sensors

MDPI

Article

RPVC: A Revocable Publicly Verifiable Computation Solution for Edge Computing

Zi Jiao, Fucai Zhou *, Qiang Wang and Jintong Sun

Software College, Northeastern University, Shenyang 110169, China; 1910470@stu.neu.edu.cn (Z.J.); wangq3635@126.com (Q.W.); 2010510@stu.neu.edu.cn (J.S.)
* Correspondence: fczhou@mail.neu.edu.cn

Abstract: With publicly verifiable computation (PVC) development, users with limited resources prefer to outsource computing tasks to cloud servers. However, existing PVC schemes are mainly proposed for cloud computing scenarios, which brings bandwidth consumption or network delay of IoT devices in edge computing. In addition, dishonest edge servers may reduce resource utilization by returning unreliable results. Therefore, we propose a revocable publicly verifiable computation(RPVC) scheme for edge computing. On the one hand, RPVC ensures that users can verify the correct results at a small cost. On the other hand, it can revoke the computing abilities of dishonest edge servers. First, polynomial commitments are employed to reduce proofs' length and generation speed. Then, we improve revocable group signature by knowledge signatures and subset covering theory. This makes it possible to revoke dishonest edge servers. Finally, theoretical analysis proves that RPVC has correctness and security, and experiments evaluate the efficiency of RPVC.

Keywords: publicly verifiable computation; revocable group signature; outsource computing; edge computing

check for updates

Citation: Jiao, Z.; Zhou, F.; Wang, Q.; Sun, J. RPVC: A Revocable Publicly Verifiable Computation Solution for Edge Computing. *Sensors* **2022**, *22*, 4012. https://doi.org/10.3390/s22114012

Academic Editor: Marco Picone

Received: 10 April 2022
Accepted: 16 May 2022
Published: 25 May 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The rapid popularization of smart devices has spawned a large number of Internet of Things (IoT) applications, one of which is the Internet of Vehicles (IoV). The reason why vehicles tend to outsource computing tasks that include road conditions and vehicle information to cloud servers during their travel is that the computing resources are limited. Edge computing can improve the response speed and user experience. As a bridge between users and cloud servers, on the one hand, they improve response speed by sharing part of the cloud computing, while users with the limited resource can rely on them to reduce computing pressure. One specific implementation is the Intelligent Transport System (ITS) [1,2], as shown in Figure 1, which is used to help users receive the best driving plan in current road and traffic conditions as soon as possible. There are four participants in ITS: User, road side unit (RSU, which can be seen as an edge server), cloud server, and car manufacturers. The car manufacturer dispatches functions for making a driving plan for the cloud server. The edge server downloads the function from the cloud server. The user sends vehicle parameters to the edge server. The edge server returns results to the user.

However, ITS has the following problems: (1) The cloud server may tamper with the functions uploaded by the car manufacturer, and the edge server may provide users with incorrect results [3]. (2) When a user is driving, the vehicle needs to switch among RSUs that serve different areas. To verify signature messages from a specific RSU, a large public key list is needed [4]. This results in overhead storage for users and overhead computation for finding public keys (3) Once a user receives messages from an edge server that it has never met, frequent communication brought by public key transmission will cause delays (4) If the identity of the edge server is exposed, adversaries can use the same attack method to threaten edge servers with similar configurations.

From the example of IoV, the requirements for edge computing are as follows: (1) Results returned by the edge server should be verifiable, and a dishonest edge server can be revoked. (2) The time for the user to verify the result and the number of keys stored should be minimized. (3) Key transmission processes between users and new edge servers should be minimized. (4) The identity of the edge server should be anonymous to users.

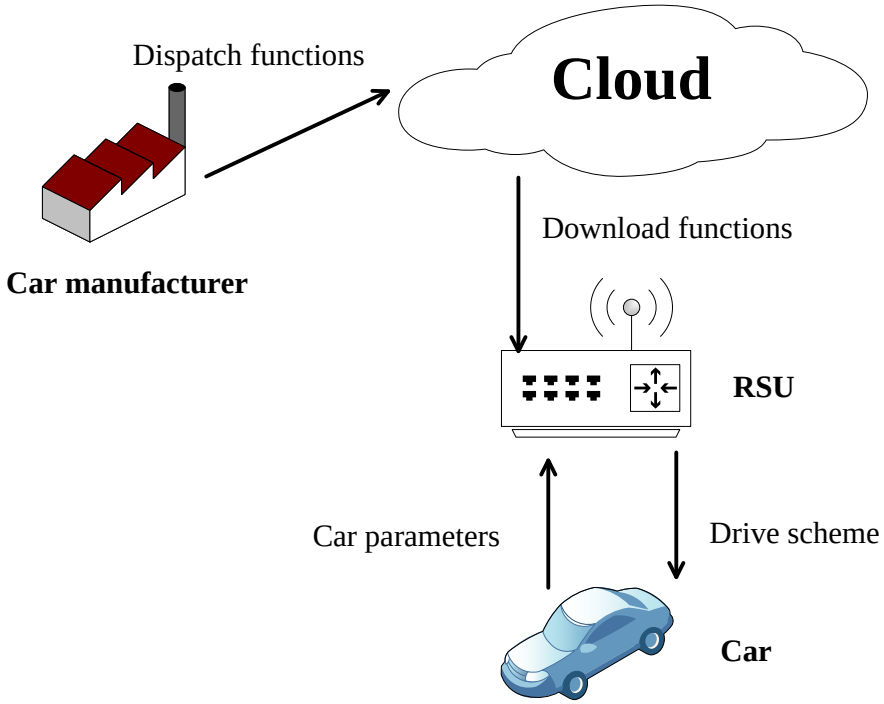

Figure 1. The architecture of a typical ITS application.

For requirement 1, verifiable computation (VC) [5] can be used to ensure the result is correct. However, the verifier in VC schemes can only be the user or the one he specified. Other participants cannot believe in the verification processes or the reliability of results. Therefore, Parno [6] first proposed publicly verifiable computation (PVC) to solve these defects. Since then, Fiore [7] expanded PVC to evaluate the higher-degree polynomial and matrix multiplication. Catalano [8] introduced a one-way function and RSA mathematical hypothesis to improve the computing speed. However, the verification process of the former uses low-efficiency bilinear pairing, and the practical implementation of the latter is very complex. Polynomial commitment [9] achieves two basic goals: making a commitment to a polynomial and providing proof that a specific point belongs to the polynomial. Therefore, the polynomial commitment can be used to improve the efficiency of existing PVC solutions. To revoke dishonest edge servers, James [10] applies the revocable key policy attribute encryption [11,12] to PVC. However, such schemes are based on time-consuming operations such as encryption and decryption, as meanwhile, the revoking process will cause other participants to synchronize the key list. In addition, the latest research [13–15] requires either a trusted computing environment such as SGX or specific hardware support, thus, the scope of their application is limited. Therefore, revocable group signatures are recommended to revoke dishonest edge servers.

For requirements 2–4, group signature schemes are suitable. That is because any group member can make signatures stand for the whole group, and anyone outside the group cannot forge the signature. Verifiers can verify the signature with only one group public key. The verifier only knows that the signature is from a member of the group, but cannot distinguish the specific signer. The group manager can open a group signature to trace the specific signer. When applied to an edge computing scenario, edge servers can form a group and set up a group manager. For users, only one group public key is required to

verify any edge server signed results, thus, reducing delay and key storage. There will be no key transmission process between users and the new edge server, moreover, the identity of the edge server is anonymous to users. The group manager can trace the signature of incorrect results to find the dishonest edge servers, so, a revocable group signature is recommended for revoking their computing ability.

The classical group signature scheme proposed by Camenisch [16] cannot revoke group members. To make group signature revocable, Song [17] proposed a revocable group signature scheme to ensure forward security. However, the time cost increases linearly with the number of group members. Camenisch [18] proposed an accumulator solution, but once the group members join or quit the group frequently, the members still in the group need to update their credentials continually. Inspired by Boneh [19], Brickell [20] presents a revocation list (RL) solution that keeps members in the group from frequently updating their credentials. However, the final signature of this scheme contains nine parts, which leads to the extremely tedious verification process. Moreover, the drawback of the latest research [21] is that there is not an extremely strong privacy demand in an IoV scenario, which will cause resource waste. At the same time, ref. [22,23] based on merkle hash tree, suggested that the storage and computational overhead vary superlinearly along with the number of users who frequently join or quit An attribute tree using secret sharing [24] and Lagrange interpolation impels the users satisfying certain attributes and can decrypt messages under the broadcast encryption [25]. The idea of subset covering or subset difference [26,27] in an attribute tree to reduce search time and communication cost can be used to improve revocable group signature.

Our contributions are as follows:

- We propose a revocable publicly verifiable computation (RPVC) model. Its main ideas are: Using the properties of PVC to ensure the results returned by the edge server are reliable. Using the properties of group signature to reduce the cost of verification and key storage for users, and keep edge server identity anonymous. If the group signature is revocable, the group manager can trace and revoke the dishonest edge server.
- After analyzing the RPVC threat model, four security goals of the RPVC model are summarized according to possible attack methods and available information for adversaries: function binding, result reliability, anonymity, and revocability.
- An RPVC scheme is given. The scheme speeds up the PVC proof generation and verification time with the help of polynomial commitment and improves the revocable group signature with a subset covering idea. Finally, the correctness analysis and security proof of the scheme are provided.
- We implemented the RPVC scheme, and experiments show that the time delay and storage cost of the RPVC scheme is acceptable when it is applied to edge computing scenarios.

