2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress

Detecting Attacks Smartly in Vehicle Cloud Computing

Wei Zhang*†‡, Siwang Zhou*, Avinash Srinivasan‡, Jie Wu‡, Yaping Lin*†
*College of Computer Science and Electronic Engineering, Hunan University, Hunan, China
†Hunan Provincial Key Laboratory of Dependable Systems and Networks, Hunan, China
‡Department of Computer and Information Sciences, Temple University, Philadelphia, USA

***Abstract*—With the advantages of vehicle networks and cloud computing, Vehicle Cloud Computing (VCC) has emerged and wins a lot of interests. In this paper, we propose an attack detection scheme in the VCC, that can detect the potential attacks launched by malicious vehicles, while preserving benign vehicles' privacy. First of all, we propose a distributed secret key distribution scheme, where multiple authorities are involved to distribute keys independently. Then, we systematically construct a novel protocol that enables the cloud to smartly detect numerous potential attacks originating from malicious vehicles, without jeopardizing the privacy of benign vehicles. To guarantee that the construction is transparent to the delay in vehicle ad hoc networks, we propose an enhanced scheme wherein the attack detection frequency is tunable to any system requirements. Further, we define an untraceability model, and show that our scheme achieves untraceability through rigorous security proof. Finally, we conduct extensive experiments to validate the efficacy and efficiency of our schemes.**

***Index Terms*—Vehicle cloud computing, untraceability, attack detection, tunable detection frequency, secret sharing.**

I. Introduction

Vehicle Ad-hoc Networks (VANETs) have the potential benefits of improving drivers' experience, and reducing the traffic accidents [1]. Meanwhile, cloud computing possesses the merits of strong computational capability, easy access, and flexible resource management. With the advantages of both VANETs and cloud computing, the Vehicle Cloud Computing(VCC) is adopted to enhance the efficiency of VANET services, e.g., Urban Surveillance Service and Vehicular Traffic Management [2]. By collecting data from a large number of vehicles, the cloud can generate a very consistent view of traffic conditions in different geographical areas. Based on these data, the cloud can help provide efficient trajectory planning for first responders and emergency services, and prevent road congestion.

However, if the security and privacy issues are not addressed adequately prior to the large scale deployment of the VCC, the potential vulnerabilities will outweigh its perceived benefits.

There are some research works concerned with security and privacy issues in VANETs [3], [4], [5], [6], [7]. Unfortunately, they are not applicable in the VCC due to the specific problems there, e.g., how to adopt the advantages of the cloud to smartly detect the potential attacks. Existing works [2], [8], [9], [10] focus on introducing new applications and conceptions for the VCC. However, none of them propose a concrete security and privacy solution.

As a matter of fact, the cloud in the VCC would be a double-edged sword. Specifically, by making full use of the capability of the cloud in a positive way, we can enjoy both a pleasant driver experience and strong security. However, when the cloud has malicious intentions, the vehicles would suffer from the threat of privacy leakage. Therefore, how to make full use of the strong capability of the cloud, while preventing it from knowing the privacy of vehicles is a very challenging problem.

In this paper, we aim to propose a scheme that can smartly detect the potential attacks launched by malicious vehicles in the VCC, while preserving benign vehicles' privacy. First of all, we propose a distributed secret key distribution scheme, where multiple authorities are involved to distribute keys independently. Then, we systematically construct a novel protocol that enables the cloud to smartly detect numerous potential attacks originating from malicious vehicles, without jeopardizing the privacy of benign vehicles. Meanwhile, our scheme is constructed to ensure the privacy of even the revoked vehicles. But, for forensic concerns, the trusted authority (TA) can easily track the revoked vehicles. To guarantee that the construction is transparent to the delay in vehicle ad hoc networks, we propose an enhanced scheme wherein the attack detection frequency is tunable to any system requirements. Further, we define an untraceability model, and show that our scheme achieves untraceability through rigorous security proof. Finally, we conduct extensive experiments to validate the efficacy and efficiency of our schemes.

Our main contributions are summarized as follows:

- To the best of our knowledge, the proposed framework is the first attempt aiming to provide a highly scalable and secure framework for the Vehicle Cloud Computing.
- We systematically construct a novel protocol for the VCC to smartly detect numerous potential attacks originating from malicious vehicles, without jeopardizing the privacy of benign vehicles.
- We design a flexible scheme which detects potential attacks with a tunable detection frequency. Such a flexible design facilitates the scheme to tune the detection frequency according to system requirement.
- We give rigorous security analysis and conduct extensive experiments to demonstrate the efficacy and efficiency of our proposed solution.

978-1-5090-2771-2/16 $31.00 © 2016 IEEE
DOI 10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.46

The rest of this paper is organized as follows. Section II presents the related work. Section III formulates the problem. Section IV demonstrates the main secure constructions. Section V presents the security proof of the proposed constructions. Section VI shows a tunable detect frequency construction. Section VII demonstrates the efficiency of our proposed scheme. In Section VIII, we conclude the paper.

II. Related Work

A. Vehicle Cloud Computing

Vehicle cloud computing(VCC), which takes full advantage of cloud computing to serve the drivers in vehicle networks, will have a remarkable impact on both traffic management and road safety [11], [12], [13]. Kumar et al. [8] designed Carcel, a cloud-assisted system for autonomous driving. With the cloud gathering sensor data from both vehicles and roadside infrastructures, Carcel helps autonomous vehicles detect obstacles on the street, and plan paths when unexpected events occur. In [9], Wang et al. defined a three-level architecture for vehicle cloud, and presented novel and real time vehicle cloud services. Gerla [2] proposed two vital applications of the vehicle cloud including the urban sensing, and efficient traffic management. In [10], Yan et al. outlined the security and privacy challenges that are specific to vehicle cloud computing.

B. Security and Privacy in VANETs

It is widely accepted that securing the vehicle ad hoc networks while preserving the privacy of drivers is very challenging [3]. Lu et al. [4] presented a novel and efficient conditional privacy preservation (ECPP) protocol to make the communications among vehicles secure. Studer et al. [5] proposed an efficient key management system called Temporary Anonymous Certified Keys (TACKs). Zhou et al. [6] proposed a lightweight and scalable protocol to detect the sybil attacks. Lin [7] presented a Local Sybil Resistance (LSR) scheme to defend against the zero-day sybil attack in a privacy preserving vehicle peer-to-peer network (VPNET).

