EAI Endorsed Transactions
on Scalable Information Systems
Research Article EAI.EU

Cloud-Blockchain Integrated Privacy-Preserving Distributed Computation Offloading Scheme in the Internet of Vehicles

Hua-Yong Wei[1*]

[1]Department of Urban Rail Transit and Information Engineering, Anhui Communications Vocational and Technical College, Hefei, Anhui 230051, China

Abstract

INTRODUCTION: With the rapid development of the Internet of Vehicles technology, on-board applications in intelligent vehicles are increasingly facing the challenge of processing latency-sensitive and computationally intensive tasks. Although vehicle-mounted computing units possess certain computing capabilities, their limited resources cannot cope with complex latency-sensitive tasks. Offloading computations to resource-rich cloud servers is a feasible solution to this problem. However, transmission costs and communication delays on remote cloud servers become major bottlenecks for offloading latency-sensitive tasks. The emergence of Mobile Edge Computing (MEC) shifts computing power from centralized cloud servers to the network edge, reducing latency. Nevertheless, existing schemes still suffer from poor fault tolerance, privacy leakage, and low computational efficiency.
OBJECTIVES: To address these limitations, this paper proposes a cloud-blockchain integrated privacy-preserving distributed computation offloading scheme (FCOS).
METHODS: FCOS significantly improves fault tolerance via a redundant distributed computing mechanism, effectively mitigating the impact of single-point failures on computations. RESULTS: Smart contracts on the blockchain guarantee verifiability and fairness throughout the computation offloading process, enhancing overall offloading security. For privacy protection, the scheme combines homomorphic encryption and blind factor technology to protect data privacy with low computational overhead on intelligent vehicle terminals.
CONCLUSION: Compared with existing methods, FCOS reduces the average computation time by 90.4% for polynomial calculations within 5000 iterations in the cloud computing phase. Under a single-point failure rate ranging from 0% to 60%, the average computation time for polynomial calculations within 10,000 iterations is reduced by 92.9% to 96.7%.

Keywords: internet of vehicles, secure computation offloading, blockchain, privacy preservation, distributed computing

Received on 14 July 2026, accepted on 30 August 2026, published on 02 November 2026

Copyright © 2026 Hua-Yong Wei, licensed to EAI. This is an open access article distributed under the terms of the CC BY-NC-SA 4.0, which permits copying, redistributing, remixing, transformation, and building upon the material in any medium so long as the original work is properly cited.

doi: 10.4108/eetsis.14026

1. Introduction

With the vigorous development of Internet of Vehicles technology, numerous latency-sensitive and computationally intensive services have emerged in on-board applications. However, the computing power and storage resources of the on-board computing units in

*Corresponding author. Email: wei20240814@163.com

intelligent vehicles are limited. When handling these highly complex computationally intensive services, significant latency may occur, or even system crashes. In addition, heavy computation also imposes a considerable energy burden on vehicles [1]. Therefore, addressing latency-sensitive and computationally intensive services in highly mobile vehicular networks is a critical issue for the further development of the Internet of Vehicles.

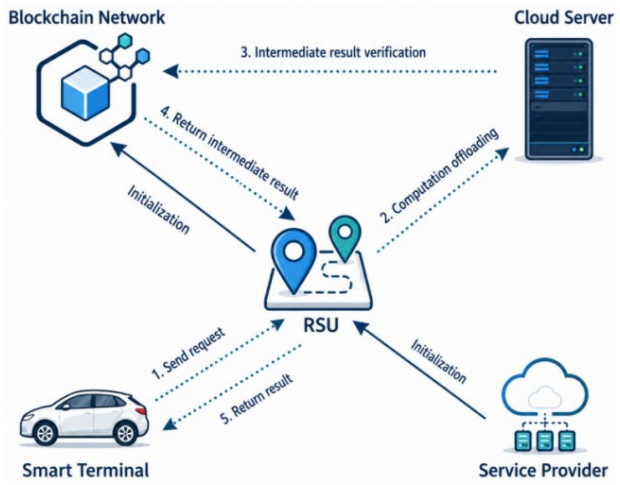


Figure 1. VEC framework architecture

A feasible solution to the above problem is computation offloading for computationally intensive vehicular services, which migrates large-scale local vehicle computations to cloud servers to improve computing efficiency and reduce vehicle energy consumption. Nevertheless, for latency-critical computing tasks, the transmission cost and communication delay between intelligent vehicles and remote cloud servers are unacceptable. With the emergence of Mobile Edge Computing (MEC) technology, researchers have integrated MEC with the Internet of Vehicles and proposed the Vehicular Edge Computing (VEC) [2] framework, whose architecture is illustrated in Figure 1.