2. Related Works

2.1. Publicly Verifiable Computation (PVC)

Verifiable computing (VC) was proposed to verify the outsource computing results by Gennaro [28] via a boolean circuit in 2010. Benabbas [29] expands VC to compute polynomials in a higher degree. Other studies such as [30–32] also consider VC However, the common defect of all the above VC schemes is that the verifier can only be the user or the one he specifies, which limits the promotion of VC. Thus, public verifiable computation by Parno [6] was first proposed to address this shortcoming. Though [33,34] can also achieve PVC, ref. [33] needs an honest user to generate the main private key, which makes the status of users unequal. In addition, the users in [34] obtain a decrypt key by interaction, which leads to low efficiency. Fiore [7], based on the solution from Benabbas, constructed a PVC scheme aimed to solve matric products. However, the verification time is long because of the use of bilinear mapping. Although Catalano [8] introduced one-way hash function and RSA assumption to improve the speed, this is hard to deploy on existing applications. Another solution to achieve efficient PVC is the work by Ding [15], his idea is to use a

trusted computation environment provided by Intel SGX. Similarly, a scheme [13,14] by Fraust and Liu also needs specific hardware support, these schemes are not suitable for complex large-scale networks.

2.2. Revocable Group Signature

Any group member can make group signature stands for the whole group, others cannot forge a group signature. The verifier can verify a group signature by a group public key without finding out the specific signer. The group manager can open the group signature and figure out who makes this signature. The original concept was proposed by Camenisch [16], but this scheme cannot revoke group members. In order to make a group signature revocable, Song [17] developed with a revocable group signature model which can guarantee forward security, but its verification time increases linearly with the number of group members. Camenisch also proposed a scheme based on an accumulator [18], but if group members frequently join or quit the group, another group member needs to update their credential in a timely manner. The same problems also occur in the scheme [22,23] by Yehia and Buser, furthermore, the mechanism of merkle tree makes the scale increase super-linearly with the joining or quitting of users. Brickell [20] put forward a solution based on the local revocation list, which is inspired by the work of Boneh [19]; group members do not need to update their credentials frequently. However, the final signature contains nine parts, and the verification processes are extremely complex. To confirm whether the signer is in the revocation list more efficiently, Nakanishi [35] brings in a subgroup idea, however, his scheme is still based on an accumulator with the same defects as the Camenisch solution. Yue [21] proposed a revocable group signature which can preserve the privacy, but the drawback is the high consumption of computation resources due to the high level security assumption.

3. Preliminaries

3.1. Bilinear Maps and Related Assumptions

Let $\mathbb{G}$ be cyclic additive group, whose order is prime p. $\mathbb{G}$ is generated by g. Define $\mathbb{G}_T$ as multiplicative group with the same order p. The bilinear pairing $e : \mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$ holds three properties: bilinearity, non-degeneracy, and computability. ε denotes negligible value.

- DL Assumption : Given g and $a \xleftarrow{\$} \mathbb{Z}_p^*$, for every adversary $\mathcal{A}_{DL}$, $Pr[\mathcal{A}_{DL}(g, g^a) = a] = \varepsilon$.
- t-polyDH Assumption [26]: Let $\alpha \xleftarrow{\$} \mathbb{Z}_p^*$, given a $(t+1)-$tuple $\langle g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^t} \rangle \in \mathbb{G}^{t+1}$ as input, for every adversary $\mathcal{A}_{t-polyDH}$,
  $Pr[\mathcal{A}_{t-polyDH}(g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^t}) = \langle \phi(x), g^{\phi(x)} \rangle] = \varepsilon$, where $\phi(x) \in \mathbb{Z}_p[x]$.
- t-SDH Assumption [36]: Let $\alpha \xleftarrow{\$} \mathbb{Z}_p^*$, given a $(t+1)-$tuple $\langle g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^t} \rangle \in \mathbb{G}^{t+1}$ as input, for every adversary $\mathcal{A}_{t-SDH}$,
  $Pr[\mathcal{A}_{t-SDH}(g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^t}) = \langle c, g^{\frac{1}{\alpha+c}} \rangle] = \varepsilon$, for any value of $c \in \mathbb{Z}_p \backslash \{-\alpha\}$.
- t-BSDH Assumption [26]: Let $\alpha \xleftarrow{\$} \mathbb{Z}_p^*$, given a $(t+1)-$tuple $\langle g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^t} \rangle \in \mathbb{G}^{t+1}$ as input, for every adversary $\mathcal{A}_{t-BSDH}$,
  $Pr[\mathcal{A}_{t-BSDH}(g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^t}) = \langle c, e(g,g)^{\frac{1}{\alpha+c}} \rangle] = \varepsilon$, for any value of $c \in \mathbb{Z}_p \backslash \{-\alpha\}$.

3.2. Signature of Knowledge

The signer can use the signature of knowledge (SKSIG) to prove he owns a secret without leaking that secret. It is a kind of non-interactive zero-knowledge prove system, which has three typical constructions: (1) signature of knowledge of discrete logarithms (SKLOG). (2) signature of knowledge of double discrete logarithms (SKLOGLOG). (3) signature of knowledge of an $e-th$ root of the discrete logarithms (SKROOTLOG). No adversary can recover the secret or create an illegal signature by the chosen message attack. More detail is in [16,37].

- SKLOG: SKLOG of element $y \in \mathbb{G}_n$ to the base g on message m is a pair $(c, s) \in \{0,1\}^k \times \mathbb{Z}_n^*$ satisfying $c = H(m||y||g||g^s y^c)$. $H(\cdot)$ is a one-way hash function. SKLOG is denoted $SKLOG[\alpha : y = g^\alpha](m)$, where α is the target of zero-knowledge proof, it should be secret to verifier. For any adversary $\mathcal{A}_{SKLOG}$, $Pr[\mathcal{A}_{SKLOG}(c, s) = \{\alpha \vee (c', s')\}] = \varepsilon$, where $(c, s) \neq (c', s')$.
- SKLOGLOG: SKLOGLOG is denoted $SKLOGLOG[\beta : y = g^{\alpha^\beta}](m)$, where β should be kept secret to the verifier, other references are public.
- SKROOTLOG: SKROOTLOG is denoted $SKROOTLOG[\beta : y = g^{\beta^e}](m)$, where β should be kept secret to the verifier, other references are public.

3.3. Strong RSA Assumption

Let p, q be two big prime integers, compute $n = pq$. Given tuple (n, e), for every adversary $\mathcal{A}_{RSA}$, $Pr[\mathcal{A}_{RSA}(n, e) = (z, d)] = \varepsilon$ such that $z^d = e \bmod n$ [38].

4. Revocable Publicly Verifiable Computation (RPVC) Model

This section first introduces the RPVC model, then provides its threat model, and finally puts forward the design goals.

4.1. RPVC Model

As shown in Figure 2, there are four entities in the RPVC model: cloud server, edge server, auditor, and user. The reason why the function owner does not become an RPVC entity is that the edge server downloaded the computing function from a cloud server. The process that the function owner entrusts the computing function to the cloud can be initialized offline Edge servers and the auditor are in the same group, the edge server has the role of a group member, and the auditor has the role of the group manager. RPVC entities are described as follows:

- Cloud Server: The cloud server receives functions initialized by different function owners and allows legal edge servers to download functions.
- Edge Server: The edge server sends a request to the auditor for joining edge computing. After the auditor approves, the edge server downloads functions from the cloud server and performs computing for users.
- Auditor: The auditor is responsible for approving the edge server's join request and revoking a dishonest edge server who provided incorrect results.
- User: The user verifies the results returned by the edge server. If the result fails to pass the verification, the user will send a revoke request to the auditor.

As shown in Figure 2, an RPVC can be divided into three phases: an initialize phase, a join phase and an outsource computing phase:

- Initialize Phase: As step ① in Figure 2, this phase can be performed offline by the function owner. The function owner selects the function private key α, and sends the computing function F and function evaluation key EK to the cloud server. Then, the function owner sends the verification key VK to users.
- Join Phase: The join phase includes step ② – ⑤ in Figure 2. The edge server applies to the auditor for joining computation in step ② . After the auditor validates the edge server's application, the auditor sends a group member certificate $Cert$ or its mask σ_{Cert} to the edge server by a secure channel in step ③ . In step ④ , the edge server downloads the computing function F and evaluation key EK from the cloud server. In step ⑤ , the auditor broadcasts data structure T which stores the currently valid edge server and group public key Gpk to users.
- Outsource Computing Phase: The outsource computing phase includes step ⑥ – ⑨ in Figure 2. In step ⑥ , the user sends function input x to the edge server. In step ⑦ , the edge server evaluates function with input x. Then, return result $y = F(x)$, corresponding proof $proof$ and computation signature $SKSIG_{comp}$ to the user. In step

⑧ , the user verifies $SKSIG_{comp}$ to confirm the result is returned by a legal edge server, next, verify y is correct with $proof$. In step ⑨ , If the result verification doesn't pass, the user sends a revoke request to the auditor. The auditor traces and revokes a dishonest edge server and reorganize T which gets rid of the information of the dishonest edge server.

