information.

In message creation, a greedy driver has the ability to change timestamp, nonce, and object data. Here, greedy drivers are limited in scope as to what values they can safely change. An invalid timestamp or hash immediately results in the discarding of the message, so the attacker is limited to adding and removing specific object data to and from the message.

Possible attacks a greedy driver can carry out with these limitations include denial of service (DoS) attacks and Sybil attacks. To carry out a DoS attack, a greedy driver can append countless object records onto the list that have positions outside the field of view of the intended recipients. This causes the verifiers to have to check these entries for matches even if they do not exist. For Sybil attacks, a greedy driver can selectively insert a fabricated object with the intent of having a target vehicle trust in its existence. However, it is difficult to mitigate Sybil attacks without compromising anonymity and privacy.

6.1.2 Snoopers. Snooping in on an already created message gives very little information to attackers. And snoopers are one of the adversaries that are most affected by our privacy and anonymity considerations. Since it is difficult to link messages to senders or even data to individual vehicles, the type of data a snooper can learn from a message is limited to merely the GPS locations of objects at a specific timestamp, which are essentially public records we want to disseminate anyways.

6.1.3 Industrial Insiders. Because we limit our contact with third-party servers, there is little risk of attacks from industrial insiders. Such attacks would have to come before the implementation of the protocol to begin with, which is not in our scope. A possible avenue of attack would be to target GPS communications to disturb our basic requirements for accurate and precise timestamps and positions, but doing so would not go unnoticed.

6.1.4 Hackers and Malicious Attackers. We group hackers and malicious attackers together due to their shared interest in disrupting and harming communications and services despite their differing motivations.

Because our protocol is designed to be fast, possible attacks on messages in transit are very unlikely. This leaves message fabrication attacks as a possible attack vector for our adversaries.

6.2 Attack Types

In addition to identifying adversaries, we will also evaluate our protocol's resilience to a variety of attacks, as well as provide possible fixes when applicable.

6.2.1 Denial of Service. We have established that DoS attacks are likely to be the most common vulnerability our protocol has in its current implementation. The verification process we describe does not check that the objects in the message actually exist, and it is difficult for our receiver to verify existence if the object is outside of its sensor range.

While we cannot prevent DoS attacks entirely, what we can do is to mitigate the impact it has on our resources. Our current implementation saves computationally expensive operations for object position matches only, so dummy messages without matching objects are the fastest to resolve, taking only a fraction of the time a valid message would. There is still an overhead in dealing with malicious messages. However, even the most DoS-resistant systems require some time to decide to drop a packet.

6.2.2 Message Suppression. Message suppression involves blocking messages in transit. For our protocol, a message suppression attack is an all-or-nothing affair, as the protocol consists of a one-way message with no expectation of a reply.

The impact of a message suppression attack is that a receiver does not get a message, denying the receiving vehicle the extra traffic data. However, vehicles still have access to their own sensors and data, so they should be able to make informed decisions, albeit of a lower quality.

6.2.3 Fabrication Attacks. Fabrication attacks are one of the few ways for an attacker without sensors to attack our protocol. Since our communication channels are not secured, anyone is able to inject a message.

Although attackers can craft their own messages, unless they have access to attribute data, they will be unable to manipulate a receiver into trusting their messages. The most they can pull off without correct hashes are DoS attacks discussed earlier.

6.2.4 Spoofing and Masquerading Attacks. Because our messages are broadcast anonymously, there is no risk of spoofing or masquerading attacks. Certainly if every node is receiving the same messages from identifiable senders, it is pointless to pretend to be a node you are not. This leaves us open to fabrication attacks as we have seen earlier, but it does not compromise our trust model.

6.2.5 Eavesdropping. Due to our privacy considerations and message structure, there is no private information leak in our protocol. Everything in the message is either public or publicly observable.

6.2.6 Alteration of Data. This type of attack can be damaging if successful due to how our trust levels are determined. An attacker can intercept a message, change the positions of specific entries in the object list, and resend them to the receiver. Since the majority of the object data and hashes come from a good node, trust may be assigned to the altered object positions, which may not exist.

The reason this type of attack is unlikely to succeed is because not only does an attacker have to receive the message, alter it, repackage it, and resend it, it has to do this in the time it takes for the original message to get from the sender to the receiver. Once the receiver gets the original message, the window for the attack is over, as the nonce in the message is only applicable once. Changing the nonce would only invalidate all the hashes in the message, making the entire message untrusted.

A man-in-the-middle attack is possible by suppressing the original message, preventing it from being received by the verifier before sending the malicious message. However, timing is the key again, as the longer it takes for the message to travel from the sender to the receiver, the less trust is assigned to the message as a whole, with a good chance that the message is dropped completely due to an expired timestamp.