C. Secret Sharing-based Schemes

Shamir [14] and Blakley [15], introduced the concept of secret sharing, where any t or more secret shares can be used to reconstruct the secret. Nonetheless, with less than t valid shares, the secret cannot be reconstructed. Based on secret sharing schemes, a lot of excellent works have been proposed. Boneh and Franklin [16] introduced a fully functional identity-based encryption scheme (IBE). The Attribute-Based Encryption (ABE) was first introduced by Sahai and Waters [17]. The ABE associates the private keys and ciphertexts with a set of attributes. A user is able to decrypt a ciphertext only if he has enough attributes. Goyal et al. [18] proposed Key-Policy Attribute-Based Encryption (KP-ABE), and Bethencourt et al. [19] introduced the Ciphertext-Policy Attribute-Based Encryption (CP-ABE).

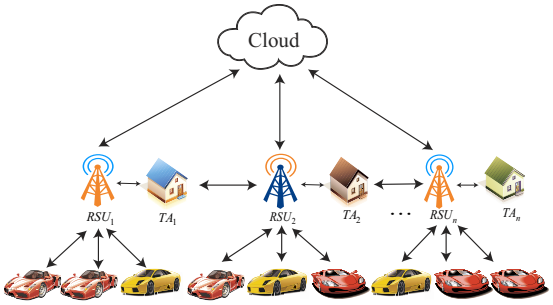


Fig. 1: System model of the vehicle cloud computing.

III. Problem Formulation

A. System Model

As shown in Fig. 1, four entities are involved in the VCC: they are the trusted authorities (TAs), the cloud, the roadside units (RSU), and the vehicles. The TAs are responsible for distributing secret keys to registering vehicles, safeguarding the RSUs, and identifying and revoking malicious vehicles. The RSUs are deployed in different areas. They are responsible for broadcasting messages for a specific area, and act as relays between the vehicles and the cloud. The cloud accepts traffic data from the RSUs, with its high computation and communication capability, the cloud will announce important traffic information, provide efficient route planning, and detect the potential attacks. The vehicles will receive their secret keys from a TA upon registration. The vehicle drivers will submit the traffic data to the cloud, which facilitates the cloud to obtain a global view of the traffic condition, in return, the cloud will help these vehicles make the best decisions.

B. Threat Model

We assume the TAs are trusted, they are responsible for distributing secret keys to registering vehicles, safeguarding the RSUs, and identifying and revoking malicious vehicles. Additionally, we assume the RSUs are trusted, we explain it from two aspects. First, the RSUs are safeguarded by the TAs. Second, due to the high mobility of vehicles, each RSU can only collect a small local data of a vehicle in a very short time. With the limited local data, there is little incentive for them to behave dishonestly. However, the cloud is not trusted. Here we treat it as 'curious but honest' [20]. Specifically, the cloud will follow our protocol to help detect the potential attacks occurring in the VANETs, but it is very curious and trying to deduce sensitive data from the vehicles' submitted data. Vehicles in the network are also not trusted, and they will try to launch attacks to benefit themselves, e.g., to pass an intersection without congestion, the attacker would launch a sybil attack to deceive others as if there are a lot of vehicles in that intersection. Further, we assume the vehicles would not collude with the cloud to either obtain or reveal sensitive information of other vehicles. We will explore the collusion cases in our future work.

C. Design goals

1) The cloud can smartly detect potential attacks, while the TAs can identify and revoke probable attackers.

2) The frequency of attack detection should be tunable, i.e., the detection scheme should be robust to variations of network conditions.

3) The scheme should preserve the privacy of participating vehicles. Specifically, the cloud should not deduce any sensitive data from the event report, or be able to track a specific vehicle.

D. Untraceability Model

In the VCC model, how to enable the cloud to detect attacks without compromising users' privacy (especially for the traceability) is vital and challenging. Before we introduce our formal constructions, we give the following untraceability game.

Setup: The challenger generates the public keys and private keys, and sends the public keys to the adversary $\mathcal{A}$.

Phase 1: $\mathcal{A}$ queries the event report of event m_i for polynomial times. The challenger keeps a list l, which is initially empty, to record the query. If m_i is recorded in l, he selects an unused ID j to generate an event report; otherwise, he randomly selects an ID j to generate the event report. After each generation, the challenger adds a pair $< j, m_i >$ in l.

Challenge: $\mathcal{A}$ submits two events m_0 and m_1 to the challenger. The challenger flips a coin μ to determine whether he uses the same ID's secret key to generate the event report for m_0 and m_1, i.e., if $\mu = 0$, he uses the same ID's secret key to generate an event report ; otherwise, he uses two different IDs. Then the challenger selects the IDs j_0 and j_1(if the challenger uses the same ID, then $j_0 = j_1$), where the pair $< m_0, j_0 >$, and $< m_1, j_1 >$ are not recorded in l, to generate the event reports. After the generation, the challenger adds $< m_0, j_0 >$, and $< m_1, j_1 >$ to l. Finally, the challenger returns two event reports to $\mathcal{A}$.

Phase 2: Phase 1 is repeated.

Guess: $\mathcal{A}$ outputs a guess μ' for μ.

$\mathcal{A}$ is said to win the game if he can correctly identify whether the two event reports are generated with the same ID's secret key. The advantage of $\mathcal{A}$ is defined as $Pr[\mu = \mu'] - 1/2$.

Definition 1. *An event report scheme is said to achieve untraceability if all probabilistic polynomial time adversaries have at most a negligible advantage ϵ to win the above untraceability game.*

E. Bilinear Map

Let G and G_1 denote two cyclic groups with a prime order p. Let g be the generator of G, and e be the bilinear map $e : G \times G \to G_1$. The bilinear map e will have the following three properties:1) Bilinear: $\forall a, b \in \mathbb{Z}_p^*$, $e(g^a, g^b) = e(g, g)^{ab}$. 2) Non-degenerate: $e(g, g) \neq 1$. 3) Computable: bilinear map $e : G \times G \to G_1$ can be efficiently computed.

IV. Secure Constructions

In this section, we elaborate on the secure construction. We first illustrate how to achieve a distributed secret key distribution. Then we show how to generate the event report, how to verify an event report, and how the cloud smartly detects probable attacks without revealing the privacy of vehicles. Finally, we demonstrate how to identify and revoke attackers.

A. Distributed secret key distribution

Assume a vehicle V_j^i under the administration of TA_i submits the registration request, TA_i will distribute the secret keys with the following two steps:

1) TA_i prepares his secret keys (a_i, b_i, c_i, k_{i1}, k_{i2}, s_i), two keyed hash functions (e.g., $HMAC - SHA1$), $H_{k_{i1}}(x)$, $H_{k_{i2}}(x)$, and a secret key distribution function $F(x) = s_i + \sum_{k \in [1, \alpha - 1]} \beta_{i,k} \cdot x^k$, where $\beta_{i,k}$ denotes the coefficient of $F(x)$. TA_i further publishes the public keys (g^{c_i}, g^{s_i}).