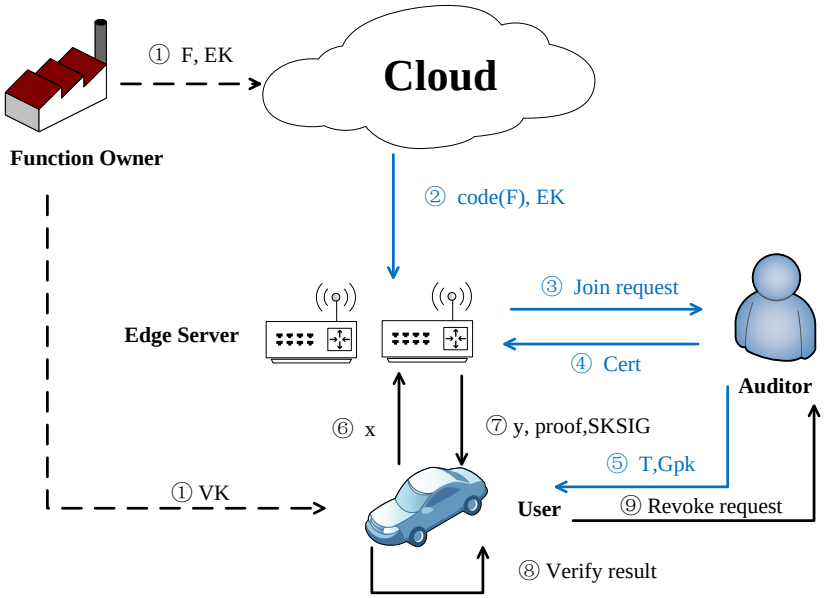

Figure 2. Model Architecture (Dotted lines stand for the initialize phase. Blue lines and black solid lines denote the join phase and outsource computing phase).

A revocable publicly verifiable computation scheme RPVC consist of five algorithms (Setup, Register, Compute, Verify, Revoke) as follows:

Setup$(1^{\lambda}, \alpha, F) \longrightarrow (EK, VK, Gsk, Gpk, T, L)$: The algorithm is used for model setup, which includes cloud server setup and auditor setup.

(1) Cloud server setup: In order to make the outsourcing computation results verifiable and so they cannot be forged, the function owner privately selects a random number α, and generates a function evaluation key EK and a function verification key VK according to the security parameter λ.

(2) Auditor setup: The auditor generates a group private key Gsk and a group public key Gpk by security parameter λ, that Gsk used for issuing a group member certificate and Gpk used for verifying the validity of group signature for results and its proof. The auditor creates a binary tree T which can quickly search all valid edge servers, and record the identities of edge servers in list L privately.

Register$(\sigma_{id}, mem(\sigma_{id}), SKSIG_{Reg}, T) \longrightarrow (\sigma_{Cert}, T)$: The auditor executes this algorithm. The auditor will receive a request from an edge server that wants to join the outsource computation. That request should use the mask of the identity private key σ_{id} to prevent the real identity id of the edge server from being exposed. The edge server should use Gpk to generate $mem(\sigma_{id})$ which is the mask code of group membership for the auditor to trace the signer. To prevent man-in-the-middle attacks, the edge server should use knowledge signature $SKSIG_{Reg}$ to prove he knows the identity private key id without leaking it. After the auditor verified $SKSIG_{Reg}$, the edge server will receive the mask of the group member certificate σ_{Cert}. Then, the auditor adds the edge server to T and L. The edge server gets the group member certificate $Cert$ by decoding σ_{Cert}.

Compute$(x, F, EK, SK_S, Cert) \longrightarrow (y, proof, SKSIG_{comp})$: The edge server executes this algorithm. The edge server evaluates the function value y by the user input x, then, uses EK to compute the corresponding proof $proof$ of y. Finally, the edge server makes a revocable group signature $SKSIG_{comp}$ to y and $proof$, with $Cert$ and a signature private key set SK_S. $SKSIG_{comp}$ can prove the identity of the edge server through non-interactive zero-knowledge proof, at the same time, it can ensure the edge server is not revoked by the auditor.
Verify$(T, Gpk, VK, y, proof, SKSIG_{Comp}) \longrightarrow \tau_y$: The user executes this algorithm. The user first verifies $SKSIG_{comp}$ by T and Gpk to ensure the result is from a legal edge server that has not been revoked. Next, the user verifies y is correct by VK and $proof$. Finally, if these two verifications are passed, the user outputs accept token $\tau_y = true$, otherwise, $\tau_y = false$.
Revoke$(T, L, \tau_y, SKSIG_{Comp}) \longrightarrow T$: The auditor executes this algorithm. The auditor opens $SKSIG_{comp}$ with L to trace the identity of the dishonest edge server under the condition of $\tau_y = false$, then removes it from T and L. From then on, the result returned by a dishonest edge server will never pass the verification.

4.2. Threat Model

For users, the auditor is trusted and other entities are semi-trusted. In other words, edge servers and cloud servers may tamper with or forge content. Based on the information available to adversaries, we consider the following two threat models:

(1) Chosen Plaintext Attack Model: In this model, the attacker may obtain encryptions of his chosen messages, such as the mask code of id, VK of the function F, or the proof for computing result y.
(2) Chosen Message Attack Model: In this model, the attacker may obtain signatures of his chosen messages, such as additional information which would be used to construct an existential universal forgery group signature.

4.3. Design Goals

To achieve RPVC in edge computing, we aim to achieve the following design goals.

(1) Function Binding: The VK and EK should only be used to verify or compute the specific function F which the function owner provided. The function binding experiment $EXP^{fb}_{RPVC}(\mathcal{A})$ is shown in Figure 3, the RPVC is *Function Binding* if $Adv^{fb}_{RPVC}(\mathcal{A})$ is negligible for any adversary $\mathcal{A}$.
(2) Result Reliability: For the user's specific input x, the edge server should not give valid results and proofs other than the real function value y. The result reliability experiment $EXP^{rr}_{RPVC}(\mathcal{A})$ is shown in Figure 3, the RPVC is *Result Reliability* if $Adv^{rr}_{RPVC}(\mathcal{A})$ is negligible for any adversary $\mathcal{A}$.
(3) Anonymity: The user should not recover the identity id of any edge server. The anonymity experiment $EXP^{ano}_{RPVC}(\mathcal{A})$ is shown in Figure 4, the RPVC is *Anonymity* if $Adv^{ano}_{RPVC}(\mathcal{A})$ is negligible for any adversary $\mathcal{A}$.
(4) Revocability: The user should not accept the results which are returned by revoked edge servers. In addition, the adversary should not show the valid signature associated with wrong results or proofs The revocability experiment $EXP^{rev}_{RPVC}(\mathcal{A})$ is shown in Figure 4, the RPVC is *Revocability* if $Adv^{rev}_{RPVC}(\mathcal{A})$ is negligible for any adversary $\mathcal{A}$.

$EXP^{fb}_{RPVC}(\mathcal{A})$ **- Function Binding Game**	$EXP^{rr}_{RPVC}(\mathcal{A})$ **- Result Reliability Game**
1. $(EK, VK) \xleftarrow{\$}$Update$(1^\lambda, \alpha, F)$ 2. $F' \longleftarrow \mathcal{A}(EK, VK, F)$ 3. If Verify$(VK, EK, F') = 1$ and $F \neq F'$ return 1 Else, return 0	1. $(EK, VK) \xleftarrow{\$}$Update$(1^\lambda, \alpha, F)$ 2. $(y, proof) \longleftarrow$Compute$(x, F, EK)$ 3. $(y^*, proof^*) \longleftarrow \mathcal{A}(x, F, EK, y, proof)$ 4. If Verify$(VK, y^*, proof^*) = 1$ and $y \neq y^*$ return 1 Else, return 0

Figure 3. Security Games for RPVC.

$EXP^{ano}_{RPVC}(\mathcal{A})$ **- Anonymity Game**	$EXP^{rev}_{RPVC}(\mathcal{A})$ **- Revocablity Game**
1. $(Gpk, T) \xleftarrow{\$}$Setup$(1^\lambda)$ 2. $(\sigma_{Cert}, T) \xleftarrow{\$}$Register$(\sigma_{id}, mem(\sigma_{id}))$ 3. $m \leftarrow \mathcal{A}(Gpk, T)$ 4. $r \xleftarrow{\$} \mathbb{Z}^*_n$ 5. $SKSIG_{Comp} \leftarrow$Compute$(Gpk, id, \sigma_{Cert}, r)$ 6. $id^* \leftarrow \mathcal{A}(Gpk, T, m, SKSIG_{Comp})$ 7. If $id^* = id$, return 1 Else, return 0	1. $(Gpk, T) \xleftarrow{\$}$Setup$(1^\lambda)$ 2. $T \leftarrow$Revoke$(\tau_y, SKSIG_{Comp}, T)$ 3. $m \leftarrow \mathcal{A}(Gpk, T)$ 4. $r \xleftarrow{\$} \mathcal{A}(\mathbb{Z}^*_n)$ 5. $SKSIG^*_{Comp} \xleftarrow{\$} \mathcal{A}(Gpk, id, \sigma_{Cert}, r, m, T)$ 6. If Verify$(VK, SKSIG^*_{Comp}) = 1$ return 1 Else, return 0

Figure 4. Security Games for RPVC.

5. Proposed RPVC Scheme

We now give the detailed construction of each algorithm in RPVC. Notations used in RPVC are in Table 1.

Table 1. Notations.

Symbol	Definition	Symbol	Definition
Gsk	Group private key	Gpk	Group public key
T	Quick access binary tree	L	Edge server list
σ_{id}	Mask code of user's id	$mem(\sigma_{id})$	Mask code of group membership
λ	Security parameter	α	Private random number choosen by function owner
F	Function		
VK	Verify Key	EK	Computing Key
$Cert$	Certificate	σ_{Cert}	Mask code of Certificate
y	Computing result	SK_S	Signature private key set
τ_y	The token decides whether accept y	$proof$	Verifiable proof of y
$SKSIG$	Signature of knowledge for joining the group or computing	x	The value which user outsource.

5.1. Setup

The setup algorithm of the RPVC scheme including cloud server setup and auditor setup.

