DEGREE PROJECT IN COMPUTER SCIENCE AND ENGINEERING, SECOND CYCLE, 30 CREDITS

STOCKHOLM, SWEDEN 2020

Efficient, Scalable and Secure Vehicular Communication System: An Experimental Study

SHUBHANKER SINGH

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Author

Shubhanker Singh <shusin@kth.se>

Examiner

Panagiotis Papadimitratos <papadim@kth.se>

Supervisor

Hongyu Jin <hongyuj@kth.se>

Abstract

Awareness of vehicles' surrounding conditions is important in today's intelligent transportation system. A wide range of effort has been put in to deploy Vehicular Communication (VC) systems to make driving conditions safer and more efficient. Vehicles are aware of their surroundings with the help of authenticated safety beacons in VC systems. Since vehicles act according to the information conveyed by such beacons, verification of beacons plays an important role in becoming aware of and predicting the status of the sender vehicle. The idea of implementing secure mechanisms to deal with a high rate of incoming beacons and processing them with high efficiency becomes a very important part of the whole VC network.

The goal of this work was to implement a scheme that deals with a high rate of the incoming beacon, preserve non-repudiation of the accepted messages which contains information about the current and near-future status of the sender vehicle, and at the same time keep the computation overhead as low as possible. Along with this, maintaining user privacy from a legal point of view as well as from a technical perspective by implementing privacy-enhancing technologies. These objectives were achieved by the introduction of Timed Efficient Stream Loss-Tolerant Authentication (TESLA), periodic signature verification, and cooperative verification respectively. Four different scenarios were implemented and evaluated, starting and building upon the baseline approach. Each approach addressed the problems that were aimed at this work and results show improved scalability and efficiency with the introduction of TESLA, periodic signature verification, and cooperative verification.

Keywords

Vehicular Public-Key Infrastructure (VPKI), Vehicular Communication (VC), Beacon, Pseudonymous Certificate (PC), TESLA, Signature Verification, Cooperative Verification

Sammanfattning

Medvetenheten om fordons omgivande förhållanden är viktig i dagens intelligenta transportsystem. Ett stort antal ansträngningar har lagts ned för att distribuera VC system för att göra körförhållandena säkrare och effektivare. Fordon är medvetna om sin omgivning med hjälp av autentiserade säkerhetsfyrar i VC system. Eftersom fordon agerar enligt den information som förmedlas av sådana fyrar, spelar verifiering av fyrar en viktig roll för att bli medveten om och förutsäga avsändarfordonets status. Idén att implementera säkra mekanismer för att hantera en hög frekvens av inkommande fyrar och bearbeta dem med hög effektivitet blir en mycket viktig del av hela VC nätverket.

Målet med detta arbete var att implementera ett schema som behandlar en hög hastighet för det inkommande fyren, bevara icke-förkastelse av de accepterade meddelandena som innehåller information om den aktuella och närmaste framtida statusen för avsändarfordonet och samtidigt håll beräkningen så låg som möjligt. Tillsammans med detta upprätthåller användarnas integritet ur juridisk synvinkel såväl som ur ett tekniskt perspektiv genom att implementera integritetsförbättrande teknik. Dessa mål uppnåddes genom införandet av TESLA, periodisk signatur verifiering respektive samarbets verifiering. Fyra olika scenarier implementerades och utvärderades med utgångspunkt från baslinjemetoden. Varje tillvägagångssätt tog upp de problem som riktades mot detta arbete och resultaten visar förbättrad skalbarhet och effektivitet med införandet av TESLA, periodisk signatur verifiering och samarbets verifiering.

Nyckelord

Fordonsinfrastruktur med allmän nyckel, Fordonskommunikation, Beacon, Pseudonymt Certifikat (PC), TESLA, Signatur Verifiering, Kooperativ Verifiering

Contents

1 Introduction **1**

1.1 Problem Area . . . 3
1.2 Related Works . . . 5
1.3 Contribution . . . 6
1.3.1 Purpose . . . 6
1.3.2 Goal . . . 7
1.4 Sustainability and Ethics . . . 7
1.5 Limitations . . . 7
1.6 Outline . . . 8

2 Background **9**

2.1 Vehicular Public-Key Infrastructure (VPKI) . . . 9
2.1.1 Overview . . . 10
2.1.2 Key Concepts . . . 11
2.1.3 Requirements . . . 14
2.2 Assumptions . . . 16
2.3 Vulnerabilities . . . 17
2.4 Adversary model . . . 18
2.4.1 Denial-of-Service (DoS) attack . . . 18
2.4.2 Message Tampering . . . 19
2.4.3 Forgery . . . 19
2.4.4 Replay Attacks . . . 19
2.4.5 Multiple Adversarial Nodes . . . 20
2.4.6 Sybil Attacks . . . 20
2.5 Tools . . . 20

3 Method **22**

3.1 Overview of the scheme . . . 22

3.2 Using Cryptography Library 23
3.3 Creation of Pseudonymous Certificate (PC) 24
3.4 Timed Efficient Stream Loss-Tolerant Authentication (TESLA) 25
3.5 TESLA key chain generation 25
3.6 Generation of Beacons 26
3.7 Validation Types 29
3.8 Observed Scenarios 30
3.8.1 Validation through Signature Verification only 30
3.8.2 TESLA key and Message Authentication Code (MAC) check . . . 30
3.8.3 Periodic Signature Verification 32
3.8.4 Cooperative Verification 32

4 Experimental Results 34
4.1 Test Bed 34
4.2 Performance Evaluation 34
4.2.1 Validation through Signature Verification only 35
4.2.2 Validation through TESLA key and MAC 35
4.2.3 Validation through Periodic Signature Verification 38
4.2.4 Validation through Cooperative Verification 40

5 Discussion 43
5.1 Analyzing the Scenarios 43
5.2 Future Work 45
5.3 Conclusion 46

References 47

Acknowledgements

Writing this thesis has been both difficult and fun to do it and it has been a very interesting experience for me. Through all of this, there have been several helpful people I would like to thank for their support.

My deep gratitude first goes to Prof. Panagiotis Papadimitratos, who expertly guided me through my graduate education and who shared the excitement of this work. His unwavering enthusiasm kept me constantly engaged with my research and his personal generosity helped make my time at KTH enjoyable.

My appreciation also extends to my supervisor, Hongyu Jin, whose mentoring and encouragement have been very significant. His insights have been very valuable and make up for a large part of this work. He steered me in the right direction at all times and gave me great support throughout my work process.

Furthermore, I would also like to thank my friends Michał Winiarski, Vinayak Tejankar, and Carlo Alberto for giving me guidance, assistance, and sharing many laughs in the time that I spent with them. To my close friends from home, I am grateful to you all and I would like to thank each and every one of you for your help and for the great moments that we have had.

Finally, I would like to dedicate my heartfelt appreciation to my dear parents, my brother, Siddhant, and my cousins. It is because of your wholehearted support that I finished my degree and you continue to be my constant source of inspiration.

Acronyms

CA	Certificate Authority
CAMs	Cooperative Awareness Messages
CRL	Certificate Revocation List
DER	Distinguished Encoding Rules
DDoS	Distributed Denial of Service
DoS	Denial of Service
DSA	Digital Signal Algorithm
ECDSA	Elliptic Curve Digital Signature Algorithm
GPS	Global Positioning System
ITS	Intelligent Transport Systems
LTCA	Long-Term Certificate Authority
MAC	Message Authentication Code
OBU	On-Board Unit
OBUs	On-Board Units
PC	Pseudonymous Certificate
PCA	Pseudonymous Certificate Authority
PKI	Public-Key Infrastructure
PRA	Pseudonym Resolution Authority
RCA	Root Certificate Authority
RSA	Rivest–Shamir–Adleman
RSU	Road-Side Unit
RSUs	Road-Side Units
SHA	Secure Hash Algorithm
SSL	Secure Sockets Layer
TCs	Trusted Components
TESLA	Timed Efficient Stream Loss-Tolerant Authentication

TLS	Transport Layer Security
VC	Vehicular Communication
VPKI	Vehicular Public-Key Infrastructure
V2V	Vehicle-to-Vehicle
V2I	Vehicle-to-Infrastructure

Chapter 1

Introduction

In today's world, with the advancement of technology and an increase in the population of people around the globe, a significant increase in the number of vehicles can be seen on the roads. Vehicles today travel at high speeds and more and more people use them for transportation every day. With more people using their vehicles every day, the chances of congestion on roads increase multi-fold. With the sudden increase in private transport, traffic congestions have become an intolerable problem in urban areas. All the large and growing cities pose this serious challenge of road traffic congestion as it leads to an undesirable increase in journey times and prevents the free flow of traffic. These congestions lead to nothing but more traffic jams as experienced by most users today. Road traffic jams continue to be a major problem in most cities resulting in huge delays, fuel wastage, and increased frustration amongst users. Even the simplest of accidents or over-usage of the same path can lead to traffic jams. A major concern related to road traffic congestion is that it often leads to hazards or accidents. It goes without saying that with more congestion, the chances of an accident happening increases way more rapidly and becomes a case of an impending disaster to happen. These result in serious injures or even death in many cases.

Without proper further information, more vehicles join and eventually the traffic worsens. It, therefore makes sense, why the users want to know about their surrounding conditions in advance for transportation on roads in today's environment. In the past decade, several efforts have been put to mitigate this problem by placing information about directions, roads, blocks, electronic tolls, etc. every few kilometers on roadsides or by updating users about traffic conditions through different methods such as broadcasts, etc. Safety is one thing that should never be compromised

and intelligent transportation systems are the ones that possess the potential to enhance transportation safety and efficiency [1, 2]. With numerous technology to assist and manage transportation, vehicles today are equipped with computing and communication platforms with enhanced sensing capabilities.

With the emergence of vehicular technologies, such as active road-signs or toll collection, a lot of development and research effort has been put into VC [3]. The upcoming VC systems integrated with a credential management infrastructure (a Vehicular Public-Key Infrastructure (VPKI)) is a research area that is widely being practically deployed [4]. Since these systems are vulnerable to attacks intended for disruptions, with the recent developments in VC systems, it becomes important to implement strong and practical security and privacy enhancing mechanisms to such systems [3].

Adversaries, for example, can modify in-transit messages, forge messages, or inject bogus messages. They can even choose a combination of these actions and attempt to compromise the protocol that operates. In some cases, the adversaries could even interfere deliberately and prevent other devices from receiving or sending information. An attacker could spread false information which would add on to unnecessary processing of these bogus messages and in case of dense environments, where a clogging Denial of Service (DoS) attack could be mounted, it would introduce significant processing overheads and even missing out on processing authentic messages. Since vehicles act according to the information received from the beacons, the above attacks also make it very hard to have a good estimation on the status of the sending vehicle since subsequent beacons were either blocked, not received, or not verified.

Verification of beacons plays an important role in predicting the status of the sender vehicles, thus, keeping a check on the authenticity of beacons at a regular rate becomes a subject of great interest. The idea of implementing secure mechanisms to deal with a high rate of incoming beacons and processing them with high efficiency becomes a very important part of the whole VC network. This work presents and implements a scheme that validates beacons by introducing periodic signature verification, Timed Efficient Stream Loss-Tolerant Authentication (TESLA) and cooperative verification to deal with a high rate of the incoming beacon, preserve non-repudiation of the accepted messages which contains the status of the sender vehicle and keep the computational overheads low.

1.1 Problem Area

With Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication gaining importance in today's VC technologies, a range of applications come into play to enhance transportation safety and efficiency, as well as infotainment. These interesting applications could help the vehicle with traffic conditions, accident warnings, tracking speeds, parking spots, etc. However, with such a wide range of tools, the possibilities of attacks and abuses on such systems increase [3, 5].

Since these VC systems are vulnerable to attacks, the user's privacy can be jeopardized. This could be when different pseudonymous certificates can be linked based on timing information or when curious VPKI entities could give out important information that is not meant to be shared [4]. An adversary could even contaminate such VC networks with bogus information and make vehicles transmit and receive false information. Without the integration of strong and practical security and privacy enhancing mechanisms, VC systems could lead to anti-social behavior, questioning the very existence of such technology [3, 5]. On the other hand, the provision of privacy and security also increases the communication overhead as well as the computational overhead for the resource-constrained On-Board Units (OBUs) if the mechanisms used are not efficient [6, 7].

Authentic safety beacons ensure awareness among neighboring vehicles within VC networks and these beacons are communicated with the help of resource-constrained On-Board Unit (OBU)s. These beacons advertise the status of vehicle kinematics such as Global Positioning System (GPS) position, braking status, speed, etc. Based on this information the receiving vehicle estimates the current and the near-future status of the sending vehicle. Vehicles receive a huge amount of beacons continuously and based on these beacons the vehicles estimate the status of the sender, however, if the beacons are not periodically validated, the accuracy of these estimations can have high error rates leading to unsafe driving situations [8]. If a vehicle finds itself in a dense environment or when an attack like Denial-of-Service happens, the situation worsens since the OBU would fail to verify all the received beacons because of processing power limitations. In such situations, it is possible that authentic beacons are also dropped along with fictitious beacons [7], thus, leading to inaccurate estimations and failure as a safety-critical application. These conditions also result in significant delays in the verification of authentic beacons and affect the overall efficiency of the whole VC network.