2) The vehicle V_j^i with ID j submits a registration request to the authority TA_i, TA_i returns two keyed secret hash functions $H_{k_{i1}}(x)$, $H_{k_{i2}}(x)$, V_j^i's secret ID $sd_j^i = g^{a_i \cdot j + b}$, and V_j^i's secret key $sk_j^i = F(e(g, g)^{a_i \cdot j})$. Note that, to compute the secret key sk_j^i, we use $e(g, g)^{a_i \cdot j}$ to substitute x in the distribution function $F(x)$, instead of using the real ID j or secret ID sd_j^i. After registration, the authority TA_i records the entry (j, sd_j^i, sk_j^i) in his secret key table.

B. Event Report

Given an event m, a vehicle's secret data, including the real ID j, the secret ID sd_j^i, the secret key sk_j^i, two keyed hash functions, $H_{k_{i1}}(x)$, $H_{k_{i2}}(x)$, and the public key of the trusted authority who administrates the vehicle. The vehicle V_j^i will generate the event report $\mathcal{E}_j^i$ for m with the following three steps.

1) The vehicle computes four secret data items, where $vs_{j0} = e(g, sd_j^i \cdot g^{j \cdot H_{k_{i1}}(m)}) = e(g, g)^{a_i \cdot j + b_i + j \cdot H_{k_{i1}}(m)}$, $vs_{j1} = g^{j \cdot H_{k_{i2}}(m) + c_i \cdot H_{k_{i1}}(m)}$, $vs_{j2} = g^{H_{k_{i1}}(m) / H_{k_{i2}}(m)}$, $vs_{j3} = e(g, g)^{sk_j^i \cdot H_{k_{i1}}(m) / H_{k_{i2}}(m)}$. These secret data will be used to help the cloud detect probable attacks.

2) The vehicle embeds some secret data in the event report, so that once the event report is modified by attackers, the cloud can detect it. Meanwhile, these data will help detect some attacks, e.g., fabrication attacks. Specifically, the vehicle first generates three random parameters: r_{j1}, r_{j2}, r_{j3}, then it computes two intermediate parameters: $vt_{j1} = g^{r_{j1} \cdot m}$, $vt_{j2} = g^{c_i \cdot H_{k_{i1}}(m) - r_{j1}}$. Third, the vehicle embeds the following four parameters: $\delta = m \oplus vs_{j0} \oplus vs_{j1} \oplus vs_{j2} \oplus vs_{j3} \oplus vt_{j1} \oplus vt_{j2}$, $vs_{j4} = j \cdot H_{k_{i2}}(m) + r_{j1}$, $vs_{j5} = r_{j1} \cdot m + r_{j2} \cdot \delta$, $vs_{j6} = g^{r_{j2}}$, $vs_{j7} = g^{r_{j3} \cdot sk_j^i}$.

3) The vehicle concatenates all these computed data $\mathcal{E}_j^i = m || vs_{j0} || vs_{j1} || vs_{j2} || vs_{j3} || \delta || vs_{j4} || vs_{j5} || vs_{j6} || vs_{j7}$, and submits the event report $\mathcal{E}_j^i$.

C. Event Verification

Upon receiving an event report from the RSU, the cloud will first verify its correctness with the following three steps.

1) The cloud computes $< vt'_{j1}, vt'_{j2} >$ based on the data extracted from $\mathcal{E}^i_j$, where $vt'_{j1} = g^{vs_{j5}} / \left((vs_{j6})^{\delta}\right)$, and $vt'_{j2} = vs_{j1}/g^{vs_{j4}}$. This procedure is the reverse process of computing the secret $< vt_{j1}, vt_{j2} >$ in the event report.

2) The cloud computes the item δ' based on vt'_{j1}, vt'_{j2}, and the data extracted from $\mathcal{E}^i_j$, where $\delta' = m \oplus vs_{j0} \oplus vs_{j1} \oplus vs_{j2} \oplus vs_{j3} \oplus vt'_{j1} \oplus vt'_{j2}$.

3) Finally, the cloud compares the computed δ' with the δ extracted from $\mathcal{E}^i_j$. If they are equal, the cloud accepts the event report $\mathcal{E}^i_j$. Otherwise, he will reject it, i.e., the event report is modified by attacks, and cannot be used for further functions.

D. Attacks Detection

Before the cloud performs the attack detection, it first divides these event report into groups, i.e., the homogeneous events are grouped together. This is essential for a common baseline to detect attacks. For each group, the cloud server detects a probable attack with the same method. For easy description, we describe the attack detection method in one group. The key idea of our scheme is to gather α homogeneous event reports in the group, and use them to reconstruct a predefined data. If these α event reports are generated by α different and benign vehicles, we can easily reconstruct the predefined data; otherwise, we cannot reconstruct that data, i.e., a probable attack is detected. The attack detection method is illustrated with the following three steps.

1) The cloud computes four items:

$$\begin{aligned} t_0 &= \frac{vs_{j0}}{vs_{j'0}} = e(g,g)^{a_i(j-j')+(j-j')\cdot H_{k_{i1}}(m)} \\ t_1 &= \frac{vs_{j1}}{vs_{j'1}} = g^{(j-j')\cdot H_{k_{i2}}(m)} \\ t_2 &= e\left(t_1, vs_{j2}\right) = e(g,g)^{(j-j')\cdot H_{k_{i1}}(m)} \\ \frac{\varphi_j}{\varphi_{j'}} &= t_0/t_2 = e(g,g)^{a_i(j-j')} \end{aligned}$$

2) The cloud computes the lagrange coefficient

$$\Delta_{\varphi_j,\Omega}(0) = \prod_{\substack{\varphi_{j'}\in\Omega, \\ \varphi_{j'}\neq\varphi_j}} \frac{-\varphi_{j'}}{\varphi_j - \varphi_{j'}} = \prod_{\substack{\varphi_{j'}\in\Omega, \\ \varphi_{j'}\neq\varphi_j}} \frac{1}{1-\varphi_j/\varphi_{j'}} \quad (1)$$

3) The cloud detects whether $e(g^{s_i}, vs_{j2})$ can be reconstructed with the following equation.

$$\begin{aligned} &\prod_{\varphi_j\in\Omega} (vs_{j3})^{\Delta_{\varphi_j,\Omega}(0)} \\ &= \prod_{\varphi_j\in\Omega} \left(e(g,g)^{sk^i_j\cdot H_{k_{i1}}(m)/H_{k_{i2}}(m)}\right)^{\Delta_{\varphi_j,\Omega}(0)} \\ &= e(g,g)^{\sum_{\varphi_j\in\Omega} sk^i_j\cdot\Delta_{\varphi_j,\Omega}(0)\cdot H_{k_{i1}}(m)/H_{k_{i2}}(m)} \\ &= e(g,g)^{s_i\cdot H_{k_{i1}}(m)/H_{k_{i2}}(m)} = e(g^{s_i}, vs_{j2}) \end{aligned} \quad (2)$$

As we can see, for the same event m, if α vehicles' homogeneous event reports can be used to reconstruct the data $e(g^{s_i}, vs_{j2})$, then no attack is detected; otherwise, there would be an attack.