(1) Cloud server setup: The function owner owns a polynomial form function $F = \Phi(x) \in \mathbb{Z}_p[x]$ with degree $deg(\Phi) \leq t$. $\Phi(x)$ which can be expressed as Equation (1)

$$\Phi(x) = \sum_{j=0}^{deg(\Phi)} \phi_j x^j \tag{1}$$

Step 1. The function owner chooses two groups $\mathbb{G}$ and $\mathbb{G}_T$ with prime order p, two groups can make bilinear maping $e : \mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$ satisfies the $t - SDH$

assumption. $\mathcal{G} =< e, \mathbb{G}, \mathbb{G}_T >$ is defined as a bilinear group with generator $g_2 \xleftarrow{R} \mathbb{G}$.

Step 2. The function owner privately chooses $\alpha \xleftarrow{R} \mathbb{Z}_p^*$, then, computes function evaluation key $EK = \{\mathbb{G}, g_2, g_2^{\alpha}, \ldots, g_2^{\alpha^t}\}$ and function verification key in Equation (2) according to α.

$$VK = \prod_{j=0}^{deg(\Phi)} \left(g_2^{\alpha^j}\right)^{\phi_j} \tag{2}$$

Step 3. The function owner sends F and EK to the cloud server, sends VK to users. This step can be completed offline, such as VK can be embedded in vehicle OBU by car manufacturers in IoV applications.

(2) Auditor setup: The auditor outputs group private key Gsk and group public tree Gpk with security parameters a, λ, then generates a subset covering complete tree (SCST) according to valid edge servers at time t, at last, updates list L.

Step 1. The auditor first privately chooses two big primes at random, and gets their product n_c as RSA modulus. Then, it generates group $\mathbb{G}_1$ in order n_c with generator $g_1 \leftarrow \mathbb{G}_1$. Next, it selects security parameters a and λ for knowledge signature. After that, it choses $e_c \xleftarrow{R} \mathbb{Z}_n$ and computes d_c which satisfies Equation (3). Finally, the auditor keeps a group private key $Gsk = (n_c, d_c)$, and broadcasts a group public key $Gpk = \{n_c, e_c, g_1.\mathbb{G}_1, a, \lambda\}$.

$$e_c d_c \equiv 1 \mod \varphi(n_c) \tag{3}$$

Step 2. Let N be the overall set of edge servers, R is the set of revoked edge servers, clearly, $N \backslash R$ is the set of valid edge servers right now. The auditor builds a minimum complete binary tree CT with a height of $l = \lceil log|N| \rceil$, at the same time, it initializes all leaf nodes to $\perp$. The root node of CT is recorded as $x_{0,0}$, other nodes can be expressed as $x_{i,j}$, where $i \in [0, \ldots, l]$, $j \in [1, \ldots, 2^i]$. According to the subset covering theory, the parent node can be used to represent the set composed of its two child nodes under the condition of both child nodes belonging to $N \backslash R$. If iterate to the root node in this way, $N \backslash R$ can be expressed by the set of parent nodes, these parent nodes represent num disjoint subsets $S_1 \cup S_2 \cup \cdots \cup S_{num}$, in which num is the minimum amount of disjoint subsets in the current valid leaf node arrangement. The resulting $SCST$ is the set of nodes from the above processes. Algorithm 1 shows how $SCST$ is generated.

Step 3. The auditor should assign random $e_{i,j}$ to each node $x_{i,j}$ on $SCST$, and calculate $d_{i,j}$ which satisfies Equation (4) and then attach a timestamp t to $SCST$. At last, it should put the edge servers' information into L.

$$e_{i,j} d_{i,j} \equiv 1 \mod \varphi(n_c) \tag{4}$$

Algorithm 1 SCST Generator.

Input: All signers set N, revoked set R

Build complete binary tree CT with N
$tmp =<>$
for all x in CT **do**
  if x in R **then**
    $tmp.add(path(x))$
  end if
end for
for all x in tmp **do**
  if x_{left} in tmp **then** add x_{left} to $SCST$
  else add x_{right} to $SCST$
  end if
end for
return $SCST$

5.2. Register

The auditor issues a group membership certificate to the edge server and broadcasts the latest $SCST$ to users.

Step 1. The edge server privately selects an identifier $id \xleftarrow{R} \mathbb{Z}_p^*$, then, computes Equation (5) as the mask of id and Equation (6) as the mask of group membership. Next, it makes the knowledge signature $SKSIG_{Reg}$ to σ_{id} and $mem(\sigma_{id})$ by Equation (7), and sends a request for joining the edge computing to the auditor. The request should involve σ_{id}, $mem(\sigma_{id})$ and $SKSIG_{Reg}$.

$$\sigma_{id} = a^{id} \mod n_c \tag{5}$$

$$mem(\sigma_{id}) = g_1^{\sigma_{id}} \tag{6}$$

$$\begin{cases} m = (\sigma_{id} \parallel mem(\sigma_{id})) \\ SKSIG_{Reg} = SKLOGLOG[id : mem(\sigma_{id}) = g_1^{\sigma_{id}}](m). \end{cases} \tag{7}$$

Step 2. If the auditor verifies $SKSIG_{Reg}$ successfully, it indicates that the edge server owns id. Based on this premise, auditor selects free leaf node $x_{l,k}(k \leq 2^i)$ on $SCST$, chooses random $e_{l,k}$ and $d_{l,k}$ which satisfies Equation (8). Obviously, the group member certificate for the edge server is Equation (9). The auditor puts the information of the edge server into L, the form of the record is $\{\sigma_{id}, mem(\sigma_{id}), d_{l,k}\}$, after that, it updates $SCST$ by Algorithm 1.

$$e_{l,k}d_{l,k} \equiv 1 \mod \varphi(n_c) \tag{8}$$

$$\sigma_{Cert} = (\sigma_{id} + e_{l,k})^{d_c} \mod n_c \tag{9}$$

Step 3. The auditor broadcasts the latest $SCST$ to users, transmits σ_{Cert} and $Cred_{id} = g_1^{d_{l,k}}$ which is used for proving the identity of the edge server is valid at time t to the edge server.

Step 4. The edge server creates a signature private key set $SK_S = \{id, \sigma_{Cert}, Cred_{id}\}$.

5.3. Compute

The edge server evaluates the function value and its proof for users, then, makes the group signature revocable on result and proof.

Step 1. The user sends function input x to the edge server.

Step 2. The edge server evaluates the function value $y = F(x)$, and calculates the corresponding $proof$ by EK with Equation (10).

$$proof = g_2^{\psi_x(\alpha)}, \ \ \psi_x(\alpha) = \frac{F(\alpha) - F(x)}{\alpha - x} \in \mathbb{Z}_p[x] \tag{10}$$

Step 3. In order to show the validity and that it has not been revoked, the edge server should make the signature q to y and $proof$ by Equation (11), where $h(\cdot)$ is a one-way hash function.

$$q = \left(g_1^{d_{l,k}}\right)^{h(y||proof)} \ mod \ n_c \tag{11}$$

Furthermore, the edge server makes a group signature by Equation (12) as in [16,37]. The final computation signature is $SKSIG_{Comp} = \{q, \tilde{g}, \tilde{z}, V_1, V_2\}$.

$$\begin{cases} \tilde{g} = g_1^r, \ r \xleftarrow{R} \mathbb{Z}_p^*, \ \tilde{z} = \tilde{z}^{\sigma_{id}} \\ V_1 = SKLOGLOG[id : \tilde{z} = \tilde{g}^{a^{id}}](y \parallel proof) \\ V_2 = SKROOTLOG[Cert : \tilde{z}\tilde{g}^{e_{l,k}} = \tilde{g}^{Cert^{e_c}}] \end{cases} \tag{12}$$

Step 4. The edge server returns $\{e_{l,k}, y, proof, SKSIG_{Comp}\}$ to user.

5.4. Verify

The user first verifies whether the result is from a valid legal edge server, then, verifies that the result is correct. If a fault occurs during the verification process, the user will send a revoke request to the auditor.

Step 1. The user synchronizes the $SCST$ from the auditor and confirms $e_{l,k} \in SCST$ by Equation (13).

$$g_1^{h(y||proof)} = q^{e_{l,k}} \ mod \ n_c \tag{13}$$

Step 2. The user rapid verifies V_1, V_2 in $SKSIG_{Comp}$ by the hash functions provided by $SKLOGLOG$ and $SKROOTLOG$.

Step 3. The user verifies the result is correct by Equation (14), and outputs an accept token $\tau_y = accept$ after the result hass passed verification. Otherwise, the user sends a revoke request to the auditor; the request contains a reject token $\tau_y = reject$ and $SKSIG_{Comp}$.

$$e(VK, g_2) = e(proof, \ g_2^{\alpha}/g_2^{x}) \cdot e(g_2, g_2)^{F(x)} \tag{14}$$

5.5. Revoke

The auditor opens the revocable group signature to lock the identity of the dishonest edge server, removes it from L, and broadcasts the updated $SCST$ which deletes the leaf node of the dishonest edge server to users.

Step 1. After the auditor received the revoke request, the auditor opened $SKSIG_{Comp}$ to trace the dishonest edge server with the help of L, and then deletes it from L.

Step 2. The auditor updates $SCST$ by means of deleting the leaf node corresponding to the dishonest edge server via Algorithm 1, then broadcasts the latest $SCST$. For example, as shown in Figure 5, there are eight signers $x_{3,1}, \ldots, x_{3,8}$. When the auditor receives the request to revoke $x_{3,2}, x_{3,5}, x_{3,6}$ at time t, $SCST$ will be updated to $\{t||x_{3,1}, x_{2,2}, x_{2,4}\}$.