The problem area that this work focuses on stems from the drawbacks of the recent work done in this field. The following describes the problems that this work addresses.

- Beacons are validated through signature verification only in many related works such as [5, 9] where the approach is to validate all the received beacons by signature verification. However, the verification of beacon signatures introduces significant processing overheads for resource-constrained vehicular OBUs [2, 7, 10]. Signature Verification is an expensive process and has a high computation overhead for both the sender and the receiver [11]. Thus, validating every beacon solely on the basis of signature verification is not the most efficient of approaches.

- The performance of any VC system suffers whenever it faces a high rate of incoming beacons. The resource-constrained OBUs need to validate these beacons in a fast and efficient manner or else the beacons would be dropped due to the large number [7]. When beacons are dropped, even legitimate beacons would be dropped in a bunch of bogus and authentic beacons. When legitimate beacons are dropped, it becomes harder for the vehicles to be aware of their neighboring vehicles, since it is through these legitimate beacons only that the vehicles are aware of their surrounding conditions. Hence, the functioning of vehicular communications under high-density environments is an issue of high interest.

- Similar related work in this field [7, 12, 13] introduce TESLA to cope with the high rate of incoming beacons in an efficient and timely manner. However, TESLA does not support non-repudiation of the data source [11]. This could result in an adversary injecting bogus information via the beacons since this information contained in the beacons was not verified. This makes it very hard for the receiver to estimate the status of the sender vehicles. A key performance metric for the beacon broadcast is the accuracy of the position estimate [8] and if only TESLA is used to validate all the beacons, then non-repudiation of accepted messages cannot be preserved after a few TESLA checks. This approach is not the most efficient of approaches.

This work focuses on experimental validation, performance evaluation, and develop upon the existing proposals in terms of security and privacy protection and improves upon the efficiency of such systems. The idea is to leverage traditional public-key cryptography and verify beacons with the help of signature verification, cooperative

message verification, and TESLA. This work extends the scheme of efficiently validating authentic beacons under a dense environment as discussed in [7] and introduces period signature verifications along with TESLA checks and cooperative verification.

1.2 Related Works

Traditional public key cryptography could provide authentication and integrity to safety beacons which is a basic functionality of the VC system, however, the use of long term certificates and key-pairs would undermine user privacy. This could happen when sender vehicles are subjected to continuous tracking with the help of safety beacons. Pseudonymous authentication can provide both security and privacy for safety beacons [4, 5, 14], however, this pseudonymity/anonymity is conditional in a way that when misbehavior is detected, the relevant Pseudonymous Certificate (PC)s can be revealed through a resolution protocol and then revoked.

Since signature verifications are expensive for resource-constrained OBUs, [5, 9] propose optimizations for decreasing communication and computational overheads, however, they do not change the fact that signatures are still needed to be verified on all the received beacons. These approaches come out to be the straightforward approach of authenticating each of the received beacons and deem relevant. [6] discusses validating beacons through cooperative beacon verification based on shared verification results, but this is not resilient to DoS attacks since a large portion of computational resources are still used to verify the fictitious beacons.

After a successful signature verification, TESLA based authentication can be used as a cheaper alternative for validating the subsequent beacons [15], however, new PCs and the beacons still needs to be validated in dense environments or under DoS attacks in respectable times. With the help of the information included in a previous beacon, the content of the next beacon can be predicted to help in the validation of beacons [16, 17]. However, this approach does not cooperate with packet loss as when a beacon is lost, the next beacon can only be validated via signature verification. If the approach of prediction is integrated with TESLA, the issue with packet loss is addressed, however, under dense environments or DoS attacks, the combined approach would only result in a majority of beacons being validated based on TESLA MACs and the prediction based authentication would barely be used [7].

An adversary can target the feature of periodic change of PCs and flood the network with fictitious beacons and delay the process of verification of new PCs. This problem is addressed by [7] which provides a cooperative approach for efficiently discovering legitimate PCs under a dense environment or a DoS attack and validates beacons with the help of TESLA and shared verification results in a timely manner. However, non-repudiation of accepted messages cannot be preserved which contains the information about the status of the sender nodes after some TESLA checks due to the fact that TESLA-based authentication does not provide non-repudiation [11]. This problem is not addressed in any of the above mentioned schemes. This work addresses exactly this problem by providing an approach that helps in validating authentic beacons efficiently under a high rate of incoming beacons while maintaining low waiting times through TESLA along with periodic signature verifications and cooperative verification so as to maintain non-repudiation of accepted messages which contains the information about the current and near-future status of the sender nodes and keep the computational overheads low.

1.3 Contribution

1.3.1 Purpose

The purpose of this work is to implement an efficient scheme, similar to prior approaches, that maintains low waiting times and low validation times and improves upon the efficiency of validating legitimate beacons which further helps in estimating the status of the vehicle kinematics of the sender vehicle. If these systems can be improved and be more efficient under high-density situations without compromising upon the privacy and security of VC systems, the beneficial impact of this could be multi-fold in society. Not only it would benefit to have a more efficient way of validating legitimate beacons, but it will also help understand the very environment the vehicle is functioning in by maintaining non-repudiation of the accepted messages which consists of the information about the current and the near-future status of the vehicles around, thus creating a safer environment.

1.3.2 Goal

The goal was to study, evaluate, and build upon the existing policies, mechanisms, and protocols that happen to operate for vehicle-to-vehicle communication so as to make such communication systems more efficient and resilient under high-density environments. In order to ensure that bogus information is not exchanged amongst vehicles, the idea was to leverage traditional public-key cryptography and take the help of TESLA and cooperative verification schemes to validate the high rate of incoming beacons in a timely manner. Also, to maintain non-repudiation of accepted messages which contains the information about the status of the sender vehicles, this work introduced periodic signature verifications along with TESLA checks and cooperative verification. All this while keeping in mind that the privacy of user data is not threatened and the overheads experienced by the OBUs during the processing doesn't affect the performance of the whole network.

1.4 Sustainability and Ethics

It is vital to also consider the environmental impact when developing new technical features. By creating a system that is more efficient than before, less computational resources are needed to achieve the same processing of beacons, reducing environmental impact. By introducing a more efficient method, we hope to decrease energy use, which results in less carbon dioxide emissions as well.

Since no sensitive data was handled during the project, nor any trade secrets that needed protection, there were no other ethical concerns during the project. All data used was synthetically generated.

1.5 Limitations

Vehicular Communications is an upcoming field and aims to create a safer and more efficient vehicular network with a lot of effort and research being conducted in the area of vehicular communication systems. This research proposes a very basic scheme to address some problems that stem from prior work in the domain of VC systems. The proposed scheme is an extension of similar work and provides a reasonable alternate approach for it to be considered effective and practical.

The delivery of this project was time-bound and was completed with a full-time work of 6-8 months. The idea was not to deploy and implement all the required services and features pertaining to VC systems, but to rather address some problems experienced in previous approaches and implement and evaluate the proposed scheme. The generation of signatures and beacons along with broadcasting and receiving were all implemented and evaluated on a personal computer and not on real-time On-Board Units. This may or may not have resulted in different waiting/validating times and related plots. Considering future work on this proposed scheme, the work can be continued further by other experts in order to be practiced in real-world scenarios.

1.6 Outline

This report is structured as follows. The next section introduces a background to the area in Chapter 2. This includes a description of the VPKI, it's requirements, assumptions, and the vulnerabilities associated. It also discusses the problem area (Section 1.1) and how it relates to the focus of this work. After the background, Chapter 3 gives an overview of the scheme that is implemented in this work and then presents the method for the creation of PCs and beacons and discusses the different validation methods that are implemented. It also discusses the different scenarios that were observed and evaluated. Chapter 4 presents the results that were obtained for the different observed cases and Chapter 5 discusses the results along with the future work scope in this area.

Chapter 2

Background

With the need to create safer and more efficient driving conditions a number of initiatives have drawn a strong support towards VC [18]. VPKI forms the base for all such VC systems and Section 2.1 describes a VPKI with it's requirements mentioned in Section 2.1.3. This chapter then goes on to describe the assumptions (Section 2.2) and the vulnerabilities (Section 2.3) associated with VPKI and this work. Section 2.4 describes the adversary model taken into consideration and different attacks that are possible. Lastly, Section 2.5 describes some tools that were implemented in this work.

2.1 Vehicular Public-Key Infrastructure (VPKI)

In today's vehicular industry, cooperation between vehicles is an area of high interest with the purpose to improve safety and comfort. The VPKI is considered to be a vital part of VC systems since the security aspects of such systems are built on this infrastructure. The VPKI can be seen as a baseline infrastructure that needs to be developed, deployed, evaluated, and maintained so as to provide credential management and security services within a VC system. The communications within the VC systems are as good as the strength of this infrastructure. A lot many aspects and the environment has to be considered to develop such broad-scale systems since even a simple loophole in the system can jeopardize the security of the entire network. All the requirements, assumptions, threats, and relevant adversary models need to be studied with high focus so that appropriate measures could be put in place in advance.

2.1.1 Overview

Security and privacy are major aspects of almost all the applications that are used in today's world. Ad-hoc networks are gaining importance in recent times but due to their decentralized nature, attacks on such systems are inevitable. Actions are triggered by safety-critical applications based on data received from other entities. This is based on the trustworthiness of the exchanged data. The worldwide commonly followed approach is to take the help of Public-Key Infrastructure (PKI) to achieve this trust where certain authorities issue certificates to network entities for authenticity [19]. The main objectives of a PKI are:

- Issuing and provisioning valid certificates
- Controlling the validity of digital credentials (i.e. private key, public key, and certificates)
- Revoking credentials of compromised entities

Eavesdropping, message injection, and falsification can be prevented by authenticating the sender and maintaining message integrity through the use of security mechanisms. By using asymmetric cryptography a sender signs a particular message with his/her own private key that is not available to other entities and stored securely. The respective public key is appended to each of these signed messages, which forms a part of the digital certificate. The receivers of these messages are able to verify the authenticity and the integrity of the message as well as the authenticity of the sender by verifying the signature.

VPKI is considered to be the central building block of secure and privacy-preserving VC systems. Different entities are provided with multiple short-lived anonymized certificates, known as *pseudonyms* which ensure message integrity and authenticity as well as preserve vehicles and user privacy. A pseudonymous certificate is a short term certificate issued by the VPKI, which the vehicles use to communicate with other nodes anonymously in the network without being tracked, monitored, or identified. These certificates do not include any information on the long term identity of the vehicle, thus making it very difficult to make a link between different messages, signed by the same entity [7]. These certificates are valid only for a short period of time and then discarded. From a users' perspective, since privacy is a very important factor, these certificates are used to provide pseudonymity. The pseudonyms or the short-lived digital certificates

are the prevalent means that are implemented to prevent the potential breach of vehicle privacy [20]. This work takes the help of PC to provide pseudonymity and is included as a part of the beacon that is generated.

VC systems entail high-rate transmissions where vehicles/OBUs transmit at a rate of 10 *Hz* i.e., 10 beacon Cooperative Awareness Messages (CAMs) per second. V2V and V2I communications are all protected with the help of Public Key Cryptography. VPKI provides pseudonyms to registered vehicles that keep on switching from one pseudonym to a non-previously used one towards message unlinkability as pseudonyms are per se inherently unlinkable [7, 21].

The efficiency of the VPKI is crucial with the emerging large-scale multi-domain VC environments. The deployment of a VPKI differs from a traditional PKI with the dimension, i.e., the number of registered users (vehicles) or entities and the multiplicity of certificates per user, being one major difference. The VPKI should be capable of issuing pseudonyms, 5 orders of magnitude more than the largest current PKI [21]. Not only each vehicle interacts with the VPKI once or a few times per day to refill its pseudonym pool but also to get the latest revocation information. According to [14, 22], a clogging DoS attack can drastically bring down the performance of a VPKI where an adversary can target the availability of different entities within the network with multiple fictitious requests. This results in the degradation of security and privacy where harmful entities could disrupt the communications between other vehicles and the VPKI.

2.1.2 Key Concepts

The domain of a secure VC system consists of four different authorities, with each playing a different role. All these authorities are assumed to be benign entities and considered completely trustworthy third-parties. Figure 2.1.1 shows the relations between different hierarchies of authorities within the domain of a VPKI. The role of each authority is described below along with some other key concepts that make a big part of the whole VC system.