E. Attacks Detection Analysis

In this subsection, we give the following examples to show how attacks are detected by our proposed methods.

1) Sybil Attack Detection: A malicious vehicle launches a sybil attack by sending the same event with multiple event reports, as if these event reports come from different vehicles. In our scheme, the cloud checks α event reports each time, since each vehicle only has one lawful secret share, if these event reports come from α different and lawful vehicles, then the pre-defined data can be reconstructed; otherwise, the sybil attack can be detected.

2) Fabrication Attack Detection: A malicious vehicle would also fabricate another vehicle to submit a false event report. Now we describe how our scheme detects the fabrication attack.

From our previous illustration, we observe that, a vehicle, say $V^i_{j'}$, would successfully forge $< vs_{j0}, vs_{j1}, vs_{j2}, vs_{j3} >$ of another vehicle in the same group, say V^i_j. The process is described as follows:

1) $V^i_{j'}$ intercepts V^i_j's event report $\mathcal{E}^i_j$, and extracts $< m, vs_{j0}, vs_{j1}, vs_{j2}, vs_{j3} >$.
2) $V^i_{j'}$ computes $< H_{k_{i1}}(m), H_{k_{i2}}(m), g^{c_i\cdot H_{k_{i1}}(m)}, g^{H_{k_{i1}}(m)/H_{k_{i2}}(m)} >$.
3) Based on the above data, $V^i_{j'}$ can easily obtain $< sd^i_j \cdot g^j, g^j, e(g,g)^{sk^i_j} >$ of V^i_j.
4) Now $V^i_{j'}$ can fabricate $< vs_{j0}, vs_{j1}, vs_{j2}, vs_{j3} >$ of $V^i_{j'}$ for another event based on the above computation results.

However, this fabrication attack would not succeed. This is because a successful fabrication also needs to forge other secret data, e.g., vs_{j4}. Without knowing V^i_j's secret ID, i.e., j, $V^i_{j'}$ cannot forge vs_{j4}. As a result, this fabrication attack can be easily detected by the event verification algorithm.

3) Injection Attack Detection: To launch a successful injection attack, the attackers have to forge the data items $< vs_{j0}, vs_{j1}, vs_{j3}, vs_{j4}, vs_{j5} >$. Without knowing the secret keys of the vehicles, the injection attack cannot succeed.

4) Replay Attack Detection: Attackers would launch a replay attack by repeating the previous event reports. Our scheme defends against the replay attack from two aspects. First, we embed the time in the event report $\mathcal{E}$, if the attacker simply repeats an outdated event report, it would be easily detected since it would cause the time to be inconsistent. Second, the cloud stores all data within time t, if the attacker repeats an event report where t is still valid, the cloud would easily detect the replay attack by searching its copy.

F. Attackers Identification

Once an attack is detected, the cloud would send the α suspicious vehicles' data to the corresponding TA. With the secret keys, the TA can soon find the attackers who launch the attack. The identification is achieved by the TA with the following steps.

1) Computes the $g^{c_i\cdot H_{k_{i1}}(m)}$, and $H_{k_{i2}}(m)$.
2) Compute g^j, where $g^j = \left(vs_{j1}/g^{c_i\cdot H_{k_{i1}}(m)}\right)^{1/H_{k_{i2}}(m)}$
3) Compute $sd^i_j = (g^j)^a \cdot g^b$

4) Search sd_j^i from the TA's secret table, find the entry $< j, sd_j^i, sk_j^i >$, and further check whether this vehicle's secret data are correctly generated.

G. Attackers Revocation

When the TA identifies a specific vehicle to be the attacker, say V_j^i, he broadcasts the data $g^{sk_j^i}$, which is based on the secret key sk_j^i of V_j^i, to all the RSUs and vehicles to inform them not to accept or relay the revoked vehicles' data. The method of identifying the revoked data can also be efficiently achieved. Specifically, when an event report is received, the revoked vehicle's event report can be easily detected if $e(g^{sk_j^i}, vs_{j2}) = vs_{j3}$ holds. When a RSU receives the revoked data $g^{sk_j^i}$, he will store it in his memory. Once a revoked vehicle still injects data, the RSU can detect it and refuse to relay this vehicle's data to the cloud.

We note that during the revocation process, the TA only needs to announce $g^{sk_j^i}$ to the public, where the secret key of the revoked vehicle is still preserved. Additionally, we do not announce the ID of the revoked vehicle, this design preserves the privacy of vehicle even if it is revoked. However, for forensic concerns, the TA can easily track the revoked vehicles.

V. Security Proof

In this section, we formulate the security goals achieved by our scheme, and present the detailed security proof.

Theorem 1. *If a probabilistic polynomial-time adversary can break our scheme in the untraceability game, then we can construct a simulator $\mathcal{B}$ to break the DMBDH (Decisional Modified Bilinear Diffie-Hellman) game [17] with a non-negligible advantage.*

Proof. Assume a probabilistic polynomial-time adversary $\mathcal{A}$ has a non-negligible advantage ϵ against our scheme in the untraceability game. Then we can build a simulator $\mathcal{B}$ that plays the DMBDH game with advantage $\epsilon/4$. The challenger flips a fair coin γ outside of $\mathcal{B}$'s view. If $\gamma = 0$, he sends $(A, B, C, Z) = (g^a, g^b, g^c, e(g,g)^{\frac{ab}{c}})$ to $\mathcal{B}$; otherwise he sends $(A, B, C, Z) = (g^a, g^b, g^c, e(g,g)^z)$ to $\mathcal{B}$, where $a, b, c, z \in \mathbb{Z}_p$, are randomly generated. The goal of $\mathcal{B}$ is to guess γ' for γ by interacting with $\mathcal{A}$ and playing the following game.

Setup: The simulator $\mathcal{B}$ sends the public keys $(g^{c_i}, g_i^s), i \in [1, N]$ to $\mathcal{A}$.

Phase 1: The adversary $\mathcal{A}$ queries the event report for event m_i for polynomial times. $\mathcal{B}$ keeps a list l, which is initially empty, to record the query. If m_i is recorded in l, he selects an unused ID j to generate event report $\mathcal{E}_j^i$; otherwise, he randomly selects an ID j to generate the event report $\mathcal{E}_j^i$. After each generation, $\mathcal{B}$ adds a pair $< j, m_i >$ to l.