7 EXPERIMENTAL EVALUATION

7.1 Methodology

To evaluate our protocol, we develop a prototype using a basic set of data to handle message creation and message verification. Any secure hash function can be used, but we elect to use MD5 due to ease of use. MD5 is neither the fastest nor the most secure hash algorithm out there, but its weaknesses do not compromise the security of the protocol. Our experiments are run on a 2.50 GHz processor on a single thread in Windows 10.

To perform our performance analysis, we run both message creation and message verification 1000 times each, increasing the number of trusted objects in each run. The number of trusted objects we test range from 5 to 70, and we measure the average runtime for each setting separately.

For message verification, we run three different tests simulating worst-case, best-case, and average-case scenarios for varying numbers of position and hash matches. Since costly hash functions are run only when GPS positions in the message match the receiver's positions, our worst-case runtimes come from scenarios where every object in the message matches with an object in the verifier's trusted object list. Conversely, our best-case runtimes come from scenarios where no object in the message matches with the verifier's object list.

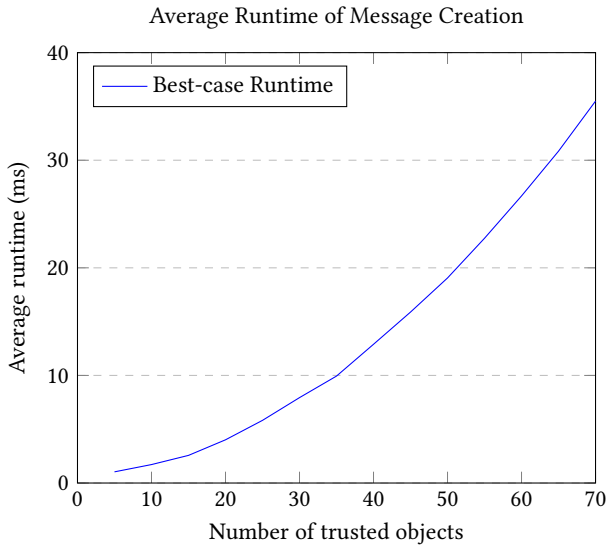

Figure 1: Message creation runtime for N trusted objects (in milliseconds)

7.2 Message Creation Evaluation

The evaluation results of message creation are shown in Figure 1. We find that message creation runs at linear time complexity with respect to the number of trusted objects.

Although we take measurements for up to 70 objects, it is unlikely to actually have that many detectable trusted objects in the vicinity of the sender in practice. A realistic scenario would be around 20 objects, for which our runtime is 4.0 ms on average.

7.3 Message Verification Evaluation

As shown in Figure 2, our current implementation of message verification runs in polynomial time complexity in a worst-case scenario, mainly due to the nested loop we utilize to match positions. There are a few scenarios that will break the inner loop early, such as when an object match is found. Hence, the actual runtime will typically be much less than the worst-case runtime.

A typical message from a benign node will have a runtime similar to the average-case line, which is comparable in runtime to message creation. The range of our runtimes might look large, but the edge cases that represent these lines are very unlikely in normal use.

Not shown in Figure 2 is the time it takes to process a message with an expired or invalid timestamp. Through our tests, we estimate the average runtime in this scenario to be 0.016 ms.

Due to the polynomial time complexity, the verification performance suffers heavily when there are more objects in each trusted object list. If we judge our performance on the realistic value of 20 objects, we can guarantee a runtime of at most 4.4 ms.

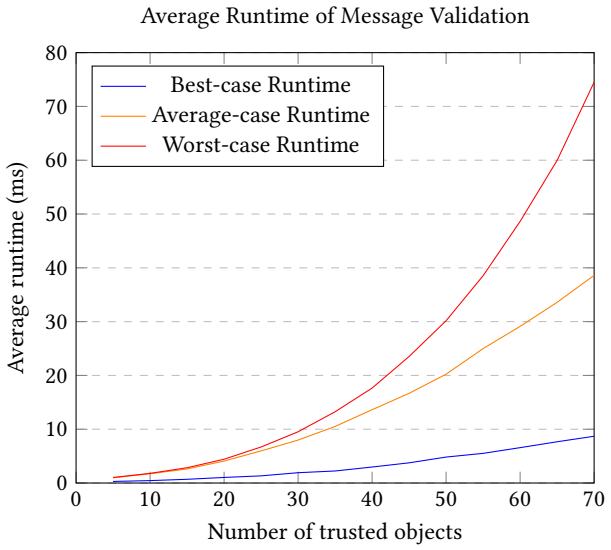

Figure 2: Message verification runtime for N trusted objects (in milliseconds)

7.4 Results

While the scalability of our protocol is not ideal, even at the maximum object count we tested, our average runtimes are still faster than the most recent similar work. Zhang et al.'s attribute-based sharing framework [7] features a constant runtime for the decryption of their transmitted message, but this constant time runs in excess of 100 ms. Even at the maximum traffic density we tested, the average runtime of our message verification process comes out to 38.6 ms, with even the total time of message creation and verification coming well below 100 ms at 74.1 ms. If we take a look at a more reasonable object count like 20, the runtimes of our message creation and verification come down to 4.0 ms and 4.1 ms, respectively, multiple times faster than the implementation used in their paper.