Root Certificate Authority (RCA)

RCA is the trust anchor within VPKI and this is where the chain of trust starts. The RCA signs its own certificate and so RCA certificates have no signer-id since these are self-signed certificates. The RCA takes the responsibility of signing and issuing certificates

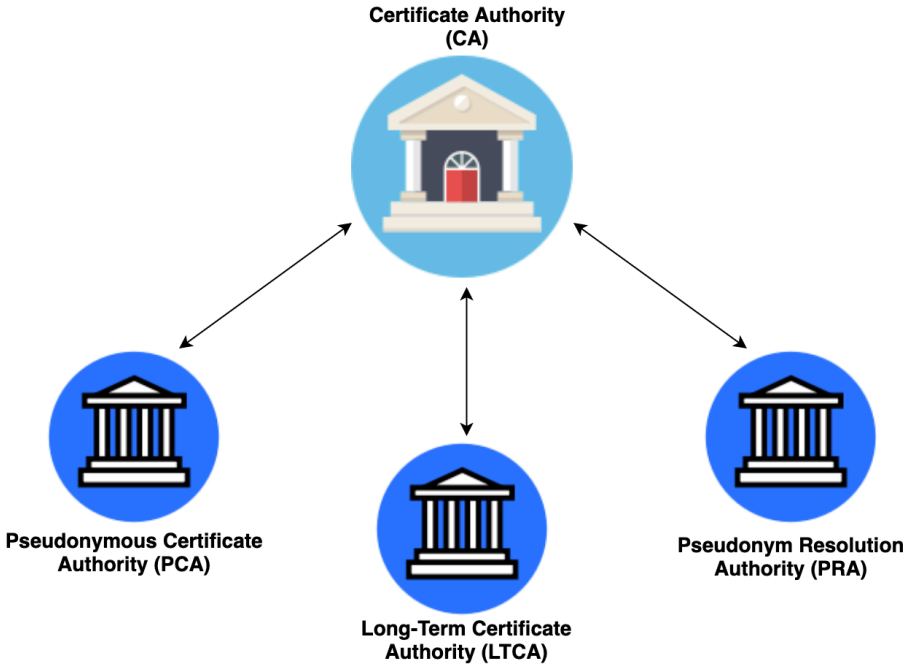


Figure 2.1.1: Authorities Hierarchy in VPKI

to other authorities in VC systems. Being the trust anchor, RCA certificates are available to every vehicle, Intelligent Transport Systems (ITS) stations, and other entities in the network.

Pseudonymous Certificate Authority (PCA)

The role of PCA is to issue pseudonymous certificates for the vehicles. To obtain pseudonymous certificates, the vehicles have to send requests to the appropriate PCA. Once the vehicles obtain the pseudonymous credentials, it is able to communicate with other entities in the VC network anonymously. Since the credentials are pseudonymous, the vehicle can communicate without being tracked, monitored, or identified. The purpose of using pseudonymous credentials is to make it very difficult to make a link between any message or data, signed by the same entity.

Long-Term Certificate Authority (LTCA)

LTCA is responsible for issuing long-term certificates for vehicles, Road-Side Units (RSUs), and other entities within the domain of a secure VC system. Vehicles and other entities obtain long-term certificates by sending requests to the appropriate LTCA. These certificates are used to further obtain pseudonymous credentials. LTCA issues tokens for the vehicles that are used in the process of obtaining pseudonymous certificates.

Pseudonym Resolution Authority (PRA)

PRA is an entity certified by the RCA which plays the role of carrying out pseudonym resolution. The PRA can query the RCA to figure out the related token to a specific pseudonym. The PRA can later ask the LTCA to find the real identity of that token. If a pseudonym is seen as malicious or faulty, the PRA can revoke all such pseudonyms.

Certificate Authorities

As described in earlier, Certificate Authority (CA) are benign entities which are completely trustworthy third-parties within the VPKI. These authorities take the responsibility of providing credential management as well as keep the systems secure by regularly evicting malicious entities and provide security services for authentic nodes. A hierarchy (figure 2.1.1) is followed within the VPKI where the RCA acts as the trust anchor and the rest of the authorities i.e., the PCA, LTCA and the PRA, follow.

Public-Key Certificates

All legitimate entities in the VC network shall posses an authentic certificate. The security operations can be performed only when all the entities follow the protocols. The LTCA provides long term certificates to each vehicle and the PCA provides a finite set of short term certificates within the VC system. Pseudonymous certificates are used to achieve pseudonymity for users for privacy related concerns.

Roaming

In VC systems, vehicles can venture into different areas to send messages or receive messages in order to communicate with foreign entities. The concept of roaming in a VPKI works similarly to a cellular network where users connect to foreign networks outside their allotted area. A VPKI shall support roaming in order for vehicles or entities to accept and exchange messages with foreign entities as long as these concerned entities act according to the network policies. This can be seen as a contract among different realms in order to provide continuous connectivity and services to legitimate entities [23].

Certificate Revocation

Malicious, compromised and bogus entities should be evicted from the network to keep the systems secure. Other legitimate entities in the network should be informed about these revoked entities as well. In order to deactivate or revoke the credentials and certificates of such malicious entities, the paramount approach is to use Certificate

Revocation List (CRL). With the help of CRL, all the other legitimate entities are informed about these revoked nodes. As mentioned in [23], since pseudonymous certificates are used in VC systems, the real identity of the certificate holder would not be determined and so the corresponding trusted authority needs to handle this by obtaining the real identity in a secure way and then deactivate it in order to comply with the privacy policies. A more detailed discussion about certificate revocation is mentioned in [24].

2.1.3 Requirements

With the advancement of VC systems, the drivers, vehicles, and concerned authorities are provided with a rich set of tools for smooth operation. But this feature also makes it possible for a wide range of abuses and attacks on these systems [25]. Consider, for example, a node spreading false information and contaminating large parts of the network, or, a vehicle modifying the content of messages of other vehicles and creating an environment of fake emergencies for everyone else involved. These simple exploits show how important it is for VC systems to be made secure, otherwise, these systems could make criminal and anti-social behavior easier and jeopardize the very development of VC systems. The following describes the requirements of VPKI and is not necessarily relevant to all aspects of network operation and application. These requirements can be seen as building blocks towards more complex specifications and not all of them are expected to be considered, implemented, and deployed in this project.

Message Authentication and Integrity

For protecting messages from alteration and forgery, factors such as message authentication and integrity are a must [23]. The received message should not be altered in order for the receiver to process and perform the required actions. Through these requirements, messages are protected from fabrication, modification and alteration [25, 26].

Entity Authentication

The liveliness of the sender can be ensured through entity authentication and also the fact, that the sender was the one who generated the message. Entity authentication ensures that the sender who sent the message is alive and the message was generated within the approved interval [23, 25].

Message Non-Repudiation

Through this requirement, an entity cannot deny sending, transmitting or receiving a message. Asymmetric-key cryptography is one mechanism that is used to provide the service of non-repudiation where a sender who signs messages with it's private key cannot deny sending the messages as private keys are stored secretly by its owner [23, 25, 26].

Privacy

From the user's point of view, privacy is the most important requirement in a VC system. Privacy protection ensures that private information from vehicles is not extracted or collected during communication [26]. VC system designs shall consider privacy since the very beginning of development and the policies should never reveal private information and protect personal information [25].

Anonymity and Unlinkability

Anonymity is obtained when the identity of an entity is not fully identified with their real identities when the entity is communicating. In a VC system, anonymity is required so that an observer cannot link messages in any way to figure out the real identity of an entity [25]. However, when it comes to forensics investigation or legal obligations, completely trustworthy parties which are legal authorities have the measures to link messages and find the real identity of the entity. Except for these authorities, no other parties shall be capable of finding the real identities of the entities in the network [25]. A more detailed discussion about identity management and privacy can be found in [18]. In order to achieve anonymity and pseudonymity properties, VC systems shall have unlinkability. With unlinkability, an observer can not make a link between two or more messages, and the user can make multiple transactions and use different services without getting identified.

Access Control and Authorization

This policy determines what services different entities are allowed to access or use. It specifies what each entity is allowed to do and what not to do in terms of usage of specific protocols or insertion of specific messages in the network. Authentication, identification, and authorization are services that come under access control [23, 25].

Availability

Even in the presence of faults, the system, services, and protocols shall remain

operational. The systems shall be fault-tolerant and be capable of handling malicious or benign faults as well as remain robust and resilient to resource depletion attacks. The services shall resume back to normal operation after dealing with such faulty nodes [23, 25].

Liability Identification

The current practice in today's transportation systems make the users liable for the accidental or deliberate actions that disrupt the operation of other nodes in the network. In the case of law, legitimate authorized entities can investigate these actions and extract the required information in order to investigate an accident [25].

2.2 Assumptions

Mitigating threats and vulnerabilities is the goal of every secure system and when it comes to security within VC systems, assumptions play a very important role since all the functional and security requirements are based on these assumptions [25]. To mitigate threats, this work also has assumptions at different stages of design and deployment. The following describes the assumptions considered in this work.

Adversaries with limited resources

The assumption is that the adversary operates with limited resources in terms of computational power and memory. The adversary does not posses global knowledge about all the information being transmitted in the network and can only eavesdrop or gather parts of it at a time. All the authentic nodes or entities cannot be enforced to behave in an adversarial manner by the attacker and various recovery procedures can be put in place to keep the system in a secure state.

Every vehicle has CA's Certificate

It is assumed that the CA certificate is available to each vehicle and the vehicle can encrypt messages with the help of the corresponding CA public key attached in the certificate. There should be secure ways to obtain the CA certificate in case the vehicles do not possess them.

No communication jamming

Here, the assumption is that the system is not threatened by the jamming of communication among different entities within the network. If any case of jamming arises then the protocols are designed to mitigate such undesirable communication

jamming.

Resilience to failures

It is assumed that the system works properly and in case of faults or failures, the system recovers fully. The failures are short-lived and the systems come back from an insecure state to a secure state in a short period of time.

Vehicle and User association

The vehicles and the users have an association of *many-to-many*, however, at each point of time, only one user can drive a vehicle [25]. The assumption is that the user is the one that operates the vehicle and could be the owner as well.

Trusted Components (TCs)

The trusted components are assumed to be tamper-resistant and function without being compromised. Their role is two-fold [25], first to protect the vehicle's cryptographic material and their use and the second to safeguard data usable for liability identification.

2.3 Vulnerabilities

A major aspect of VC systems is to improve upon the time horizon of information exchanged within the network in order to increase transportation efficiency and the quality along with higher standards of safety. VC systems today are equipped with a rich set of tools and applications to improve safety and privacy. However, with such a wide range of applications, a large number of different types of abuses are also possible within the domain of VC systems.

Take for example an attacker introducing fake information and subsequently contaminating large parts of a VC network. This bogus information could be about false weather conditions, accidents that didn't happen, traffic jams that don't exist. With so much false information, the users will always be at the risk of taking forced wrong decisions which in turn could turn fatal for them. Legitimate vehicles will be misled with wrong information and the compromised entity can force other entities to follow wrong instructions and this could jeopardize the stakeholders within the VC systems. Another exploit could be when a driver's privacy is compromised by tracking the vehicle's geographical positions and it's transactions and infer some private information from the captured data [3].

Another major exploit could be when a clogging DoS attack is mounted where a huge amount of fictitious messages are sent to a particular entity and the OBUs equipped on such entities would not be able to process all the received messages due to the extraordinary number of beacons received. This will in turn make the OBUs drop even authentic beacons and leave the entity unaware of neighboring vehicle status [7]. Once a vehicle starts to drop authenticate beacons or stops validating even legitimate beacons periodically, it makes inaccurate estimations of the current and near-future status of neighboring vehicles [8] leading to undesirable situations.

2.4 Adversary model

With a variety of tools available for penetrating systems in today's world, many types of attacks and exploits are possible. Different cases of security breaches in different systems have shown over the years, how systems are vulnerable to attacks. Since there exists a possibility of exploits at different stages and levels of every system, an adversary model helps to identify and mitigate these exploits by making it difficult to attack a secure system in terms of time and cost.

Since each system is vulnerable to exploits of different kinds, a VC system is no exception in this case. The aim is to keep the operations of a VC system in a secure state and mitigate all possible attacks so that an imposter cannot stage an attack with the help of any weakness or vulnerability in the system [25]. The following discusses some ways in which different adversaries can stage possibly huge attacks with severe consequences.

2.4.1 Denial-of-Service (DoS) attack

Adversaries could stage simple yet very effective clogging DoS attacks where the receiver is overwhelmed with a high rate of incoming beacons and maintain to be effective in terms of validating authentic beacons. In such attacks, an attacker could generate large amounts of fictitious beacons and prevent the timely reception and validation of legitimate beacons [27]. In case of multiple such adversaries attacking the network with fictitious beacons, then the attack could be classified as a distributed attack i.e a Distributed Denial of Service (DDoS) attack. Hence, when a node receives a pool of valid and fictitious beacons, it becomes very challenging to discover valid messages. These types of attacks are cheap since the attacker needs to just generate

random streams of bytes as signatures and attach them to beacons to make them look like valid beacons. Not only this affects all the neighbors, but the beacon queues become saturated due to the high fictitious beacon rate, which in turn results in significant waiting times for authenticated beacons to be validated. This also utilizes a majority of computational resources since a large number of fictitious signatures need to be verified.