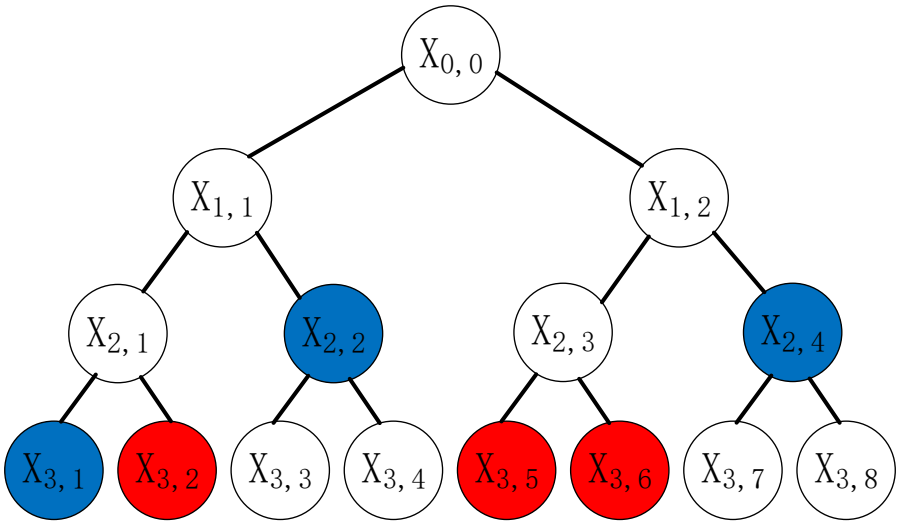

Figure 5. Example of SCST generation process. (The nodes to be revoked are represented in red, and the new subset covering the node is represented in blue).

6. Scheme Analysis

This section first illustrates the correctness of RPVC from the correctness of the result and signature. Then, combined with the security model proved the security of RPVC from the aspects of function binding, result reliability, anonymity, and revocability.

6.1. Correctness

6.1.1. Correctness of Result

The edge computing results and its verification processes are correct, because:

$$\begin{aligned} & e(proof, g_2^{\alpha}/g_2^{x}) \cdot e(g_2, g_2)^{F(x)} \\ &= e\left(g_2^{\psi_x(\alpha)}, g_2^{\alpha - x}\right) \cdot e(g_2, g_2)^{F(x)} \\ &= e(g_2, g_2)^{\psi_x(\alpha)(\alpha - x) + F(x)} \\ &= e(g_2, g_2)^{\frac{F(\alpha) - F(x)}{(\alpha - x)}(\alpha - x) + F(x)} \\ &= e(g_2, g_2)^{F(\alpha)} = e(VK, g_2) \end{aligned}$$

6.1.2. Correctness of Signature

The signature q used by the edge server to prove it has not been revoked and the verification process of q is correct, because:

$$\begin{aligned} & q^{e_{l,k}} \ mod \ n_c \\ &= g_1^{d_{l,k} h(y||proof) e_{l,k}} \ mod \ n_c \\ &= g_1^{h(y||proof)} \end{aligned}$$

6.2. Security Analysis

The proof method of RPVC uses "game-playing" technology which was proposed in [39–41]. This technology uses the game sequence specification to prove that the possibility of the adversary winning the game is negligible, and the probability of the adversary winning two adjacent games should be controlled within a negligible range (i.e., indistinguishable in polynomial time). Define the probability of **Game *i*** happens is $Pr(S_i)$.

6.2.1. Proof of Function Binding

Game 0: This is the original function binding game in Figure 3, $\mathcal{A}$ trying to find another F' which has the same VK with F. $\mathcal{A}$ obtains EK, VK and F as information. Clearly, $Adv^{fb}_{RPVC}(\mathcal{A}) = Pr(S_0)$.
Game 1: This game is the same as Game 0 except that replace F' with $F + \hat{F}$. If $\mathcal{A}$ can find F', he must find a different function $\hat{F}$. So, $Pr(S_1) = Pr(S_0)$.
Game 2: This game is the same as Game 1 except that replace the winning condition to $\hat{\Phi}(\alpha) = 0 \wedge \hat{\phi}_j$ not all 0. $\hat{\Phi}(\alpha) = 0$ since

$$\widehat{VK} = g_2^{\hat{\Phi}(\alpha)} = g_2^{\Phi'(\alpha)-\Phi(\alpha)} = g_2^{\Phi'(\alpha)} / g_2^{\Phi(\alpha)} = 1.$$

Because $F' \neq F$, the polynomial coefficients $\hat{\phi}_j$ cannot be all 0. Clearly, $Pr(S_2) = Pr(S_1)$.
Claim 1:

$$Pr(S_2) \leq Adv^{DL}_{g,g^a}(\mathcal{B}^{\mathcal{A}})$$

Let n be the degree of polynomial, algorithm $\mathcal{B}$ is a tool that assumes $\mathcal{A}$ can solve a class of problems including the games. If the simplest case in these problems is the current difficult mathematical problem, it means that the adversary cannot break the security characteristic. $\mathcal{B}^{\mathcal{A}}(n, F, F')$ computing a collision as in the following steps:

(1) $\alpha' \leftarrow \mathcal{A}(n, \Phi(\alpha), \Phi'(\alpha)), 2 \leq n \leq t$
(2) If $\alpha = \alpha'$ return 1
Else return 0

Proof. *Game 2* is equivalent to $\mathcal{B}^{\mathcal{A}}(t, F, F')$. It will be the simplest polynomial of degree one problem when $n = 2$, so $Adv^{fb}_{n=t}(\mathcal{B}^{\mathcal{A}}) \leq Adv^{fb}_{n=2}(\mathcal{B}^{\mathcal{A}})$. That means to solve the latter is easier than the former, $\mathcal{A}$ will own more advantages. Now let $n = 2$, the processes to find a collision are as follows:

$$g^{(\phi_1-\phi_1')\alpha'+\phi_0-\phi_0'} = 1$$
$$\Leftrightarrow (\phi_1 - \phi_1')\alpha' + \phi_0 - \phi_0' = 0$$
$$\Leftrightarrow \alpha' = \frac{\phi_0' - \phi_0}{(\phi_1 - \phi_1')}$$

Clearly, find α' via g_2 and g_2^{α} on cyclic group $\mathbb{G}$ is a more efficient way, which exactly is DL assumption. Hence,

$$Adv^{fb}_{RPVC}(\mathcal{A}) \leq Adv^{fb}_{n=2}\left(\mathcal{B}^{\mathcal{A}}\right) \leq Adv^{DL}_{g_2,g_2^{\alpha}}\left(\mathcal{B}^{\mathcal{A}}\right) \leq \varepsilon$$

That is, the probability of the adversary successfully attacking is negligible, the RPVC scheme achieves *Function Binding*. □

6.2.2. Proof of Result Reliability

Game 0: This is an original result reliability game in Figure 3, $\mathcal{A}$ trying to find malicious y^* and $proof^*$ that can pass the user verification. Clearly, $Adv^{rr}_{RPVC}(\mathcal{A}) = Pr(S_0)$.
Game 1: The adversary can compute $\frac{1}{\alpha - x}$. The reason why *Game* 0 $\approx_p$ *Game* 1 and $Pr(S_1) = Pr(S_0)$ is:

$$Verify(VK,\ y^*, proof^*) = Verify(VK,\ y, proof)$$
$$\Leftrightarrow e(g_2, g_2)^{\psi_x(\alpha)(\alpha-x)} \cdot e(g_2, g_2)^{y} = e(g_2, g_2)^{\psi_x^*(\alpha)(\alpha-x)} \cdot e(g_2, g_2)^{y^*}$$
$$\Leftrightarrow \psi_x(\alpha)(\alpha - x) + y = \psi_x^*(\alpha)(\alpha - x) + y^*$$
$$\Leftrightarrow \frac{\psi_x(\alpha) - \psi_x^*(\alpha)}{y^* - y} = \frac{1}{\alpha - x}$$

Game 2: For g_2 is part of Gpk, the adversary can give valid pair $(-x, g_2^{\frac{1}{\alpha-x}})$. Obviously, *Game* 1 $\approx_p$ *Game* 2.
Claim 2:

$$Pr(S_2) \leq Adv_{EK}^{t-SDH}(\mathcal{B}_2^{\mathcal{A}})$$

$\mathcal{B}_2^{\mathcal{A}}(EK, V)$ can give a valid pair $(c, g_2^{\frac{1}{\alpha+c}})$, where V is some kind of valid algorithm, $c \in \mathbb{Z}_n^*$. $\mathcal{B}_2^{\mathcal{A}}$ is a bridge between *Game* 2 and t-SDH difficult mathematical.