Challenge: The adversary $\mathcal{A}$ submits two events m_0 and m_1 to $\mathcal{B}$. $\mathcal{B}$ flips a coin μ to determine whether he uses the same ID's secret key to generate the event report for m_0 and m_1, i.e., if $\mu = 0$, he uses the same ID's secret key to generate an event report for m_0 and m_1; otherwise he uses two different IDs to generate the event reports. Then $\mathcal{B}$ selects the IDs j_0 and j_1(if $\mathcal{B}$ uses the same ID, then $j_0 = j_1$), where the pair $< m_0, j_0 >$, and $< m_1, j_1 >$ are not recorded in l, to generate the event reports. Now we describe the generation in two cases. The intuition behind the following assignments is that, if we use the same vehicle j's secret data to generate event reports for m_0, and m_1, we implicitly set the secret key $sk_j^i = \frac{c}{a}$, $H_{k_{i1}}(m_0) = r_1 b$, $H_{k_{i1}}(m_1) = (r_1 - r_2) \cdot r_6 \cdot b$, $vs_{03}/vs_{13} = Z$.

Case 1: $\mathcal{B}$ sets $\mu = 0$, then $\mathcal{B}$ uses j's secret data to generate the event reports for m_0 and m_1. $\mathcal{B}$ first generates random tuples $(r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8, r_9)$.

For m_0, $\mathcal{B}$ simulates the event report as follows,
$vs_{00} = e(g, sd_j^i \cdot B^{r_1 \cdot j}) = e(g,g)^{a_i \cdot j + b_i + j \cdot r_1 \cdot b}$,
$vs_{01} = g^{j \cdot r_2} \cdot B^{c_i \cdot r_1} = g^{j \cdot r_2 + c_i \cdot r_1 \cdot b}$,
$vs_{02} = B^{r_1/r_2} = g^{b \cdot (r_1/r_2)}$,
$vs_{03} = Z^{r_1/r_2}$, $vt_{01} = g^{r_3 \cdot m}$, $vt_{02} = B^{r_1 \cdot c_i} = g^{c_i \cdot r_1 \cdot b - r_3}$,
$\delta = m_0 \oplus vs_{00} \oplus vs_{01} \oplus vs_{02} \oplus vs_{03} \oplus vt_{01} \oplus vt_{02}$,
$vs_{04} = j \cdot r_2 + r_3$, $vs_{05} = r_3 \cdot m + r_4 \cdot \delta$, $vs_{06} = g^{r_4}$,
$vs_{07} = A^{r_5} = g^{a \cdot r_5} = g^{r_5' \cdot \frac{a}{c}}$, where $g^{r_5'} = C^{r_5} = g^{c \cdot r_5}$,
$\mathcal{E}_0^i = m_0 || vs_{00} || vs_{01} || vs_{02} || vs_{03} || \delta || vs_{04} || vs_{05} \ || vs_{06} || vs_{07}$.

For m_1, $\mathcal{B}$ simulates the event report as follows,
$vs_{10} = e(g, sd_j^i \cdot B^{(r_1 - r_2) \cdot r_6 \cdot j}) = e(g,g)^{a_i \cdot j + b_i + j \cdot (r_1 - r_2) \cdot r_6 \cdot b}$,
$vs_{11} = g^{j \cdot (r_2 \cdot r_6)} \cdot B^{c_i \cdot ((r_1 - r_2) \cdot r_6)} = g^{j \cdot (r_2 \cdot r_6) + c_i \cdot ((r_1 - r_2) \cdot r_6) \cdot b}$,
$vs_{12} = B^{(r_1 - r_2)/r_2} = g^{b \cdot (r_1 - r_2)/r_2}$, $vs_{13} = Z^{(r_1 - r_2)/r_2}$,
$vt_{11} = g^{r_7 \cdot m}$, $vt_{12} = B^{((r_1 - r_2) \cdot r_6) \cdot c_i} = g^{c_i \cdot ((r_1 - r_2) \cdot r_6) \cdot b - r_7}$,
$\delta = m_1 \oplus vs_{10} \oplus vs_{11} \oplus vs_{12} \oplus vs_{13} \oplus vt_{11} \oplus vt_{12}$,
$vs_{14} = j \cdot (r_2 \cdot r_6) + r_7$, $vs_{15} = r_7 \cdot m + r_8 \cdot \delta$, $vs_{16} = g^{r_8}$,
$vs_{17} = A^{r_9} = g^{a \cdot r_9} = g^{r_9' \cdot \frac{a}{c}}$, where $g^{r_9'} = C^{r_9} = g^{c \cdot r_9}$,
$\mathcal{E}_1^i = m_1 || vs_{10} || vs_{11} || vs_{12} || vs_{13} || \delta || vs_{14} || vs_{15} \ || vs_{16} || vs_{17}$.

Case 2: $\mathcal{B}$ sets $\mu = 1$, then $\mathcal{B}$ uses j_0's secret data to generate the event reports for m_0, and j_1's secret data to generate the event report for m_1. In case 2, $\mathcal{B}$ uses the similar method to generate the event report, the only difference is that, when generating vs_{03}, and vs_{13}, $\mathcal{B}$ randomly chooses $z_0, z_1 \in \mathcal{Z}_p$, and sets $vs_{03} = e(g,g)^{z_0}$, and $vs_{13} = e(g,g)^{z_1}$.

After the generation, $\mathcal{B}$ adds $< m_0, j_0 >$, $< m_1, j_1 >$ to l. Finally, $\mathcal{B}$ returns generated event reports to the adversary.

Phase 2: Phase 1 is repeated.

Guess: $\mathcal{A}$ outputs its guess $\mu' \in \{0, 1\}$ for μ. If $\mu' = \mu = 0$, $\mathcal{B}$ outputs $\gamma' = 0$ to indicate that it is given a DMBDH tuple; otherwise, $\mathcal{B}$ outputs $\gamma' = 1$.