2.4.2 Message Tampering

Through modifications, messages that are in-transit can be modified to make absolutely no sense. However, the integrity of a message is maintained with the help of a mechanism known as MAC. A MAC can be altered only if the corresponding key is known and since the MAC is signed by the sender, an eavesdropper cannot modify, alter or re-transmit a fake message. This way the receiver will detect if a message has been altered and will alert the sender about the contamination.

2.4.3 Forgery

In the case of cryptographic keys and credentials being compromised, masquerading and forgery might happen and this would result in unauthorized entities impersonating an identity and contaminating the network with bogus information. However, with the use of correctly implemented cryptographic primitives, these operations are considered secure and do not threaten the system [25].

2.4.4 Replay Attacks

Another exploit is when an adversary gathers past messages and uses this information to stage an attack by choosing what message can be modified, omitted, forged, or delayed [25]. Each time a pseudonymous certificate or a CRL is requested, a new session is established. As soon as the response is received from the receiver, the session expires and becomes invalid. Assuming that the communication credentials are stored securely and changed periodically, an eavesdropper finds it harder to recollect messages and attack. With the help of timestamps, nonce and cryptographically protected messages, replay attacks, and re-transmission of old messages are mitigated.

2.4.5 Multiple Adversarial Nodes

There are possibilities for adversaries to affect multiple nodes at once and stage a complex combined attack. Consider a scenario where multiple adversarial nodes collude to spread bogus information and try to cause widespread panic among the VC network. However, since each vehicle has to sign a message before transmitting it, in the case of undesirable events caused due to the very bogus information that was spread, the legal authorities can investigate, monitor, and trace back to the attacker to detect the guilty nodes and evict them. Also, since vehicles request for pseudonymous certificates and CRLs using directly the IP-address of the recipient, there is no routing involved among the vehicles. Hence, the threat of dropping of data by adversaries on the way does not exist.

2.4.6 Sybil Attacks

Sybil attacks are severe attacks where an intruder tries to maliciously claim or steal multiple identities to interrupt the functioning of vehicular networks by disseminating false identities. Through this attack, the attackers can abuse pseudo-identities and manipulate fake identities to compromise the effectiveness of the system [28]. Sybil attacks can initiate different types of attacks such as DoS attacks, or fabricating wrong traffic information or injecting false information in the network and disrupting various services. Various defense mechanisms are discussed in [28] and [29] for further details.

2.5 Tools

On-Board Unit (OBU) and Road-Side Unit (RSU)

In order for a vehicle to connect to or be a part of a network of VPKI, the vehicle needs to be equipped with a small yet very important electronic device, called the OBU. Similarly, Road-Side Unit (RSU) are devices located on the roadside that provides V2I connectivity to nearby vehicles and thus communication within a vehicular network. The purpose of both these units is to provide communication among different vehicles and provide important information about transportation such as location, speeds, traffic conditions, hazards, weather, etc [30]. Figure 2.5.1 gives a clearer picture of the same.

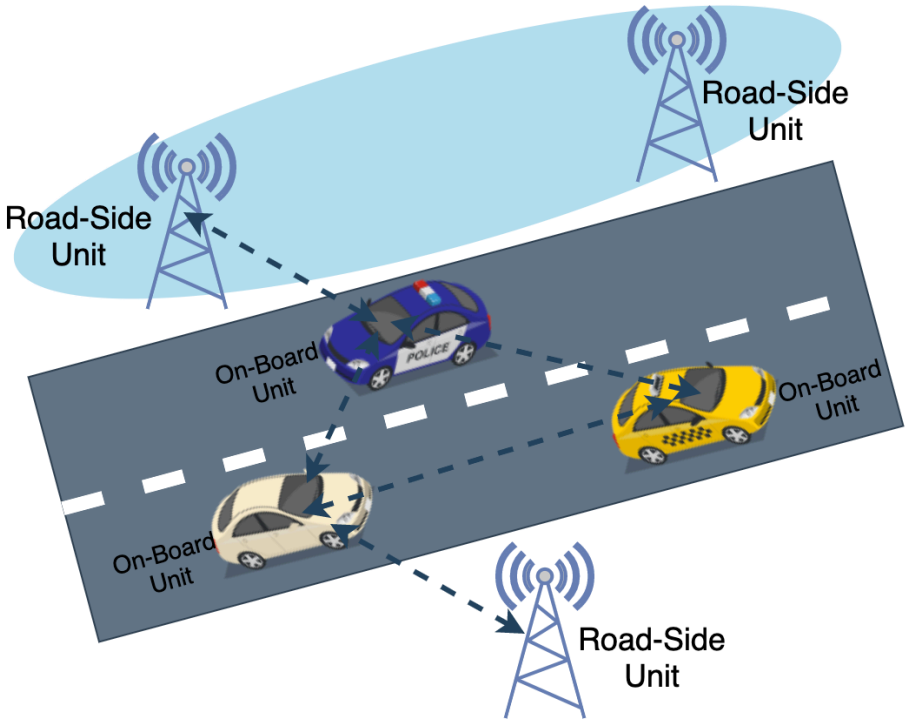


Figure 2.5.1: OBU and RSU. Arrows show communication through wireless mediums.

OpenSSL

OpenSSL is a toolkit for the Transport Layer Security (TLS) and Secure Sockets Layer (SSL) protocols which is robust, commercial-grade and provides a general-purpose cryptography library [31]. This work makes use of OpenSSL library functions for key-pair generation, signature generation, and signature verification.

Elliptic Curve Digital Signature Algorithm (ECDSA)

The ECDSA is a variant of the Digital Signal Algorithm (DSA) that uses the elliptic curve cryptography. Digital signatures play a vital role in providing integrity, authentication, and non-repudiation in communication networks. The digital signature gives the user a reason to believe that the message was sent by an authentic sender and the content is not altered. This work uses the ECDSA within the OpenSSL for key generation, signature generation, and signature verification. When compared to Rivest–Shamir–Adleman (RSA), the key used for encryption by the elliptical curve is shorter than the one used in RSA. This makes the ECDSA a much more attractive option for applications that function on very limited resources in terms of memory, power, or bandwidth such as the OBU. More about the implementation of ECDSA is discussed in [32].

Chapter 3

Method

This chapter starts by giving an overview of the scheme that is implemented in this work in Section 3.1. The method employed to validate beacons in this work consists of a few steps. The first step was the generation of a Pseudonymous Certificate described in Section 3.3. The next step was the creation of TESLA key chains (Section 3.5) that are to be included in the beacons. After the creation of the PC and the TESLA key chain, the beacons are generated with all the fields and its format is mentioned in Section 3.6. Once the beacons are generated which consists of the PC and the TESLA keys, this chapter goes on to describe the different types of validations that are implemented in this work in Section 3.7. Lastly, after describing the types of validation, Section 3.8 describes in detail, the four observed scenarios that were evaluated in this work.

3.1 Overview of the scheme

This scheme extends the traditional V2V message verification by taking the help of TESLA and shared verification results via cooperative verification to address the issue of a high rate of incoming beacons and maintain low computational overheads along with maintaining non-repudiation of accepted messages which contains the information about the current and near-future status of the sender nodes.

The basic idea is to introduce period signature verification along with TESLA and supplement the process with shared verification results via cooperative verification to validate the high rate of the incoming beacon. The TESLA keys and MACs can expedite message verification and remain resilient to extreme network conditions,

however, since TESLA does not provide non-repudiation, there exists the possibility of false beacon injection by legitimate, yet malicious, vehicles and non-repudiation of the accepted messages is not preserved after a few TESLA checks. Hence, with the introduction of a period signature verification, non-repudiation is maintained. On top of this, each beacon includes brief identifiers of previously verified beacons by other vehicles that indicate that the corresponding beacon has been verified by the sender. The identifiers provide significant help and point to potentially valid beacons in a set of bogus and legitimate beacons since these were already verified by a neighboring vehicle, meaning they were already verified based on signature. These shared verification results are exactly where the nodes/vehicles benefit from each other and further reduce the computational overheads incurred due to the periodic signature verifications since multiple beacons that were supposed to be verified based on signature have already been verified through the cooperative verification results. In other words, due to cooperative verification, beacons can skip the periodic signature verification to even further reduce computational overheads and still maintain non-repudiation of the accepted messages which consists of the status of the sender nodes.

3.2 Using Cryptography Library

Since this project is based on the usage of the traditional public-key cryptography, this work makes use of OpenSSL library functions for key-pair generation, signature generation, and signature verification. The key-pair consists of a 'Private Key' and a 'Public Key'. The OpenSSL library function used to generate a key pair is EC_KEY_generate_key(). The purpose of having a private and a public key is to sign a particular piece of information or data by one key and verify the same signature with the help of the other key respectively. The OpenSSL library function used to create a signature is ECDSA_sign() which takes as input, along with other inputs, the hash of the data to be signed and the key to be used to sign the data. This generates a Distinguished Encoding Rules (DER) encoded signature. The OpenSSL library function used to verify a signature is ECDSA_verify() which takes as input, along with other inputs, the hash of the data to be verified and the key to be used to verify the data.

3.3 Creation of Pseudonymous Certificate (PC)

In this work, generating a pseudonymous certificate, that is to be attached to a beacon, requires the sender to initially send its public key to the concerned Certificate Authority. The public key is sent to be signed by the CA as proof that the public key used by the sender is an authentic one and can be trusted based on the trust assumed on a third party which in this case is the CA. Along with sending the sender's public key, the sender also specifies the lifetime of the pseudonym that the user wants to have. More specifically, the user specifies the 'Start Time' and the 'End Time' of a particular pseudonym. All the time that is considered and compared in this work is based on the epoch clock i.e the time passed since 1 January 1970. When specifying the Start Time and the End Time, the following two conditions apply.

- Start Time has to be greater than or equal to the Current Time (i.e. Starttime ≥ Currenttime)
- End Time has to be greater than the Start Time (i.e. Endtime > Starttime)

Once the user has entered the correct life span of the pseudonym that is required for further communication and presented its public key, the CA uses its private key to sign this content of data. While doing so, the CA also attaches an ID so that a particular pseudonym can be identified as well. The CA then hashes this content of information and signs it using its private key. The signature generated is then attached to the PC. The format of the PC in this work is shown in figure 3.3.1. The following shows what content is attached to a generated PC.

- PC ID
  The PC ID is a unique identifier for a particular PC. It is a number generated with the help of a random number generator and attached at the very beginning of a newly generated PC.
- User's Public Key
  This field consists of the user's public key which was validated by the concerned CA and can now be trusted by other entities based on the trust assumed on the CA.
- Timestamp
  The next two fields consist of the start time and the end time of a particular PC. The PC is valid for use only in this time interval. The time is in seconds and in

with respect to the epoch time.

- CA Signature
  This field consists of the signature generated by the CA. The CA uses its private key to sign the user's public key and the timestamp to validate the key with the help of an OpenSSL library function that is ECDSA_sign(). The signature generated is DER encoded and is converted to the Base64 format and then attached to the PC.

PC_ID || User's Public Key || PC_Starttime || PC_Endtime || CA_Signature

Figure 3.3.1: Format of a PC

3.4 Timed Efficient Stream Loss-Tolerant Authentication (TESLA)

The receiver side of communication uses TESLA to check the authenticity and integrity of the source of each packet it receives in multicast or broadcast data streams. With the use of TESLA, no prior trust is required between receivers and the operations per packet is low-cost at both the sender and receiver. TESLA provides low computation and communication overheads, and robustness to packet loss, however, it does not support non-repudiation. In certain circumstances, TESLA implementation is also used to protect receivers against denial of service attacks. [11] discusses about TESLA in more detail. This work introduces TESLA checks along with other types of authentication methods, similar to prior approaches, to efficiently discover authentic beacons and maintain low validation and waiting times.

3.5 TESLA key chain generation

This work assumes a beacon frequency of 10 Hz. Every node consists of a pre-generated list of the TESLA key chain [33] and the keys are then attached to the beacons that are to be broadcasted. The length of the TESLA key that can be authenticated is directly proportional to the product of the beacon frequency and the PC lifetime. If the PC lifetime is assumed to be 5 min then this pre-generated list consists of 3000 TESLA keys and can be refreshed and generated again periodically depending on the usage of a

particular node if all the keys were used to be attached to beacons and sent. Each PC is required to maintain an independent TESLA key chain in order to ensure unlinkability among the broadcasted beacons under different PCs.

K_i is the key that is disclosed during the time slot, T_{i+1}, and is attached in the broadcasted beacon. K'_i is the key used to calculate the TESLA MAC for the beacon broadcasted during T_i. H() and H'() are two different hash functions as shown in figure 3.5.1. Two things to consider here is that in the chain, K_i = H(K_{i+1}) and K'_i = H'(K_i). Each sender uses its private key corresponding to the currently valid PC to sign each beacon and the TESLA MAC attached to each beacon is calculated based on the corresponding TESLA key. Each TESLA key authenticates a broadcasted beacon during its corresponding time slot. The time slot considered here is 0.1 sec and each node broadcasts only one beacon within a time slot. For example, K'_i should be used to generate MAC for $Beacon_i$ and K'_{i+2} should be used to generate MAC for $Beacon_{i+2}$ and so on. In case a beacon is skipped or dropped, for example at T_{i+2} then the corresponding key K_{i+1} (along with K'_{i+1}) is also skipped. This way the TESLA MACs can be authenticated with correct TESLA keys based on the beacon reception times [7].