Proof. When $EK = \{\mathbb{G}, g_2, g_2^{\alpha}, \ldots, g_2^{\alpha^t}\}$, algotithm $\mathcal{B}_2^{\mathcal{A}}$ is t-SDH assumption. Though the adversary can give a valid algorithm $V(x, y, proof, y^*, proof^*)$, $Adv_{EK,V}^{t-SDH}(\mathcal{B}_2^{\mathcal{A}}) \leq Adv_{EK}^{t-SDH}(\mathcal{B}_2^{\mathcal{A}})$. The output of algorithm V is $(-x, g_2^{\frac{1}{\alpha-x}})$, where

$$g_2^{\frac{1}{\alpha-x}} = (\frac{proof}{proof^*})^{\frac{1}{y^*-y.}}$$

So the following equation holds,

$$Adv_{RPVC}^{rr}(\mathcal{A}) \leq Adv_{EK}^{t-SDH}(\mathcal{B}_2^{\mathcal{A}}) \leq \varepsilon$$

That is, the probability of the adversary succesfully attacking is negligible, the RPVC scheme achieves *Result Reliability*. □

6.2.3. Proof of Anonymity

Game 0: This is an original anonymity game in Figure 4, $\mathcal{A}$ trying to figure out the identity of the edge server. Clearly, $Adv_{RPVC}^{ano}(\mathcal{A}) = Pr(S_0)$.
Game 1: The adversary has the ability to extract *id* from at least one of the following parts: $\widetilde{z}, V_1, V_2$. Explicitly, *Game* 0 $\iff$ *Game* 1 and $Pr(S_1) = Pr(S_0)$ for the identity information in $SKSIG_{Comp}$ only including $\widetilde{z}, V_1, V_2$.
Game 2: This game is the same as Game 1 except that the adversary has the ability to extract *id* from V_1 or V_2. Function $F_1(y, x)$ denotes the probability of extract x from $y = g^x$ under the DL assumption. Extract *id* from $\widetilde{z}$ should sequentially execute: $F_1(\widetilde{z}, r\sigma_{id})$, $F_1(\widetilde{g}, r)$ and $F_1(\sigma_{id}, id)$. The recursive proof method can refer to the literature [41]. The above process can be expressed as:

$$|Pr(S_1) - Pr(S_2)| \leq Pr(extract \quad id \quad from \quad \widetilde{z}),$$

and it is not more than

$$Pr(F_1(\widetilde{z}, r\sigma_{id}))Pr(F_1(\widetilde{g}, r))Pr(F_1(\sigma_{id}, id)) \leq (Adv_{g,g^a}^{DL}(\cdot))^3$$

Finally, *Game* 1 $\approx_p$ *Game* 2 and $Pr(S_2) = Pr(S_1)$, due to

$$(Adv_{g,g^a}^{DL}(\cdot))^3 \leq \varepsilon^3 \leq \varepsilon$$

Game 3: This game is the same as Game 2 except that the adversary can extract *id* from V_1. Function $F_2(y, x)$ and $F_3(y, x)$ denotes the probability of extract x from y under the RSA assumption and SKROOTLOG signature. Extract *id* from V_2 should sequentially execute: $F_3(V_2, \sigma_{cert})$, $F_2(\sigma_{cert}, \sigma_{id})$ and $F_1(\sigma_{id}, id)$. Similiar to Game 2, it can infer that

$$|Pr(S_3) - Pr(S_2)| \leq Pr(Event_3) \leq Adv_{g,g^a}^{DL}(\cdot)Adv_{n,e}^{RSA}(\cdot)Adv_{SKSIG}^{SKROOTLOG}(\cdot) \leq \varepsilon,$$

so $Game\ 3 \approx_p Game\ 2$ and $Pr(S_2) = Pr(S_3)$. The RPVC scheme achieves *Anonymity* for

$$Pr(S_3) \leq Adv_{SKSIG}^{SKLOGLOG}(\cdot) \leq \varepsilon$$

6.2.4. Proof of Revocability

Game 0: This is an original revocable game in Figure 4, a revoked $\mathcal{A}$ trying to succesfully sign results or faking an honest user's signature on wrong results. Clearly, $Adv_{RPVC}^{rev}(\mathcal{A}) = Pr(S_0)$.
Game 1: This game is the same as Game 0, the adversary can recovery corresponding $d_{l,k}$ by $e_{l,k}$ or replace valid q in malicious $SKSIG_{Comp}$. $Game\ 0 \Leftrightarrow Game\ 1$.
Game 2: This game is the same as Game 1 except that the adversary can recover corresponding $d_{l,k}$ by $e_{l,k}$. If an adversary can replace valid q in malicious $SKSIG_{Comp}$, he must make sure V_2 can be verified succesfully So

$$|Pr(S_1) - Pr(S_2)| \leq Adv_{SKSIG}^{SKROOTLOG}(\cdot) \leq \varepsilon,$$

further, $Game\ 1 \approx_p Game\ 2$. The RPVC scheme achieves *Revocability* for

$$Pr(S_2) \leq Adv_{n,e}^{RSA}(\cdot) \leq \varepsilon$$

7. Performance Analysis

In Table 2, we compare other existing group signature schemes in the IoV scenario with RPVC. The results in Table 2 show that our scheme uses a superior audit method to find the dishonest participants, and the core cryptographic algorithm is the non-interactive zero-knowledge signature, which is mainly based on a hash function that is more efficient than existing schemes. Besides, RPVC updates the SCST at a regular time, which provides participants with more fault tolerance.

Table 2. Comparison of RPVC with existing schemes.

Scheme	Auditable	Audit Method	Update Frequency	Key Generator	Cryptographic Algorithm
[42]	No	Cannot	Dynamic	Single	Symmetric
[43]	No	Cannot	Dynamic	Multi-party	Symmetric
[44]	Yes	Iterate list	Dynamic	Multi-party	ECC
[45]	Yes	Direct	Never	Single	BBS
[46]	Yes	Iterate list	Dynamic	Single	ECC
RPVC	Yes	Query the tree	Timed	Single	Zero-knowledge signature

In order to compare the performance of the RPVC more intuitively, we conduct a series of experiments to evaluate the cost and efficiency of the RPVC. The experimental environment is deployed on a PC with Ubuntu 20.0.4 TLS, bilinear pairing rely on bn256 (github.com/ethereum/go-ethereum/crypto/bn256/cloudflare, accessed on 20 March 2022), other libraries including PBC 0.5.14 (https://crypto.stanford.edu/pbc/, accessed on 15 March 2022) and GMP-6.2.1 (https://gmplib.org/, accessed on 15 March 2022).

Some basic assumptions in the experiments are as follows: The service radius of RSU is about 2.5 km [47], users' vehicle speed is not more than 180 km/h. 3G network speed is about 300 KB/s, 4G network speed is about 2.4 MB/s [48]. The reaction time of a driving human to brake is 600–1400 ms [49].

The test contains two parts: One is the process of the edge server applying to join edge computing and the auditor revokes a dishonest edge server, the other one is the user asking for outsourcing computing and receiving verifiable reliable results. The former has three test items: (1) The execution time for the auditor. (2) The size of $SCST$ which the user received from the auditor. (3) The storage space consumed by the user. The latter also has three test items: (4) The extra cost for the edge server to apply RPVC. (5) The time consumed for user verification. (6) The total time delay after applying RPVC.

For test item 1, the execution time for the auditor can be divided into the time to add the edge server into the group and the time to generate *SCST*. As shown in Table 3, the time to add an edge server into the group is about 27.996 ms, which is independent of the scale of the edge server. As shown in Table 4, with edge servers scale increase in the group, the time of the auditor adding or removing an edge server increases proportionally. However, even if the number of edge server reaches 2^{15} (RSU service can cover about 321,700 km^2), the auditor can generate *SCST* within 1 ms. The driving distance is only 1.45 m during the user vehicle receives *SCST* at the highest speed, far less than the service radius of RSU. That is, users have enough time to safely synchronize the current valid edge server. For test item 2, as shown in Table 4, the size of *SCST* is independent from the scale of the edge server, *SCST* is only about 5 kB. For test item 3, the local storage space for the user is multiplied with the increase of the edge server scale, which is shown in Table 5. However, even when the number of edge servers comes up to 2^{15}, storage is less than 15 MB.

Table 3. Execution Time of Four Stages of Group Signature (ms).

Register	Signature	Verify	Open
27.996	28.162	29.001	287.466

Table 4. Add/Delete An Edge Server.

Test Content \ Scale	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}
Size of SCST(KB)	5.974	5.340	5.340	5.023	4.709
Execution time(ms)	0.085	0.158	0.270	0.524	0.986

Table 5. Cumulative SCST Size.

Edge Server Scale	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}
Size(MB)	1.213	2.363	4.352	7.703	14.785

In test items 4–6, we set the degree of polynomial and input x as independent variables, the time cost as dependent variable (default is ns, 10^{-9}s). The rule to choose the independent variable x is: randomly select a value from each range, ranges including $[0, 2^4]$, $[2^4, 2^5]$, $[2^5, 2^6]$, $[2^6, 2^7]$, $[2^7, 2^8]$. Results of test item 4 are shown in Figure 6a, the extra cost of applying the RPVC proportionally tothe polynomial degree, the larger the x, the smaller the curve fluctuation. For test item 5, as shown in Figure 6b, the time of user verification fluctuates between 36 ms and 38 ms, which is less affected by independent variables. Figure 6c indicates the total extra time delay brought by the RPVC application. Even if the degree of a polynomial function is up to 100, the total delay is less than 100 ms, which is far less than the driver's reaction time [49].

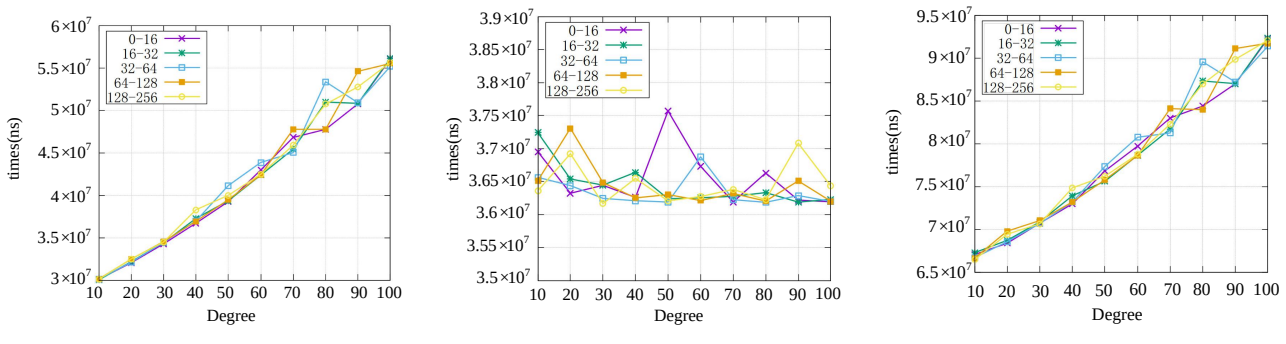

Figure 6. RPVC overhead. (**a**) Edge sever overhead. (**b**) User verify time. (**c**) Total delay.