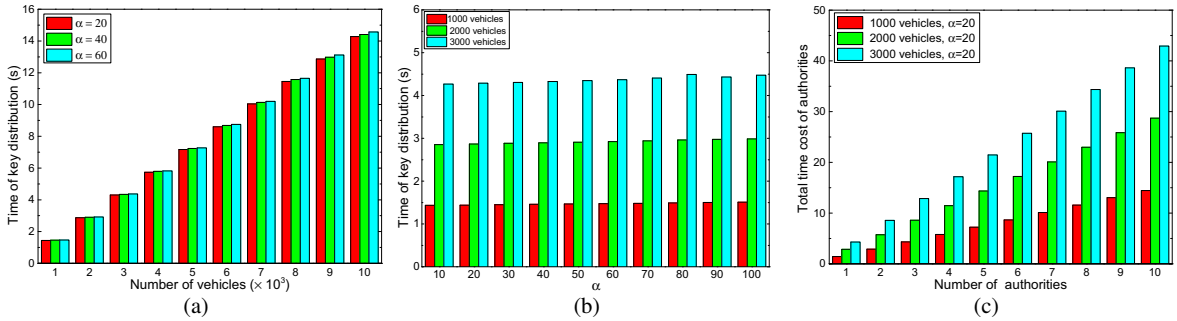
As shown above, $\mathcal{B}$'s method of generating the event reports is the same with that of our scheme.

In the case that $\gamma = 1$, the adversary gains no information about μ, therefore, $\Pr[\gamma' = \gamma | \gamma = 1] = \frac{1}{2}$.

In the case that $\gamma = 0, \mu = 1$, the adversary also gains no information about μ, thus, $\Pr[\gamma' = \gamma | \gamma = 0, \mu = 1] = \frac{1}{2}$.

In the case that $\gamma = \mu = 0$, the adversary can compute $vs_{03}/vs_{13} = Z$, if Z is equal to $e(g,g)^{\frac{ab}{c}}$, then the two event reports come from the same vehicle, i.e., the vehicle is tracked by the adversary. Since the adversary advantage is ϵ, therefore, $\Pr[\gamma' = \gamma | \gamma = \mu = 0] = \frac{1}{2} + \varepsilon$.

The overall advantage of $\mathcal{B}$ in the DMBDH game is $\frac{1}{2}\Pr[\gamma' = \gamma | \gamma = 1] + \frac{1}{4}\Pr[\gamma' = \gamma | \gamma = 0, \mu = 1] + \frac{1}{4}\Pr[\gamma' = \gamma | \gamma = \mu = 0] - \frac{1}{2} = \frac{\varepsilon}{4}$ □



(a) (b) (c)

Fig. 2: Time cost of vehicle registration.

VI. Tunable Detection Frequency

In the aforementioned sections, we introduce how to perform secure construction to detect the probable attacks launched by the malicious vehicles, without compromising the privacy of benign vehicles. From the analysis, we know that, once the number of homogeneous vehicles(the vehicles report the same event and belong to the same TA) is less than the threshold α, the cloud cannot reconstruct the secret. An alternative way is to wait until α data are gathered; however, this will cause some delay for attack detection. In this section, we introduce the enhanced secure construction, which achieves a tunable detection frequency without being affected by the delay in the VANETs.

The key idea of the enhanced construction is to pre-set $\alpha - 1$ decoy items in the RSU (under the safeguard of its accompanied TA), when a RSU finds that the number of homogeneous vehicles is less than the threshold α, he will pad some event reports with the decoy items, so that the number of homogeneous vehicles would be greater than $\alpha - 1$. From the viewpoint of the cloud server, he would not know whether an event report is generated by the vehicles or RSU. Therefore, the cloud can use the aforementioned method to detect probable attacks.

A. Distributing Keys for Decoy Items

To enable the cloud to detect probable attacks timely, we propose to pre-set some decoy items on the RSUs (under the safeguard of their accompanied TA). On one hand, the newly-introduced decoy items should help achieve timely detection, on the other hand, these decoy items should not be used to break the privacy of vehicles. Tab. I shows the keys generated by an authority TA_i.

B. Enhanced Event Report

Compared with the aforementioned event report, we only need to redesign the following four data items, i.e., $vs_{j3} = e(g,g)^{sk_j^i \cdot H_{k_{i2}}(m)}$, $vs_{j4} = g^{d_i \cdot H_{k_{i1}}(m)}$, $vs_{j5} = g^{H_{k_{i2}}(m)}$, $vs_{j6} = g^{c_i \cdot H_{k_{i1}}(m) - d_i \cdot H_{k_{i2}}(m)}$. These data would help the RSUs generate a lively event report with the decoy items.

C. Event Report with Decoy Data

Before a RSU submits its collected data to the cloud, the RSU will first check whether the number of vehicles in each group is greater than α. Assume only α' homogeneous vehicles submit event reports, where $0 < \alpha' < \alpha$, the RSU would generate $\alpha - \alpha'$ lively event reports with $\alpha - \alpha'$ decoy items. For the lth decoy data, the RSU reconstructs a lively event report with the following steps:

1) The RSU extracts $< m, vs_{j2}, vs_{j4}, vs_{j5}, vs_{j6} >$ from the vehicle V_j^i's event report $\mathcal{E}_j^i$, and sets $vs_{l2} = vs_{j2}$, $vs_{l4} = vs_{j4}$, $vs_{l5} = vs_{j5}$, $vs_{l6} = vs_{j6}$.

2) With the decoy items and the extracted data, the RSU computes $vs_{l0} = p_{d0} \cdot e(p_{d1}, vs_{j4}) = e(g,g)^{a \cdot r_{i,l} + r_{i,l} \cdot H_{k_{i1}}(m)}$, $vs_{l1} = vs_{j5}^{p_{d2}} \cdot vs_{j6}$, and $vs_{l3} = e(g, vs_{j5})^{p_{d3}} / e(g^d, vs_{j5}) = e(g,g)^{sk_l^i \cdot H_{k_{i2}}(m)}$, where $sk_l^i = F(e(g,g)^{a_i \cdot r_{i,l}})$.

3) The RSU computes four items, where $vt_{l1} = g^{r_{l1} \cdot (p_{d2} - p_{d3})}$, $\delta = m \oplus vs_{l0} \oplus vs_{l1} \oplus vs_{l3} \oplus vs_{l4} \oplus vs_{l5} \oplus vs_{l6} \oplus vt_{l1}$, $vs_{l7} = r_{l1} \cdot (p_{d2} - p_{d3}) \cdot m + \delta \cdot r_{l2}$, $vs_{l8} = g^{r_{l2}}$.

4) Finally, The RSU outputs the event report $\mathcal{E}_j^i$. As we can see, from the viewpoint of the cloud, there is no difference between the event reports generated by vehicles and those generated with the decoy items.