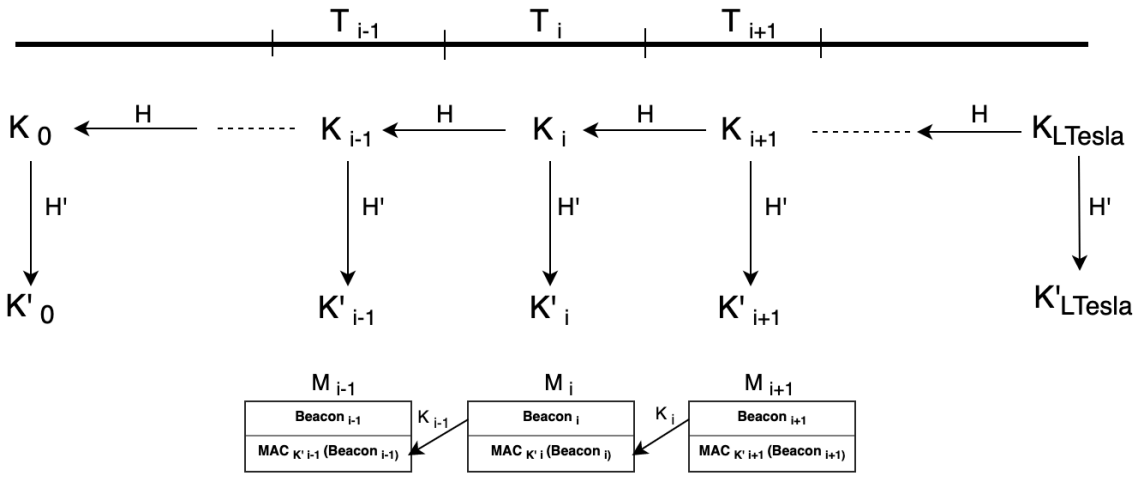


Figure 3.5.1: TESLA Key Chain Generation

3.6 Generation of Beacons

A beacon is the piece of information or message that is broadcasted by the OBUs in the network. The purpose of beacons is to create awareness among the entities about all the other entities in the respective region. These beacons carry information about the

vehicle such as the status of the vehicle which includes information about its speed, direction, route, traffic nearby, etc. All this is useful information for vehicles that receive these beacons since this makes the vehicle have a better sense of awareness about the surrounding it is operating in. In order to be identified as legitimate beacon, signatures, and timestamps are added as well. Hence, the beacon content is signed by the vehicle's private key and its public key is attached along with the signature and the timestamp to make sure the freshness of a particular beacon.

A vehicle also adds the hashes of already verified beacons to the beacons it is about to broadcast so that these hashes can be used as means of cooperative beacon verification for other vehicles that receive such beacons. To make it easier for the receiver to validate authentic beacons, the sender entity attaches TESLA keys and MAC along with all the other information as well. The TESLA keys and MAC are the result of the TESLA chains that is formed beforehand so that the beacons can be linked to each other. This works in a way such that the TESLA key sent in one beacon, acts as the key to be used to verify the MAC attached in the beacon received before this. TESLA key generation and linking of messages are described in Section 3.5.

Along with this, a pseudonymous certificate is also attached to maintain the anonymity and privacy of the user. The PC generated, as described in Section 3.3, has signatures from the CA to make the whole beacon a valid beacon that can be verified at the receiver end. All this information is added to make sure that a particular beacon is identified as a legitimate beacon at the receiver side and is being sent by an authentic vehicle with no intentions to harm the network. Considering the scheme implemented in this work, the following shows what content is attached to a generated beacon.

- Beacon ID

  The beacon ID is a unique identifier for a particular beacon. It is a number generated with the help of a random number generator and attached at the very beginning of a newly generated beacon.

- Status

  This field contains information about the current status of the vehicle and its environment. This information could be the live location, speed, direction, traffic conditions, or any other useful information that makes the nearby vehicles aware of the environment it is functioning in.

- Timestamp

Timestamps play a very important role in ensuring the freshness of a message and so the respective current timestamp is attached. The timestamp is in order of seconds and calculated based on the epoch time.

- TESLA Hash
  This field contains the hash of a pre-generated TESLA key. This hash can be used to validate the MAC received in a previous beacon since the hash is a part of a TESLA key chain. The hash is generated with the help of the 256-bit Secure Hash Algorithm (SHA) and attached in the beacon.

- Cooperative Hash of beacons
  This field consists of the hash of an already validated beacon. The sender sends 'NULL' if no beacon was verified till that point or sends the hash of a newly verified beacon. SHA is used to generate the hash and this cooperative hash can speed up the process of verifying an authentic beacon.

- Vehicle's Signature
  The next field has the signature of the vehicle that sends the particular beacon. The signature is generated with the help of the private key of the vehicle which the vehicle is supposed to keep securely. The signature generated is a DER encoded signature and is converted to the Base64 format so that it can be transmitted over and attached to the beacon in a readable format.

- Pseudonymous Certificate (PC)
  The next thing attached to the beacon is the pseudonymous certificate (PC). The generation of a PC is discussed in section 3.3 which includes the fields PC_ID, vehicle's public key, the lifetime of the PC, CA signature and the hash of the whole PC.

- TESLA MAC
  This is the last field of the beacon and consists of the hash of the whole beacon. This MAC is generated with the help of the TESLA key contained in the subsequent beacon and a predetermined random key. This MAC serves the purpose of verification of the TESLA key chain attached in each beacon and validates the beacons.

One important thing to consider is to make sure that these beacons consist and convey just enough information such that only important information is broadcasted and user-

Beacon_ID || Status || Timestamp || TESLA Key || Cooperative_Hash || Vehicle_Signature || Pseudonymous Certificate || TESLA MAC

Figure 3.6.1: Format of a Beacon

privacy issues don't arise. Broadcasting only just the important information is vital because if more content of information is sent, that means more signature verification has to be carried out or more information has to be processed, meaning an increase in computational overheads at the receiver end. The idea that this work is based on is to reduce these overheads and so the attempt is to send only the needed and important information in these beacons so that these can be validated with the least amount of signature verifications, cooperative verifications or TESLA.

3.7 Validation Types

This work authenticates the beacons with the help of the three validation types mentioned earlier. These are signature verification, TESLA check, and cooperative validation via shared verification results. The following describes the three validation types implemented in this work.

- Signature Verification
  Validation through signature verification is the most basic form of implementing authentication of beacons. In this case, the sender signs the beacons with its private key and the receiver verifies this signature with the help of the public key it receives. This public key is verified by the associated CA. Digital signatures provide authentication, integrity, and non-repudiation. These validations give a very strong reason for the receiver to believe that the beacon sent by the sender is authentic.

- TESLA Key and MAC check
  In this case, since each beacon that is broadcasted is attached with a TESLA key and these keys are a part of the TESLA chain, the subsequent beacons can be verified just by validating the hash chain. The MACs are also signed with the help of these TESLA keys and helps the receiver to validate the beacon just by verifying the MAC. This mechanism provides authenticity and integrity, however, it does

not guarantee non-repudiation [11].

- Cooperative Verification via Shared Verification Results
  Hashes of already verified beacons are attached to newly generated beacons and sent to other peers to indicate that some beacons were already verified via signature verifications and need not be verified again. This is how neighbors can cooperatively validate beacons and help each other in speeding up the verification of different beacons by sharing their own verification results.

3.8 Observed Scenarios

After the generation of PC and beacons and specifying the mechanisms involved to validate the beacons, this work takes into consideration four different scenarios. These scenarios include the above described validations as a combination of the three. The following describes each implemented scenario in detail.

3.8.1 Validation through Signature Verification only

The first scenario implements the validation of beacons only via the use of signature verification. The sender signs each beacon that is to be broadcasted with its private key. At the receiver end, whenever a new beacon is received, the PC of the received beacon is checked. Every node in this network maintains a list of PCs. Every time a new PC is received, then the PC is cached and stored in this list. Once it is cached, the receiver then validates the beacon after conducting signature verification. Signature verification is done via the use of CA verified public key attached in the beacon as described earlier. Once a beacon is verified, it is stored in a separate list of accepted beacons, which is also maintained by each node in the network. In this way, each beacon that is received is validated via signature verification one after the other. Algorithm 1 describes the above as shown in figure 3.8.1.

3.8.2 TESLA key and MAC check

In this case, the first newly received beacon is checked for its PC. The newly received PC is first cached and then the beacon is validated via signature verification. Once the first beacon is validated, the subsequently received beacons are validated with the help of TESLA keys. Since each beacon is attached with a TESLA key and a MAC, and the keys

```
1.  Mi = { Beaconi , MAC K'i(Beaconi) }
2.  if PC is not in PC list then
3.  |  Cache PC
4.  |  Verify Signature and PC
5.  |  if Signature and PC is verified then
6.  |  |  Accept Mi and store in separate list of accepted messages
7.  |  end if
8.  end if
9.  else
10. |  Verify Signature and PC
11. |  if Signature and PC is verified then
12. |  |  Accept Mi and store in separate list of accepted messages
13. |  end if
14. end if
```

Figure 3.8.1: **Algorithm 1**- Signature Verification only

are part of the TESLA chain, the keys, and the MACs can be verified with the help of the beacon that is received next in the order. For example, first Beacon 1 is verified with the help of signature verification, then Beacon 2 is received, then as soon as Beacon 3 is received, the key in Beacon 3 is used to verify the key and the MAC attached in Beacon 2. In other words, the first beacon is validated via signature verification and the rest incoming beacons are validated with the help of TESLA key and MAC. Algorithm 2 describes the above as shown in figure 3.8.2.

```
1.  Mi = { Beaconi , MAC K'i(Beaconi) }
2.  if PC is not in PC list then
3.  |  Cache PC
4.  |  Verify Signature and PC
5.  |  if Signature and PC is verified then
6.  |  |  Accept Mi and store in separate list of accepted messages
7.  |  end if
8.  end if
9.  else
10. |  Verify TESLA Key by checking from the key list
11. |  Verify TESLA MAC by using key attached in beacon
12. |  if Key and MAC is verified then
13. |  |  Accept Mi and store in separate list of accepted messages
14. |  end if
15. end if
```

Figure 3.8.2: **Algorithm 2**- TESLA Key and MAC check after Signature Verification

3.8.3 Periodic Signature Verification

This scenario is a combination of the previous two cases wherein a newly received beacon is validated with the help of signature verification, the next few are validated with the help of the TESLA key and MAC, and then again a beacon is validated through signature verification. This cycle of validating through signature verification happens periodically after every few beacons. For example, Beacon 1 when received is validated through signature verification, Beacon 2 to 4 is validated with the help of TESLA key and MAC and Beacon 5 is validated again through signature verification. This periodic signature verification is every 5 beacons. In other words, every fifth beacon is validated via signature verification whereas the rest of the beacons are validated with the help of TESLA key and MAC. Algorithm 3 describes the above as shown in figure 3.8.3.

```
1.  Mi = { Beaconi , MAC K'i(Beaconi) }
2.  if PC is not in PC list then
3.      Cache PC
4.      Verify Signature and PC
5.      if Signature and PC is verified then
6.          Accept Mi and store in separate list of accepted messages
7.      end if
8.  end if
9.  else
10.     if received Beacon number is a multiple of 5 then
11.         Verify Signature and PC
12.         if Signature and PC is verified then
13.             Accept Mi and store in separate list of accepted messages
14.             Exit check
15.         end if
16.     end if
17.     Verify TESLA Key by checking from the key list
18.     Verify TESLA MAC by using key attached in beacon
19.     if Key and MAC is verified then
20.         Accept Mi and store in separate list of accepted messages
21.     end if
22. end if
```

Figure 3.8.3: **Algorithm 3**- Periodic Signature Verification along with TESLA Key and MAC check

3.8.4 Cooperative Verification

This scenario is a combination of the previous three cases along with the addition of the cooperative hash provided by a neighbor within the network. In this case, the first beacon is validated with the help of signature verification, the next few are validated with the help of the TESLA key and MAC, and the periodic validation through signature

verification is also included. Along with this, neighboring nodes share their verification results of beacons that were validated by them through signature verifications on their end. Hence, whenever a node receives a cooperative hash from a neighbor attached in a beacon, the periodic signature verification check on this node's side is pushed a number ahead since the beacon it just received was already verified based on signature. For example, Beacon 1 when received is validated through signature verification, Beacon 2 to 4 is validated with the help of TESLA key and MAC and Beacon 5 is supposed to be validated again through signature verification. However, Beacon 3 was attached with a cooperative hash which pushes the signature verification check at Beacon 5 to Beacon 8. In other words, the periodic signature verification check is pushed every time a cooperative hash is received (by a factor of 5 in this case). Everything else functions as described earlier.

```
1.  Mi = { Beaconi , MAC K'i(Beaconi) }
2.  if PC is not in PC list then
3.      Cache PC
4.      Verify Signature and PC
5.      if Signature and PC is verified then
6.          Accept Mi and store in separate list of accepted messages
7.      end if
8.  end if
9.  else
10.     If attached cooperative hash is found in stored hash list of beacons then
11.         Accept Mi and store in separate list of accepted messages
12.         Push periodic signature verification by one
13.         Exit check
14.     end if
15.     if received Beacon number is a multiple of 5 then
16.         Verify Signature and PC
17.         if Signature and PC is verified then
18.             Accept Mi and store in separate list of accepted messages
19.             Exit check
20.         end if
21.     end if
22.     Verify TESLA Key by checking from the key list
23.     Verify TESLA MAC by using key attached in beacon
24.     if Key and MAC is verified then
25.         Accept Mi and store in separate list of accepted messages
26.     end if
27. end if
```

Figure 3.8.4: **Algorithm 4**- Checking for cooperative hash along with periodic Signature Verification and TESLA Key and MAC check

Chapter 4

Experimental Results

The results presented in this chapter shows the results that were obtained from the four scenarios that are described in section 3.8. Section 4.1 provides the details about the Test Bed for this work. Results from implementing the baseline approach is presented in section 4.2.1 and introducing TESLA is presented in section 4.2.2. To overcome the drawbacks from the first two approaches, the results obtained from introducing periodic signature verification and cooperative verification are presented in section 4.2.3 and section 4.2.4 respectively.