From the above six test items, it is clear that the RPVC can be used to improve the security of existing edge computing applications. We can summarize the key influencing factor from Figure 7: if the polynomial degree is larger than 40, the performance of the edge server takes the most portion of total time delay, the portion gets larger with the increase of degree. So, a better edge server may expand the application scope of the RPVC.

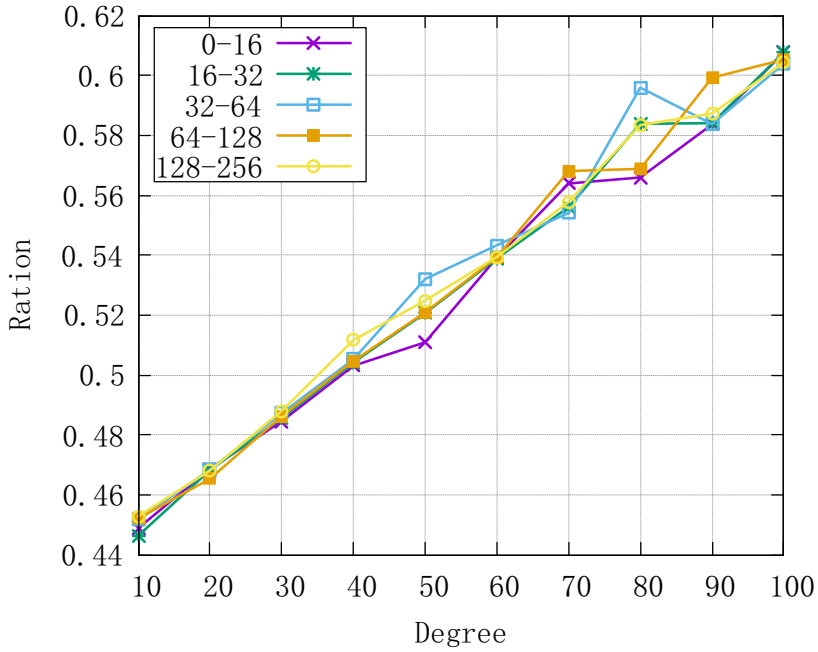

Figure 7. Proportion of edge server delay.

8. Discussion

For the requirements of edge computing in the IoV scenario, the RPVC first achieves the goal of results returned by the edge server being verifiable. At the same time, the identity of the edge server is anonymous to user vehicles and a dishonest edge server can be revoked. From the test results, when a new edge server takes part in outsourced computing, user vehicles do not need to exchange keys with it. The time in which the auditor adds one edge server into the group can be fixed, nearly 28 ms that is independent of the scale of the edge server. The time of user vehicles receiving SCST mainly depends on the communication delay because the generated speed of SCST is less than 1 ms. Though the total delay for user vehicles increases with the degree of the polynomial, it is less than 95 ms when the degree is up to 100 (a very complex computation). Furthermore, the storage overhead is acceptable for user vehicles, even if the number of edge servers comes up to 2^{15}, storage demand is less than 15 MB.

The low delay and overhead are owed to the subset covering complete tree and non-interactive zero-knowledge signature. SCST makes user queries faster than iterating local revoke lists at a small cost. Besides, the non-interactive zero-knowledge signature is mainly based on the hash function, which is more efficient than other large number or exponent multiply schemes. The practical applications of the research can be used to assist the construction of intelligent transportation or vehicle networking.

For future work, we will first reduce the size of SCST for the larger scale of the edge server. Next, machine learning and federated learning can be introduced to improve the performance of edge servers, good solutions can be found in [50–52]. In addition, different regions have different traffic rules and habits, these should be considered. Finally, we will extend the outsource function to varied forms, such as verifiable matrix computation.

9. Conclusions

In this article, we proposed an RPVC model for the edge computing scenario which can be used in IoV applications. The RPVC model cannot only ensure the results returned by edge servers are reliable, but can also revoke dishonest edge servers. The following security proofs show that the RPVC has characteristics of function binding, result reliability, anonymity, and revocability. Experiments show that a new edge server which takes part in edge computing does not need transfer keys to users, and an auditor can approve the request in a fixed time (28 ms). Due to the SCST, users have a low overhead storage and a faster query time, even when the number of edge servers came up to 2^{15}, storage demand is less than 15 MB. Because of the non-interactive zero-knowledge signature, even the degree of outsource function up to 100, the total delay of users is about 95 ms. Thus, applying RPVC to existing IoV applications is acceptable. In the future, we are committed to reducing the size of SCST, trying to introduce machine learning or federated learning to improve the performance of edge servers and supporting verifiable matrix computation.

Author Contributions: Conceptualization, Z.J. and F.Z.; methodology, F.Z.; software, Z.J.; validation, J.S.; formal analysis, Z.J.; investigation, Z.J.; resources, Q.W.; data curation, J.S.; writing—original draft preparation, Z.J.; writing—review and editing, F.Z. and Q.W.; visualization, J.S.; project administration, F.Z. and Q.W.; funding acquisition, F.Z. and Q.W. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Natural Science Foundation of China under Grant 62072090, 62173101, 61902057 and Fundamental Research Funds for the Central Universities under Grant No. N2217009.

Data Availability Statement: The datasets generated for this study are available on request to the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Liu, L.; Wu, B.; Shi, W. A comparison of communication mechanisms in vehicular edge computing. In Proceedings of the 3rd USENIX Workshop on Hot Topics in Edge Computing (HotEdge 20), Santa Clara, CA, USA, 25–26 June 2020.
2. Wu, L.; Zhang, R.; Li, Q.; Ma, C.; Shi, X. A mobile edge computing-based applications execution framework for Internet of Vehicles. *Front. Comput. Sci.* **2022**, *16*, 165506. [CrossRef]
3. Hbaieb, A.; Ayed, S.; Chaari, L. A survey of trust management in the Internet of Vehicles. *Comput. Netw.* **2022**, *203*, 108558. [CrossRef]
4. Lin, H.; Hsieh, M. A dynamic key management and secure data transfer based on m-tree structure with multi-level security framework for Internet of vehicles. *Connect. Sci.* **2022**, *34*, 1089–1118. [CrossRef]
5. Liu, S.; Yan, Z. Verifiable Edge Computing for Indoor Positioning. In Proceedings of the 2020 IEEE International Conference on Communications, ICC 2020, Dublin, Ireland, 7–11 June 2020; pp. 1–6. [CrossRef]
6. Parno, B.; Raykova, M.; Vaikuntanathan, V. How to delegate and verify in public: Verifiable computation from attribute-based encryption. In Proceedings of the Theory of Cryptography Conference, Taormina, Sicily, Italy, 19–21 March 2012; pp. 422–439.
7. Fiore, D.; Gennaro, R.; Pastro, V. Efficiently Verifiable Computation on Encrypted Data. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, 3–7 November 2014; Association for Computing Machinery: New York, NY, USA, 3–7 November 2014; pp. 844–855. [CrossRef]
8. Catalano, D.; Fiore, D.; Warinschi, B. Homomorphic signatures with efficient verification for polynomial functions. In Proceedings of the Annual Cryptology Conference, Santa Barbara, CA, USA, 17–21 August 2014; pp. 371–389.
9. Kate, A.; Zaverucha, G.M.; Goldberg, I. Polynomial commitments. *Tech. Rep.* **2010**. Available online: https://cacr.uwaterloo.ca/techreports/2010/cacr2010-10.pdf (accessed on 1 February 2022).
10. Alderman, J.; Janson, C.; Cid, C.; Crampton, J. Revocation in Publicly Verifiable Outsourced Computation. In Proceedings of the Information Security and Cryptology—10th International Conference, Inscrypt 2014, Beijing, China, 13–15 December 2014; Revised Selected Papers; Lin, D., Yung, M., Zhou, J., Eds.; Springer: Berlin, Germany, 2014; Volume 8957, pp. 51–71. [CrossRef]
11. Attrapadung, N.; Imai, H. Attribute-Based Encryption Supporting Direct/Indirect Revocation Modes. In Proceedings of the Cryptography and Coding, 12th IMA International Conference, Cryptography and Coding, Cirencester, UK, 15–17 December 2009; Parker, M.G., Ed.; Springer: Berlin, Germany, 2009; Volume 5921, pp. 278–300. [CrossRef]
12. Boldyreva, A.; Goyal, V.; Kumar, V. Identity-based encryption with efficient revocation. In Proceedings of the 2008 ACM Conference on Computer and Communications Security—CCS 2008, Alexandria, VA, USA, 27–31 October 2008; Ning, P., Syverson, P.F., Jha, S., Eds.; ACM: New York, NY, USA, 2008; pp. 417–426. [CrossRef]