The speed of our protocol allows us to meet the ETSI maximum latency times for all the use cases described in the technical report on intelligent transport systems [1]. These times range from 50 ms to 500 ms for various safety applications, a figure our protocol clears in all but the most extreme of scenarios.

Not counting for transmission time, our prototype exhibits decent performance given a typical traffic scenario despite lacking optimization and algorithm refinements. With further work, we can potentially save more time while keeping the same computational power level.

8 LIMITATIONS AND FUTURE WORK

Many of the limitations of our protocol are built into the basic requirements themselves. The need for precise and accurate timestamps, GPS positions, and attribute data also necessitates that nodes be uniform in their capabilities and measuring capacity, which may not always be possible due to differences in manufacturers. The requirement that only nodes be able to discern attribute data also hinges on the assumption that attackers would not eventually be able to reach the same technological level.

Scalability is also an important limiting factor to the performance of our protocol. As technology advances and sensors are able to detect objects farther away, it only stands to reason that our trusted object lists will grow. Our protocol is unfit to handle these large lists in an acceptable time frame without further optimization. One possible solution would be to limit the amount of objects that are used in message creation and verification, which would set a hard cap on the runtime of our algorithm.

Optimization is a prime target for future work. There are a couple of ways to change the algorithm that can result in better performance and efficiency. We could implement positional verification checks, which consider only objects that are within an expected object area. Another significant improvement would be to check not only for object existence, but object absence as well, fortifying the protocol against attacks attempting to alter or insert bogus objects.

Not described in our project is exactly how we intend to send the messages created. We specify that our ideal method is an anonymous broadcast scheme, but do not clarify further. One possible avenue for future work would be to define the specifications for such a network transmission scheme for sending and receiving messages with anonymity and low latency.

9 CONCLUSION

Our protocol for one-way trust establishment for inter-vehicular communication manages to fulfill all three requirements outlined in the problem statement, featuring a fast and private way to determine the trust level of an object list without the need for an authoritative third-party to facilitate communications.

This is not to say our solution is perfect though. As computational power increases, the viability of traditionally slow cryptographic communication schemes will increase proportionally as well. Since our protocol loses effectiveness as the net computational power of malicious agents increase, there will be a point where more secure communication protocols will provide better values. But until then, our protocol's trade-off in favor of speed makes it well suited to spearheading inter-vehicular communication protocols in time for the automated vehicle boom.

REFERENCES

[1] TCITS ETSI. 2009. *Intelligent transport systems (ITS); vehicular communications; basic set of applications; definitions.* Technical Report ETSI TR 102 6382009. European Telecommunication Standards Institute.
[2] Liting Huang. 2012. *Secure and privacy-preserving broadcast authentication for IVC.* Master's thesis. University of Twente.
[3] Dinesh V. Jamthe, D. B. Khadse, and Yashwant Malode. 2016. Security Requirements for IVC Network. *International Research Journal of Engineering and Technology (IRJET)* 3, 2 (2016).
[4] Gregory Neven, Gianmarco Baldini, Jan Camenisch, and Ricardo Neisse. 2017. Privacy-preserving attribute-based credentials in cooperative intelligent transport systems. In *2017 IEEE Vehicular Networking Conference (VNC).* IEEE, 131–138.
[5] Philipp Wex, Jochen Breuer, Albert Held, Tim Leinmuller, and Luca Delgrossi. 2008. Trust issues for vehicular ad hoc networks. In *VTC Spring 2008-IEEE Vehicular Technology Conference.* IEEE, 2800–2804.
[6] Han Yiliang, Lin Xi, Jiang Di, and Fang Dingyi. 2017. Attribute-based authenticated protocol for secure communication of VANET. In *2017 29th Chinese Control And Decision Conference (CCDC).* IEEE, 4078–4081.
[7] Leyou Zhang, Jun Wang, and Yi Mu. 2020. Secure and Privacy-Preserving Attribute-Based Sharing Framework in Vehicles Ad Hoc Networks. *IEEE Access* 8 (2020), 116781–116795.