4.1 Test Bed

All experiments were performed on a single test machine, which was equipped with a 2.3 GHz Dual-Core processor, and 8 GB of RAM. The machine was running macOS (version 10.15.16 Catalina) with dedicated hardware. To create a setup to emulate real-world scenarios, assuming each node was running one script, multiple scripts were run simultaneously to generate, broadcast, and receive beacons from each other.

4.2 Performance Evaluation

Simulations were carried out on a 2.3 GHz Dual-Core processor machine as mentioned in Section 4.1. A single program file included all the functions to generate signatures, verify signatures, generate beacons as well as broadcast and receive beacons. Assuming each vehicle to run one program file, multiple files were run simultaneously so as to depict a network with multiple vehicles, each sending and receiving beacons at the same

time. This work assumes a beacon frequency of 10 Hz, a value used in several related works [6, 7] and observes the timings for a load of 5 to 20 vehicles.

There were four different scenarios observed and evaluated, the details of which are described in section 3.8. For each simulation setting, a random of 5 seeded experiments were carried out and the results are averaged over these 5 runs. The following results and figures show the performance of each of the observed cases in terms of waiting times and the validation type.

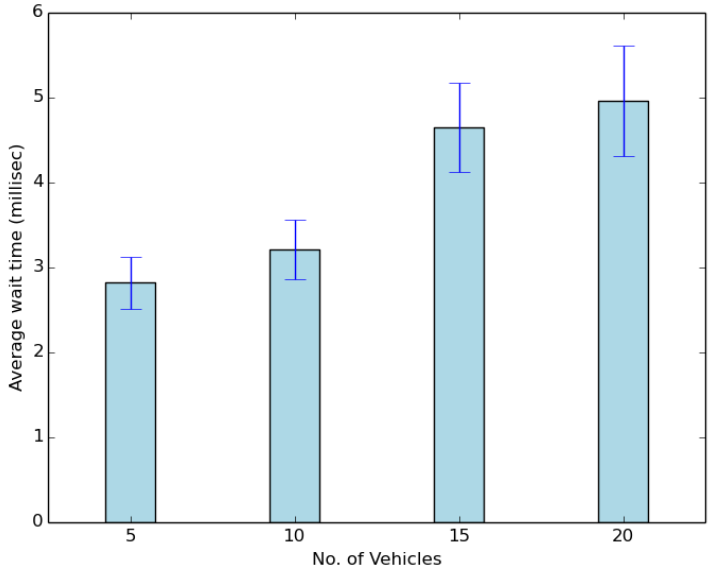
4.2.1 Validation through Signature Verification only

The first observed case was when every incoming beacon was validated with the help of signature verification only as described in section 3.8.1. Figure 4.2.1 shows the waiting times and the validation type count for a different number of vehicles along with the confidence interval for each plot.

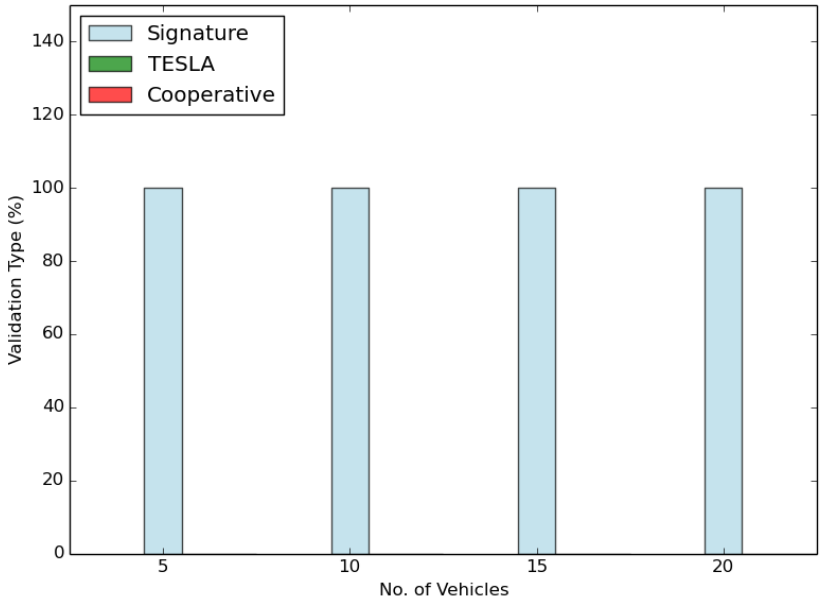
This is the straightforward approach in most of the related works where every incoming beacon is validated via signature verification. The average waiting times are shown in figure 4.2.1 (a) with a 95% confidence interval. Figure 4.2.1 (b) shows the percentage of the validation type and since all of them were verified based on signature, it means 100% of the received beacons are verified based on signature in this scenario. All the other approaches are based on this simple scheme and so is this work where the implemented scheme tackles the problem of a high incoming beacon rate by introducing periodic signature verifications along with cooperative verifications instead of all the beacons being validated through signature verification only. This is to follow the idea of reducing signature verifications since these are expensive and adds to computational overheads but at the same time, not altogether discarding this approach since one has to also consider the authenticity of messages and maintain non-repudiation of the accepted messages which consists of the status of vehicle kinematics.

4.2.2 Validation through TESLA key and MAC

In this case, whenever a new PC was received, the beacon was verified based on signature and then the rest of the incoming beacons with the same PC was verified with the help of TESLA key and MAC as described in section 3.8.2. Figure 4.2.2 shows the waiting times and the validation type in percentage for the different number of vehicles along with the confidence interval for each plot.

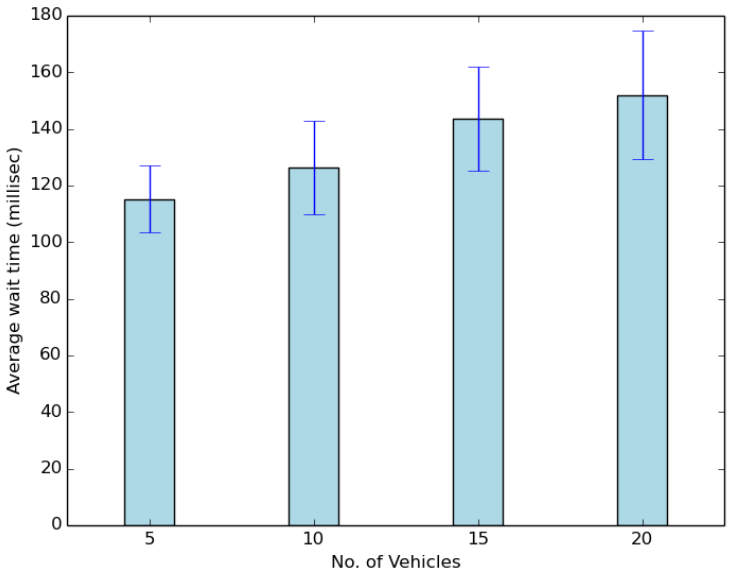


(a)

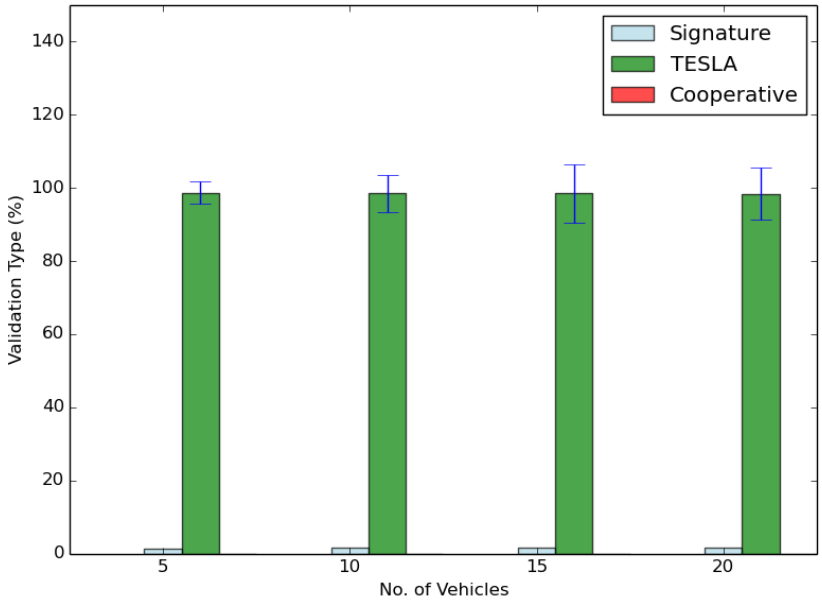


(b)

Figure 4.2.1: Validation through Signature Verification only. (a) Observed waiting times in millisecond for different number of vehicles. (b) Validation Type in percentage for different number of vehicles.



(a)



(b)

Figure 4.2.2: Validation through TESLA key and MAC. (a) Observed waiting times in millisecond for different number of vehicles. (b) Validation Type in percentage for different number of vehicles.

One can observe the average waiting times for each beacon to be verified from figure 4.2.2 (a). Every beacon after the first beacon is verified based on TESLA MACs and for every beacon that has to be validated thereafter has to wait for the next beacon to be received since the corresponding beacon can be validated based on the key received in the subsequent beacon. In other words, a beacon has to wait one beacon interval for it to be verified via TESLA MAC. Figure 4.2.2 (b) shows that almost all the beacons were validated based on TESLA MACs and a very small percentage was verified based on signature. For example, in case 4 neighboring vehicles are around, only 4 beacons were verified based on signatures, while the rest were verified based on TESLA MACs.

This scenario results in dealing with the high rate of incoming beacons since almost all the beacons are verified based on TESLA MACs which is a more efficient method to validate the beacons. TESLA manages to handle saturated networks in an efficient manner, however, when this approach is implemented, non-repudiation of the accepted messages cannot be preserved. Since TESLA does not provide non-repudiation, an adversary can inject bogus information in beacons and affect the accuracy of estimation at the receiver without being held accountable for the false beacons. The next approach addresses this issue.

4.2.3 Validation through Periodic Signature Verification

In this case, a beacon is verified based on signature periodically after every few beacons were checked via TESLA. The observed cases were for a period of 3, 5, and 7 respectively. Section 3.8.3 describes this scenario in detail. Figure 4.2.3 shows the waiting times in milliseconds for a signature verification period of 3, 5, and 7 respectively for the different number of vehicles along with the confidence interval for each plot. Figure 4.2.4 shows the validation type in percentage for the different number of vehicles along with the confidence interval for each plot for the different periods considered.