13. Faust, S.; Hazay, C.; Kretzler, D.; Schlosser, B. Generic Compiler for Publicly Verifiable Covert Multi-Party Computation. In *Proceedings of the Advances in Cryptology—EUROCRYPT 2021—40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, 17–21 October 2021*; Proceedings, Part II; Canteaut, A., Standaert, F., Eds.; Springer: Berlin, Germany, 2021; Volume 12697, pp. 782–811. [CrossRef]
14. Liu, P.; Ma, X.; Zhang, W. Optimizing Fund Allocation for Game-based Verifiable Computation Outsourcing. In Proceedings of the International Conference on Cloud Computing, Virtual Event, 9–10 December 2022; pp. 60–71.
15. Ding, W.; Sun, W.; Yan, Z.; Deng, R.H. An efficient and secure scheme of verifiable computation for Intel SGX. *arXiv* **2021**, arXiv:2106.14253.
16. Camenisch, J.; Stadler, M. Efficient group signature schemes for large groups. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, California, USA, 17–21 August, 1997; pp. 410–424.
17. Song, D.X. Practical forward secure group signature schemes. In Proceedings of the 8th ACM Conference on Computer and Communications Security, Philadelphia, PA, USA, 6–8 November 2001; pp. 225–234.
18. Camenisch, J.; Lysyanskaya, A. Dynamic accumulators and application to efficient revocation of anonymous credentials. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 18–22 August 2002; pp. 61–76.
19. Boneh, D.; Shacham, H. Group signatures with verifier-local revocation. In Proceedings of the 11th ACM Conference on Computer and Communications Security, Washington, DC, USA, 25–29 October 2004; pp. 168–177.
20. Brickell, E.; Camenisch, J.; Chen, L. Direct Anonymous Attestation. In Proceedings of the 11th ACM Conference on Computer and Communications Security, Washington, DC, USA, 25–29 October 2004; Association for Computing Machinery: New York, NY, USA, 2004; pp. 132–145. [CrossRef]
21. Yue, X.; Xi, M.; Chen, B.; Gao, M.; He, Y.; Xu, J. A Revocable Group Signatures Scheme to Provide Privacy-Preserving Authentications. *Mob. Netw. Appl.* **2021**, *26*, 1412–1429. [CrossRef]
22. Yehia, M.; AlTawy, R.; Gulliver, T.A. GM^{MT}: A Revocable Group Merkle Multi-tree Signature Scheme. In Proceedings of the Cryptology and Network Security—20th International Conference, CANS 2021, Vienna, Austria, 13–15 December 2021; Conti, M., Stevens, M., Krenn, S., Eds.; Springer: Berlin, Germany, 2021; Volume 13099, pp. 136–157. [CrossRef]
23. Buser, M.; Liu, J.K.; Steinfeld, R.; Sakzad, A.; Sun, S. DGM: A Dynamic and Revocable Group Merkle Signature. In Proceedings of the Computer Security–ESORICS 2019—24th European Symposium on Research in Computer Security, Luxembourg, 23–27 September 2019; Proceedings, Part I; Sako, K., Schneider, S.A., Ryan, P.Y.A., Eds.; Springer: Berlin, Germany, 2019; Volume 11735, pp. 194–214. [CrossRef]
24. Shamir, A. How to share a secret. *Commun. ACM* **1979**, *22*, 612–613. [CrossRef]
25. Boneh, D.; Waters, B.; Zhandry, M. Low overhead broadcast encryption from multilinear maps. In Proceedings of the Annual Cryptology Conference, Santa Barbara, CA, USA, 17–21 August, 2014; pp. 206–223.
26. Seita, Y.; Nakanishi, T. Speeding up revocable group signature with compact revocation list using vector commitments. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **2019**, *102*, 1676–1687. [CrossRef]
27. Libert, B.; Peters, T.; Yung, M. Group signatures with almost-for-free revocation. In Proceedings of the Annual Cryptology Conference, Santa Barbara, CA, USA, 19–23 August 2012; pp. 571–589.
28. Gennaro, R.; Gentry, C.; Parno, B. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In Proceedings of the Annual Cryptology Conference, Santa Barbara, CA, USA, 15–19 August 2010; pp. 465–482.
29. Benabbas, S.; Gennaro, R.; Vahlis, Y. Verifiable delegation of computation over large datasets. In Proceedings of the Annual Cryptology Conference, Santa Barbara, CA, USA, 14–18 August 2011; pp. 111–131.
30. Backes, M.; Fiore, D.; Reischuk, R.M. Verifiable delegation of computation on outsourced data. In Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, Berlin, Germany, 4–8 November 2013; pp. 863–874.
31. Gentry, C. Fully homomorphic encryption using ideal lattices. In Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing, Bethesda, MD, USA, 31 May–2 June 2009; pp. 169–178.
32. Wang, C.; Ren, K.; Wang, J.; Wang, Q. Harnessing the Cloud for Securely Outsourcing Large-Scale Systems of Linear Equations. *IEEE Trans. Parallel Distrib. Syst.* **2013**, *24*, 1172–1181. [CrossRef]
33. Goldwasser, S.; Gordon, S.D.; Goyal, V.; Jain, A.; Katz, J.; Liu, F.H.; Sahai, A.; Shi, E.; Zhou, H.S. Multi-input functional encryption. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, 11–15 May 2014; pp. 578–602.
34. López-Alt, A.; Tromer, E.; Vaikuntanathan, V. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing, New York, NY, USA, 19–22 May 2012; pp. 1219–1234.
35. Nakanishi, T.; Fujii, H.; Hira, Y.; Funabiki, N. Revocable group signature schemes with constant costs for signing and verifying. In Proceedings of the International Workshop on Public Key Cryptography, Irvine, CA, USA, 18–20 March 2009; pp. 463–480.
36. Boneh, D.; Boyen, X. Short signatures without random oracles. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, 2–6 May 2004; pp. 56–73.
37. Feige, U.; Fiat, A.; Shamir, A. Zero-knowledge proofs of identity. *J. Cryptol.* **1988**, *1*, 77–94. [CrossRef]
38. Rivest, R.L.; Shamir, A.; Adleman, L. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Commun. ACM* **1978**, *21*, 120–126. [CrossRef]
39. Shoup, V. Sequences of games: a tool for taming complexity in security proofs. *IACR Cryptol. ePrint Arch.* **2004**, *2004*, 332.

40. Maurer, U. Abstract models of computation in cryptography. In Proceedings of the IMA International Conference on Cryptography and Coding, Cirencester, UK, 19–21 December 2005; pp. 1–12.
41. Maurer, U. Constructive cryptography–a new paradigm for security definitions and proofs. In Proceedings of the Joint Workshop on Theory of Security and Applications, Saarbruecken, Germany, 31 March–1 April 2011; pp. 33–56.
42. Liu, L.; Wang, Y.; Zhang, J.; Yang, Q. A Secure and Efficient Group Key Agreement Scheme for VANET. *Sensors* **2019**, *19*, 482. [CrossRef] [PubMed]
43. Paliwal, S.; Chandrakar, A. A Conditional Privacy Preserving Authentication and Multi Party Group Key Establishment Scheme for Real-Time Application in VANETs. *IACR Cryptol. ePrint Arch.* **2019**. Available online: https://eprint.iacr.org/2019/1041.pdf (accessed on 17 March 2022).
44. Zhang, C.; Xue, X.; Feng, L.; Zeng, X.; Ma, J Group-Signature and Group Session Key Combined Safety Message Authentication Protocol for VANETs. *IEEE Access* **2019**, *7*, 178310–178320. [CrossRef]
45. Lim, K.; Liu, W.; Wang, X.; Joung, J. SSKM: Scalable and Secure Key Management Scheme for Group Signature Based Authentication and CRL in VANET. *Electronics* **2019**, *8*, 1330. [CrossRef]
46. Zhang, J.; Zhong, H.; Cui, J.; Tian, M.; Xu, Y.; Liu, L. Edge Computing-Based Privacy-Preserving Authentication Framework and Protocol for 5G-Enabled Vehicular Networks. *IEEE Trans. Veh. Technol.* **2020**, *69*, 7940–7954. [CrossRef]
47. Um, J. Performance Analysis According to RSU Range of VANET-based Communication Vehicle. *Int. J. Sci. Eng. Smart Veh.* **2020**, *4*, 1–6. [CrossRef]
48. Khatouni, A.S.; Mellia, M.; Marsan, M.A.; Alfredsson, S.; Karlsson, J.; Brunstrom, A.; Alay, O.; Lutu, A.; Midoglu, C.; Mancuso, V. Speedtest-like measurements in 3g/4g networks: The monroe experience. In Proceedings of the 2017 29th International Teletraffic Congress (ITC 29), Genoa, Italy, 4–8 September 2017; Volume 1, pp. 169–177.
49. Hugemann, W. Driver Reaction Times in Road Traffic. 2002. Available online: https://www.unfallrekonstruktion.de/pdf/evu_2002_reaction_english.pdf (accessed on 5 March 2022).
50. Zhang, T.; Wang, S.; Li, G.; Liu, F.; Zhu, G.; Wang, R. Accelerating Edge Intelligence via Integrated Sensing and Communication. *arXiv* **2021**, arXiv:2107.09574.
51. Zhang, Z.; Wang, S.; Hong, Y.; Zhou, L.; Hao, Q. Distributed Dynamic Map Fusion via Federated Learning for Intelligent Networked Vehicles. In Proceedings of the IEEE International Conference on Robotics and Automation, ICRA 2021, Xi'an, China, 30 May 30–5 June 2021; pp. 953–959. [CrossRef]
52. Huang, S.; Wang, S.; Wang, R.; Wen, M.; Huang, K. Reconfigurable Intelligent Surface Assisted Mobile Edge Computing With Heterogeneous Learning Tasks. *IEEE Trans. Cogn. Commun. Netw.* **2021**, *7*, 369–382. [CrossRef]