TABLE I: Keys generated during the key distribution

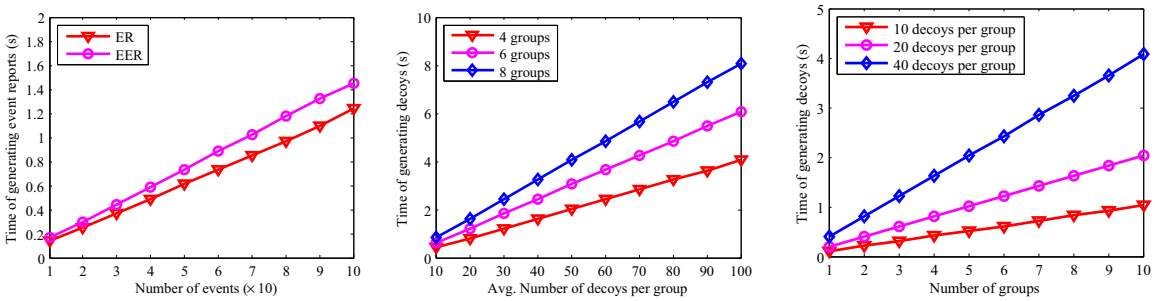
TA_i's secret keys	$a_i, b_i, c_i, d_i, k_{i1}, k_{i2}, s_i$
TA_i's public keys	g^{c_i}, g^{d_i}, g^{s_i}
V_j^i's secret data	j, $sd_j^i = g^{a_i \cdot j + b}$; $sk_j^i = F(e(g,g)^{a_i \cdot j})$
V_j^i's keyed hash function	$H_{k_{i1}}(x)$, $H_{k_{i2}}(x)$
$RSUs$' decoy items	$p_{d0} = e(g,g)^{a_i \cdot r_{i,l} + b_i}$; $p_{d1} = g^{r_{i,l}/d_i}$; $p_{d2} = r_{i,l} + d_i$; $p_{d3} = F(e(g,g)^{a_i \cdot r_{i,l}}) + d_i$

D. Enhanced Event Verification

The cloud verifies the event report with a similar method described in Section IV.

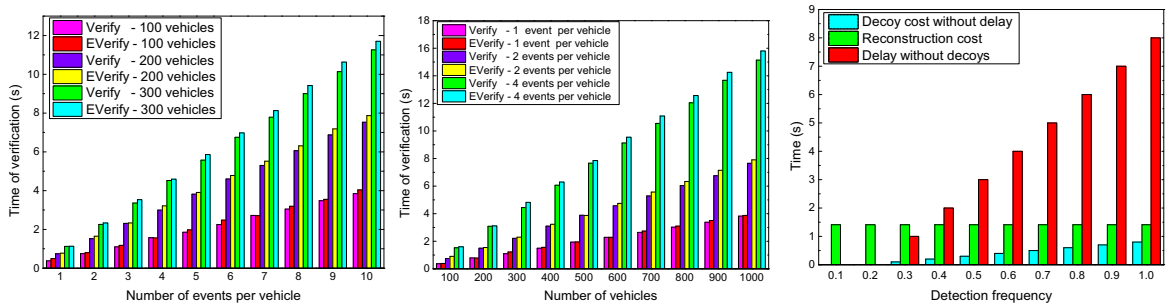
E. Attacks Detection

From the cloud's viewpoint, there is no difference between the event report generated by vehicles and those generated with the decoy items. Therefore, the cloud will use the similar methods described in Section IV to detect probable attacks.



(a) Vehicles (b) RSU (c) RSU

Fig. 3: Time cost of generating event reports.



(a) Verification cost (b) Verification cost (c) Tunable detection frequency

Fig. 4: Time cost of operation.

Specifically, the cloud will check whether the $e(g^{s_i}, vs_{j5})$ can be reconstructed to determine whether an attack is occurred.

$$\prod_{\varphi_j \in \Omega} (vs_{j3})^{\Delta_{\varphi_j,\Omega}(0)} = \prod_{\varphi_j \in \Omega} \left(e(g,g)^{sk_j^i \cdot H_{k_{i2}}(m)}\right)^{\Delta_{\varphi_j,\Omega}(0)}$$
$$= e(g,g)^{\sum_{\varphi_j \in \Omega} sk_j^i \cdot \Delta_{\varphi_j,\Omega}(0) \cdot H_{k_{i2}}(m)} = e(g^{s_i}, vs_{j5}) \quad (3)$$

VII. Performance Evaluation

A. Evaluation Settings

The experiment programs are coded using the Python programming language on a PC with 3.4GHZ Intel Core CPU and 16GB memory. We adopt the cryptographic framework and settings proposed in [21], and implement all necessary routines for authorities to generate secret keys during vehicle registration, for vehicles to report messages, for cloud to detect probable attacks, and for the trusted authority to identify the attackers.

B. Evaluation Results

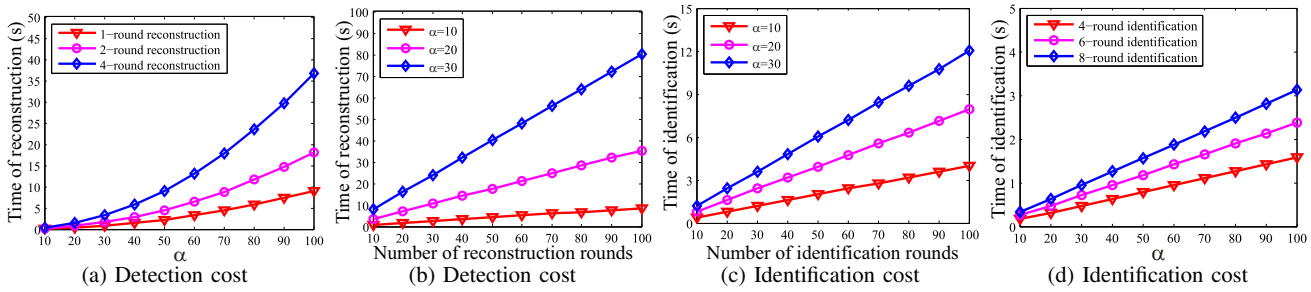
Fig. 2 shows the time cost of vehicle registration. From Fig. 2 (a) and (b), we observe that, the more vehicles that are required to be registered, the more time that is spent on registration. Meanwhile, the degree of the distribution function F, i.e., α, has little impact for the registration cost. Fig. 2 (c) shows that, the more authorities are involved, the more total time is spent by these authorities.

Fig. 3 depicts the time cost of event report generation. Fig. 3(a) shows the time cost of generating event reports for vehicles. We observe that, the enhanced event report requires a bit more time. The fundamental reason is that, the enhanced event report requires the vehicle to generate more data items to help the RSU generate decoy items. Figs. 3(b) and (c) show the time cost of generating decoy items for RSUs. As we can see, as the average number of decoys in each group and number of groups increase, the time cost of generating decoys would increase correspondingly.

Fig. 4(a) and (b) demonstrate the time cost of event verification. We observe that, with the number of vehicles, and the number of events per vehicle increase, the time cost of the verification algorithm and the enhanced verification algorithm increases. The reason here is clear: with vehicles submitting more events, the cloud has to spend more time on conducting the verification.