Since there were more number of signature verifications introduced in this scenario due to a periodic nature, it was not only the first beacon that was verified based on signature. The signatures were verified based on the set period of 3, 5, or 7 respectively. Figure 4.2.4 shows the count percentage of signature verifications went up in each case due to the periodicity introduced in this scenario. For example, take the case of 10 vehicles in figure 4.2.4 (a) with a period of 3, where almost 25% of the beacons were verified based

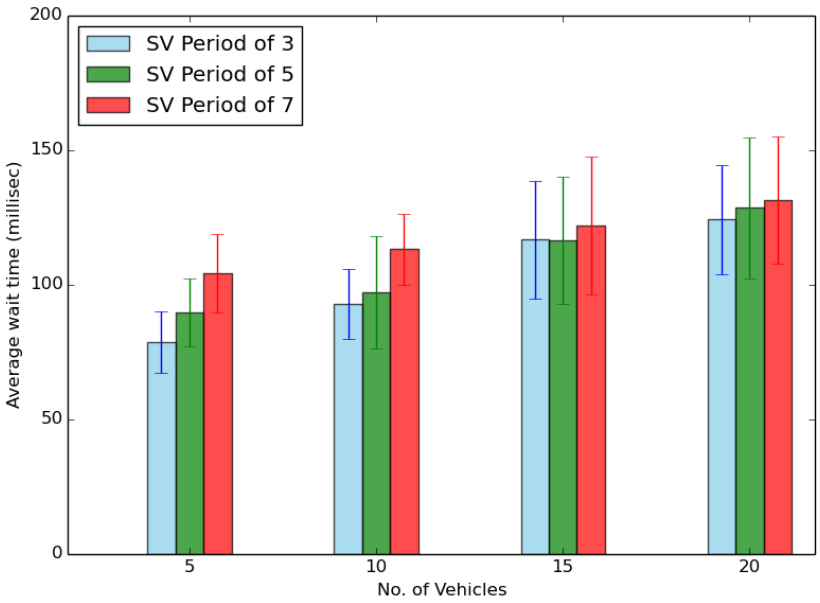
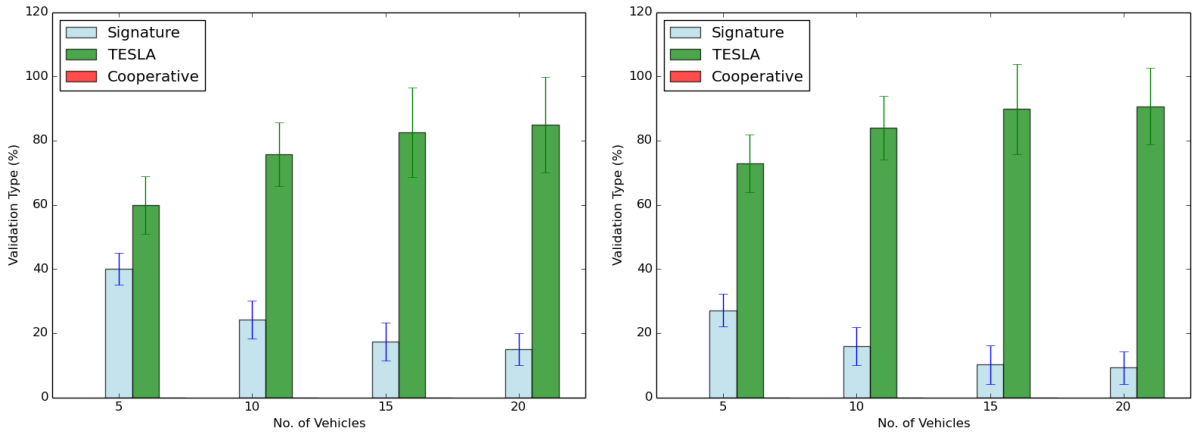
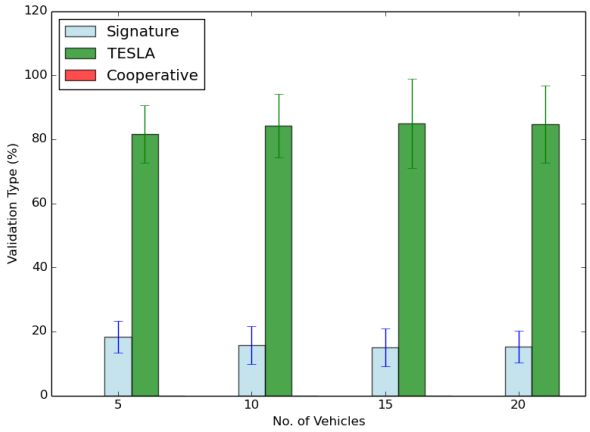


Figure 4.2.3: Validation through Periodic Signature Verification. Waiting times in milliseconds for a signature verification period of 3, 5 and 7 respectively for different number of vehicles.



(a) (b)



(c)

Figure 4.2.4: Validation Type in percentage. (a) Period of 3 (b) Period of 5 (c) Period of 7

on signature, while the rest were validated through TESLA MACs.

The result of this scenario shows that when periodic signature verification is introduced, not only the high rate of incoming beacons is handled due to the validations through TESLA MACs, the periodic signature verifications ensure that non-repudiation of the accepted messages, which contains information about the current and near-future status of sender vehicles, is also preserved. Since beacons are verified based on signatures periodically, the authenticity of the information contained in the beacon can be relied upon. However, when this approach is implemented, the processing overheads experienced by OBUs increases due to the introduction of more number of signature verifications. The next approach addresses this issue.

4.2.4 Validation through Cooperative Verification

This case is a combination of all the previous three cases and included cooperative verification as well. Thus, along with periodic signature verification and TESLA checks of beacons, there were also beacons that were validated via cooperative verifications. The observed cases were for a period of 3, 5, and 7 respectively along with cooperative verification. Section 3.8.4 describes this scenario in detail. Figure 4.2.5 shows the waiting times in milliseconds for a signature verification period of 3, 5, and 7 respectively along with cooperative verifications for the different number of vehicles along with the confidence interval for each plot. Figure 4.2.6 shows the validation type in percentage for the different number of vehicles along with the confidence interval for each plot for the different periods considered.

In this case, the number of signature verifications is brought down due to the introduction of cooperative verification. When cooperative verification is introduced, some periodic signature verifications are skipped since the corresponding hashes in the beacons point out already verified beacons by the sender vehicles. This results in beacons being skipped for signature verification as the vehicles take the help of the cooperative verification. Figure 4.2.6 shows the count percentage of cooperative verifications overtakes the count percentage of signature verification due to the introduction of cooperative verification. For example, take the case of 5 vehicles in figure 4.2.6 (b) with a signature verification period of 5, where 20% of the beacons were validated with the help of cooperative verification and pushes the number of signature verification down to 5% while the rest of the beacons are validated based on TESLA

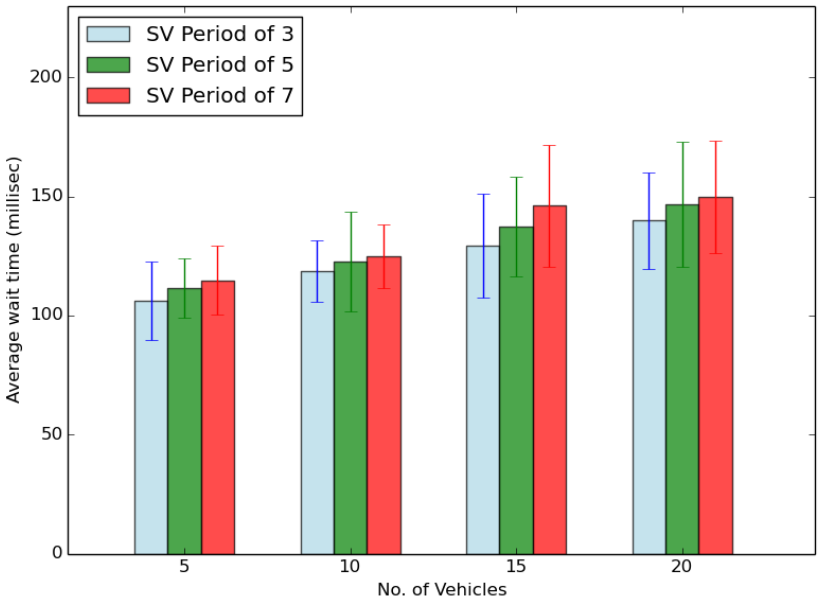


Figure 4.2.5: Validation through Cooperative Verification. Waiting times in milliseconds for a signature verification period of 3, 5 and 7 respectively for different number of vehicles.

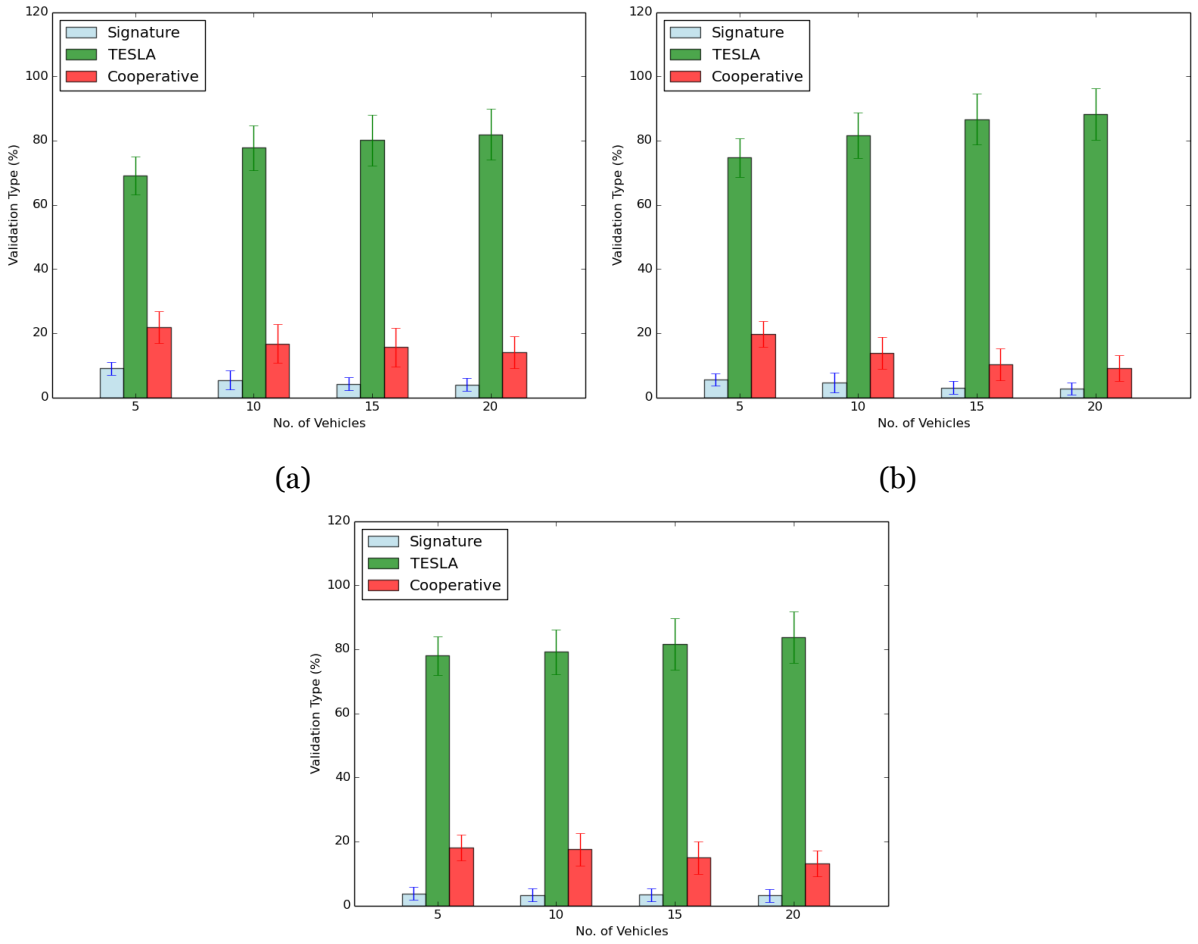


Figure 4.2.6: Validation Type in percentage. (a) Period of 3 (b) Period of 5 (c) Period of 7

MACs.

This scenario introduces cooperative verification along with periodic signature verification and TESLA, which not only handles the high rate of incoming beacons and maintains non-repudiation of the accepted messages which consists of the status of sender vehicles but also reduces the processing overheads that were introduced due to the rise in signature verifications. In this way, when the periodic signature verification is implemented along with TESLA and cooperative verification, a good balance is maintained when it comes to handling a high rate of the incoming beacon, preserving non-repudiation of the accepted messages which consists of the status of sender vehicles and the processing overheads.

The waiting times as seen in figure 4.2.2, figure 4.2.3 and figure 4.2.5 show higher average times when compared to the waiting times of the baseline approach shown in figure 4.2.1. This is because of the introduction of TESLA in the rest of the three approaches. For a beacon to be validated via TESLA, the receiver has to wait for the next beacon to be received since, it is in this subsequent beacon, that contains the TESLA key to verify the MAC received from the previous beacon. This means, the verification process takes at least 100ms for a beacon to be validated since the broadcasting rate is 10Hz, which is the reason why the difference in waiting times can be seen for the scenarios.

Chapter 5

Discussion

The discussion around the results presented in this report is structured as follows. Section 5.1 analyses the results that were achieved by implementing the proposed scheme and how the scheme addressed the problems mentioned in earlier chapters. Section 5.2 discusses the future work that is suggested for further improvements and lastly, section 5.3 concludes all the results.

5.1 Analyzing the Scenarios

The straightforward approach of receiving every beacon and validating every beacon by verifying their signatures is the baseline approach that most related works apply and build upon. Signature verifications are important because these verifications keep a check on the authenticity of the content of the received beacon as well as ensures that the beacon was sent by a legitimate entity. Section 4.2.1 presented the results of this approach which showed that 100% of the received beacons were verified based on signature. However, the drawback of this approach is that since signature verification is an expensive process, it introduces processing overheads on the resource-constrained OBUs. The situation of processing beacons worsens under high-density environments or under DoS attacks where the rate of incoming beacons is very high and the beacons need to be validated in quick times otherwise, even legitimate beacons would be dropped if the system gets saturated.

This is where the introduction of TESLA came into help like mentioned in many related works. TESLA helps in dealing with the problem of a high rate of incoming

beacons when the system is under a DoS attack by validating the beacons in a fast and efficient manner. The TESLA key chains were generated beforehand and attached to the beacons. In this approach, each beacon was validated after receiving the corresponding key from the next received beacon. Section 4.2.2 presented the results of this approach which showed that only the beacon with a newly received PC was verified based on signature while the rest were verified based on TESLA MACs to make the process of validating beacons more efficient and faster even under high-density environments. However, with most beacons being validated through TESLA MACs, non-repudiation of accepted beacons cannot be preserved. Since there were no signature verifications after the initial beacon was verified based on signature, an adversary could have injected bogus information and resulted in the possibility of false beacon injection by legitimate, yet malicious, vehicles.

With the motivation obtained from a related work [8], this work introduced periodic signature verification. This approach implemented a signature verification of every third, fifth, or seventh beacon periodically to deal with the problem of lack of non-repudiation in TESLA. This way, more beacons were validated based on signature verification periodically and this ensured that the information contained in the beacon was legitimate and not injected by an adversary. Section 4.2.3 presented the results of this approach which showed more number of signature verifications when compared to the previous approach. This approach not only addressed the problem of a high rate of incoming beacons by using TESLA, but also maintained non-repudiation of the accepted beacons through periodic signature verifications. However, with the introduction of more number of signature verifications, the processing overheads were introduced again for the resource-constrained OBUs.

By taking the help of the principles involved in [6, 7], the motivation to introduce cooperative verification addressed the problem of increased computational overheads. This approach combined the implementation of a periodic signature verification with cooperative verification along with TESLA. With the introduction of cooperative verification, the signature verifications can be skipped since the corresponding cooperative hash attached in the beacons pointed out to the beacons which were already verified based on signature by the sender vehicle. This meant that these corresponding beacons were not needed to be verified based on signature during the periodic checks. Section 4.2.4 presented the results of this approach which showed that there were more number of cooperative verifications than signature verifications showing that some

signature verifications were skipped since the corresponding beacons were validated based on cooperative verification. This way, this approach maintained a good balance of addressing the issues of a high rate of the incoming beacon, the authenticity of the information in a beacon to estimate the status of the sender vehicle, and the keeping the computational overheads low in an efficient manner.

5.2 Future Work

This is, comparatively, a small study. If time and resources were infinite, the work would have been improved in many directions. The following are some areas for future research.

- Introducing packet loss
  In a more realistic scenario, packet losses are very common and it would be very interesting to observe the performance of the implemented scheme under the presence of packet loss since these would introduce more delays and more issues that need to be addressed in other ways.
- Attaching multiple cooperative hashes
  This work only includes only one hash in every broadcasted beacon that is meant as cooperative verification. Future work with attaching multiple cooperative hashes could result in further reducing the overheads involved and will be an interesting area to work on.
- Stressing under high density
  Another interesting area would be to observe the performance of this scheme when stressed under even a higher rate of incoming beacon like a clogging DoS attack.

5.3 Conclusion

This work aimed to implement an efficient scheme that deals with the problem of a high rate of incoming beacons, preserving non-repudiation of the accepted messages which contains information about the current and near-future status of the sender vehicle, and keeping the computational overheads involved during the processing of beacons low.

The introduction of TESLA key and MAC addressed the problem with the performance of the system under a high-density environment. TESLA validates the high rate of incoming beacons in a fast and efficient way which kept the waiting/validation timings low. The introduction of periodic signature verification addressed the problem of the lack of non-repudiation in symmetric-key-based authentication, while keeping the verification efficient. Combining the above with the introduction of cooperative verification, reduced the number of signature verifications and in turn reduced the computational overheads that were experienced by the resource-constrained OBUs. The implemented scheme with the combination of periodic signature verification, TESLA and cooperative verification addressed all the problems that were aimed in an efficient manner.

References

[1] Panos Papadimitratos, Arnaud La Fortelle, Knut Evenssen, Roberto Brignolo, and Stefano Cosenza. “Vehicular communication systems: Enabling technologies, applications, and future outlook on intelligent transportation”. In: *IEEE Communications Magazine* 47.11 (2009), pp. 84–95.

[2] Georgios Karagiannis, Onur Altintas, Eylem Ekici, Geert Heijenk, Boangoat Jarupan, Kenneth Lin, and Timothy Weil. “Vehicular networking: A survey and tutorial on requirements, architectures, challenges, standards and solutions”. In: *IEEE Communications Surveys and Tutorials* 13.4 (2011), pp. 584–616.

[3] Panos Papadimitratos, Levente Buttyan, Tamás Holczer, Elmer Schoch, Julien Freudiger, Maxim Raya, Zhendong Ma, Frank Kargl, Antonio Kung, and Jean Pierre Hubaux. “Secure vehicular communication systems: Design and architecture”. In: *IEEE Communications Magazine* 46.11 (2008), pp. 100–109.

[4] Mohammad Khodaei, Hongyu Jin, and Panos Papadimitratos. “SECMACE: Scalable and Robust Identity and Credential Management Infrastructure in Vehicular Communication Systems”. In: *IEEE Transactions on Intelligent Transportation Systems* 19.5 (2018), pp. 1430–1444.

[5] Giorgio Calandriello, Panos Papadimitratos, Jean Pierre Hubaux, and Antonio Lioy. “On the performance of secure vehicular communication systems”. In: *IEEE Transactions on Dependable and Secure Computing* 8.6 (2011), pp. 898–912.

[6] Hongyu Jin and Panos Papadimitratos. “Scaling VANET security through cooperative message verification”. In: *IEEE Vehicular Networking Conference, VNC* 2016-Janua (2016), pp. 275–278.

[7] Hongyu Jin and Panos Papadimitratos. “DoS-resilient cooperative beacon verification for vehicular communication systems”. In: *Ad Hoc Networks* 90 (2019), p. 101775.

[8] Hoa Hung Nguyen and Han You Jeong. “Mobility-Adaptive Beacon Broadcast for Vehicular Cooperative Safety-Critical Applications”. In: *IEEE Transactions on Intelligent Transportation Systems* 19.6 (2018), pp. 1996–2010.

[9] Giorgio Calandriello, Panos Papadimitratos, Jean-Pierre Hubaux, and Antonio Lioy. “Efficient and Robust Pseudonymous Authentication in VANET”. In: *ACM VANET*. 2007.

[10] Frank Kargl, Panos Papadimitratos, Levente Buttyan, Michael Müter, Elmer Schoch, Njörn Wiedersheim, Ta Vinh Thong, Giorgio Calandriello, Albert Held, Antonio Kung, and Jean Pierre Hubaux. “Secure vehicular communication systems: Implementation, performance, and research challenges”. In: *IEEE Communications Magazine* 46.11 (2008), pp. 110–118.

[11] Adrian Perrig, Dawn Song, Rein Canetti, J Doug Tygar, and Bob Briscoe. *RFC 4082 - Timed Efficient Stream Loss-Tolerant Authentication (TESLA): Multicast Source Authentication Transform Introduction*. 2005.

[12] Rathi Suganya and Vinod K Ravikumar. “GSIS: A Secure and Timed-Efficient Stream Loss-Tolerant Authentication Protocol For Vehicular Communications”. In: *Int. J. Advanced Networking and Applications* (2013), pp. 1837–1841.

[13] Xiaodong Lin, Chenxi Zhang, Xiaoting Sun, Pin Han Ho, and Xuemin Shen. “Performance enhancement for secure vehicular communications”. In: *GLOBECOM - IEEE Global Telecommunications Conference* (2007), pp. 480–485.

[14] Mohammad Khodaei, Hongyu Jin, and Panos Papadimitratos. “Towards Deploying a Scalable & Robust Vehicular Identity and Credential Management Infrastructure”. In: *EEE Vehicular Networking Conference (VNC), Paderborn* (2014).

[15] Ahren Studer, Fan Bai, Bhargav Bellur, and Adrian Perrig. “Flexible, Extensible, and Efficient VANET Authentication”. In: *Journal of Communications and Networks* 11 (2009), pp. 574–588.

[16] Chen Lyu, Dawu Gu, Yunze Zeng, and Prasant Mohapatra. “PBA: Prediction-based Authentication for Vehicle-to-Vehicle Communications”. In: *IEEE Transactions on Dependable and Secure Computing* 13 (2016), pp. 71–83.

[17] Hsu-Chun Hsiao, Ahren Studer, Chen Chen, Adrian Perrig, Fan Bai, Bhargav Bellur, and Aravind Iyer. “Flooding-Resilient Broadcast Authentication for VANETs”. In: *MOBICOM*. 2011.

[18] Panos Papadimitratos, Antonio Kung, Jean Pierre Hubaux, and Frank Kargl. “Privacy and Identity Management for Vehicular Communication Systems : A Position Paper”. In: *Workshop on Standards for Privacy in User-Centric Identity Management, Zurich, Switzerland* (2006).

[19] Norbert Bißmeyer, Hagen Stübing, Elmar Schoch, Stefan Götz, Jan Peter Stotz, and Brigitte Lonc. “A Generic Public Key Infrastructure for Securing Car-to-X Communication”. In: *18th World Congress on Intelligent Transport Systems*. 2011.

[20] Nikolaos Alexiou, Marcello Laganà, Stylianos Gisdakis, Mohammad Khodaei, and Panos Papadimitratos. “VeSPA: Vehicular Security and Privacy-preserving Architecture”. In: *ACM HotWiSec*. 2020.

[21] Hamid Noroozi, Mohammad Khodaei, and Panos Papadimitratos. “VPKIaaS: A Highly Available and Dynamically Scalable Vehicular Public Key Infrastructure”. In: *ACM Conference on Security & Privacy in Wireless and Mobile Networks*. 2018.

[22] Mohammad Khodaei and Panos Papadimitratos. “Evaluating On-demand Pseudonym Acquisition Policies in Vehicular Communication Systems”. In: *IoV - VoI*. 2016.

[23] Panos Papadimitratos. “”On the Road”-Reflections on the Security of Vehicular Communication Systems”. In: *IEEE International Conference on Vehicular Electronics and Safety* (2008).

[24] Jean Pierre Hubaux, Panos Papadimitratos, and Ghita Mezzour. “Certificate Revocation List Distribution in Vehicular Communication Systems”. In: *ACM International Workshop on VehiculAr Inter-NETworking (VANET)*. 2008.

[25] Panos Papadimitratos, Virgil Gligor, and Jean Pierre Hubaux. “Securing Vehicular Communications - Assumptions, Requirements, and Principles”. In: *Workshop on Embedded Security in Cars (ESCAR)* (2006).

[26] Tim Leinmuller, Levente Buttyan, Jean-Pierre Hubaux, Frank Kargl, Rainer Kroh, Panos Papadimitratos, Maxim Raya, and Elmar Schoch. “Secure Vehicle Communication (SeVeCom)”. In: *Demonstration at 7th Annual International Conference on Mobile Systems, Applications and Services (ACM MobiSys 2009)* June 2014 (2009).

[27] Qijun Gu and Peng Liu. *Denial of Service Attacks*. 2012, pp. 454–468.

[28] Kuan Zhang, Xiaohui Liang, Rongxing Lu, and Xuemin Shen. “Sybil attacks and their defenses in the internet of things”. In: *IEEE Internet of Things Journal* 1.5 (2014), pp. 372–383.

[29] Soyoung Park, Baber Aslam, Damla Turgut, and Cliff C. Zou. “Defense against sybil attack in vehicular ad hoc network based on roadside unit support”. In: *Proceedings - IEEE Military Communications Conference MILCOM* June 2014 (2009).

[30] Mukesh Saini, Abdulhameed Alelaiwi, and Abdulmotaleb El Saddik. “How close are we to realizing a pragmatic VANET solution? A meta-survey”. In: *ACM Computing Surveys* 48.2 (2015).

[31] *OpenSSL*. URL: `https://www.openssl.org/`.

[32] Abdessalem Abidi, Belgacem Bouallegue, and Fatma Kahri. “Implementation of elliptic curve digital signature algorithm (ECDSA)”. In: *GSCIT 2014 - Global Summit on Computer and Information Technology* June 2014 (2014).

[33] Adrian Perrig, Ran Canetti, J Doug Tygar, and Dawn Song. “Efficient Authentication and Signing of Multicast Streams over Lossy Channels”. In: *Proceeding 2000 IEEE Symposium on Security and Privacy*. 2000.