In Fig. 4 (c), for a easy understanding, we set $\lambda = 20$, i.e., the ratio that vehicles get through a RSU is 25, $\rho_i = 0.1$, i.e., vehicles registered with a specific authority occupies 10% of vehicles getting through the RSU, and $\alpha = 20$, i.e., the cloud needs to collect 20 homogeneous vehicles' data to conduct a detection. From the figure, we observe that, without decoys, when the detection frequency increases, the detection would suffer from serious delay. Conversely, if we introduce the decoys, there is no detection delay at the expense of a small decoy cost for the RSU. We can also see that, as the detection frequency increases, the reconstruction cost remains nearly the same, which demonstrates that our design adapts to a tunable detection frequency without being affected by the delay.

Fig. 5 illustrates the time cost of detecting and identifying attackers. Fig. 5(a) shows that, as the degree of distribution



(a) Detection cost (b) Detection cost (c) Identification cost (d) Identification cost

Fig. 5: Time cost of detecting and identifying attackers.

function, α, increases, the time cost of reconstructing the predefined secret would increase quadratically. The fundamental reason for this is that, during the process of reconstructing the secret, we need to conduct $O(\alpha^2)$ computations. Since we often assign a small α in real applications, the value of α would not affect the performance of the whole scheme. Fig. 5(c) and (d) demonstrate the time cost of identifying the attackers. As we can see, as the number of identification rounds and α increase, the identifying cost would accordingly increase linearly.

VIII. Conclusion

In this paper, for the first time, we design a privacy preserving attack detection scheme in the vehicle cloud computing. First, we propose involving multiple authorities to distribute keys independently. This design will adapt our scheme to many registration scenarios. Then we present the details of secure constructions, which enable the cloud to smartly detect many potential attacks without compromising the privacy of vehicles. Meanwhile, our constructions preserve the privacy of vehicles even if they are revoked, but for forensic concerns, the TA can easily track the revoked vehicles. We also investigate the time delays in the vehicle network, and propose a corresponding construction to detect potential attacks with a tunable detection frequency. We further define an untraceability model, and show that our scheme achieves untraceability through rigorous security proof. Finally, we conduct extensive experiments to validate the efficacy and efficiency of our scheme.

Acknowledgment

This work is supported in part by the National Natural Science Foundation of China (Project No. 61472125, 61173038).

References

[1] S. Ezell, "Explaining international it application leadership: Intelligent transportation systems," *The Information Technology & Information Foundation*, 2010.

[2] M. Gerla, "Vehicular cloud computing," in *Ad Hoc Networking Workshop (Med-Hoc-Net), 2012 The 11th Annual Mediterranean*. IEEE, 2012, pp. 152–155.

[3] P. Papadimitratos, L. Buttyan, T. Holczer, E. Schoch, J. Freudiger, M. Raya, Z. Ma, F. Kargl, A. Kung, and J.-P. Hubaux, "Secure vehicular communication systems: design and architecture," *Communications Magazine, IEEE*, vol. 46, no. 11, pp. 100–109, 2008.

[4] R. Lu, X. Lin, H. Zhu, P.-H. Ho, and X. Shen, "Ecpp: Efficient conditional privacy preservation protocol for secure vehicular communications," in *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*. IEEE, 2008.

[5] A. Studer, E. Shi, F. Bai, and A. Perrig, "Tacking together efficient authentication, revocation, and privacy in vanets," in *Sensor, Mesh and Ad Hoc Communications and Networks, 2009. SECON'09. 6th Annual IEEE Communications Society Conference on*. IEEE, 2009, pp. 1–9.

[6] T. Zhou, R. R. Choudhury, P. Ning, and K. Chakrabarty, "P2dapłsybil attacks detection in vehicular ad hoc networks," *Selected Areas in Communications, IEEE Journal on*, vol. 29, no. 3, pp. 582–594, 2011.

[7] X. Lin, "Lsr: mitigating zero-day sybil vulnerability in privacy-preserving vehicular peer-to-peer networks," *Selected Areas in Communications, IEEE Journal on*, vol. 31, no. 9, pp. 237–246, 2013.

[8] S. Kumar, S. Gollakota, and D. Katabi, "A cloud-assisted design for autonomous driving," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, 2012, pp. 41–46.

[9] J. Wang, J. Cho, S. Lee, and T. Ma, "Real time services for future cloud computing enabled vehicle networks," in *Wireless Communications and Signal Processing (WCSP), 2011 International Conference on*. IEEE, 2011, pp. 1–5.

[10] G. Yan, D. Wen, S. Olariu, and M. C. Weigle, "Security challenges in vehicular cloud computing," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 14, no. 1, pp. 284–294, 2013.

[11] M. Abuelela and S. Olariu, "Taking vanet to the clouds," in *Proceedings of the 8th International Conference on Advances in Mobile Computing and Multimedia*. ACM, 2010, pp. 6–13.

[12] S. Olariu, T. Hristov, and G. Yan, "The next paradigm shift: from vehicular networks to vehicular clouds," *Mobile ad hoc networking: cutting edge directions. 2nd ed. NJ, USA: John Wiley & Sons, Inc., Hoboken*, 2013.

[13] S. Olariu, I. Khalil, and M. Abuelela, "Taking vanet to the clouds," *International Journal of Pervasive Computing and Communications*, vol. 7, no. 1, pp. 7–21, 2011.

[14] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[15] G. R. Blakley, "Safeguarding cryptographic keys," in *Managing Requirements Knowledge, International Workshop on*. IEEE Computer Society, 1899, pp. 313–313.

[16] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Advances in Cryptology-CRYPTO 2001*. Springer, 2001, pp. 213–229.

[17] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Advances in Cryptology–EUROCRYPT 2005*. Springer, 2005, pp. 457–473.

[18] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM conference on Computer and communications security*. Acm, 2006, pp. 89–98.

[19] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Security and Privacy, 2007. SP'07. IEEE Symposium on*. IEEE, 2007, pp. 321–334.

[20] B. Wang, W. Song, W. Lou, and Y. T. Hou, "Inverted index based multi-keyword public-key searchable encryption with strong privacy guarantee," in *INFOCOM, 2015 Proceedings IEEE*. Hong Kong: IEEE, 2015, pp. 2092–2110.

[21] J. A. Akinyele, C. Garman, I. Miers, M. W. Pagano, M. Rushanan, M. Green, and A. D. Rubin, "Charm: a framework for rapidly prototyping cryptosystems," *Journal of Cryptographic Engineering*, vol. 3, no. 2, pp. 111–128, 2013.