The VEC framework remedies the insufficient computing capacity and limited energy supply of intelligent vehicles, enhancing the real-time performance and reliability of data processing, and providing high-performance, low-latency service guarantees for intelligent vehicles. Meanwhile, optimizing computing paradigms for edge clouds and remote clouds can further boost the efficiency of computation offloading. Recent research has further shown that MEC can jointly optimize offloading ratios, transmit power, and bandwidth allocation for resource-constrained IoT terminals under latency constraints [3]. For example, Guan et al. [4] proposed an outsourced computing scheme for polynomial calculations, which reduces computational complexity via Horner's rule and improves the efficiency of computation offloading accordingly.

Although the VEC framework has numerous advantages, it still faces various challenges in practical applications. Once data or computations from intelligent vehicles are uploaded to remote cloud servers, direct control over them is lost, and the computations are fully executed by semi-trusted third-party remote cloud servers. This leads to critical security risks including data leakage, privacy disclosure and internal attacks from cloud servers [5, 6]. Regarding the correctness and integrity of computation results, remote cloud servers may return arbitrary fake results to save computing resources or return erroneous results due to external malicious attacks.

Blockchain has been introduced into IoV for decentralized reputation management and distributed vehicular coordination [7, 8], while blockchain-assisted vehicular computation-offloading frameworks have been developed to improve secure and scalable edge-service coordination [9, 10]. Related cloud-provenance research has also shown that Ethereum-based ledgers can provide tamper-resistant records of data creation and processing operations, although transaction throughput and latency remain important scalability considerations [11]. Smart-contract-based cloud research further shows that blockchain and smart contracts can provide decentralized data-integrity verification and tamper-resistant access control [12]. In some blockchain-assisted designs, computationally intensive logic may be executed through smart contracts; however, Ethereum Gas fees increase with computational workload [13], so large on-chain computations can impose substantial economic cost. To reduce the Gas consumption of smart contracts, inspired by verifiable outsourced computing [14], a cloud-blockchain hybrid computation offloading scheme has been put forward [9]. The core idea is to outsource heavy computations to high-performance remote cloud servers, while smart contracts are used to verify the integrity of the remote computing results [15]. Since the verification cost of verifiable outsourced computing is far lower than the cost of executing the entire computation locally, this scheme can effectively reduce the overall Gas consumption of smart contracts.

It is worth noting that the inherent transparency of blockchain conflicts with privacy protection requirements. The outsourced computing functions of service providers may contain trade secrets that need to remain confidential. Meanwhile, computation requests submitted by intelligent vehicles may include sensitive information such as location data, driving trajectories and vehicle status, which must also be protected from unauthorized access. To address this issue, Yao et al. [16] proposed a blockchain-based privacy-preserving vehicular trust-management scheme that combines consortium blockchain and homomorphic encryption. Complementarily, edge-computing privacy research has shown that local training, edge aggregation, and adaptive differential privacy can reduce raw-data exposure during distributed communication [17]. Song et al. [18] developed a blockchain-based secure outsourcing

scheme for polynomial multiplication to protect outsourced polynomial data.

In addition, centralized cloud-blockchain computation services remain vulnerable to availability risks and single-point failures [19]. If a remote cloud server fails, the entire computing task may become blocked or stalled, seriously violating the low-latency requirement of vehicular network computing. Although fault-tolerance mechanisms such as redundancy, replication, checkpointing and recovery can mitigate such failures [20], recent cloud task-scheduling research has also shown that fault-aware scheduling can improve service availability and execution success under single-point failures [21]. However, such cloud-centric mechanisms are difficult to adapt directly to highly mobile VEC scenarios without additional coordination and latency control. Due to the high mobility of vehicles and the highly dynamic nature of vehicular networks, directly applying conventional fault-tolerance mechanisms may introduce excessive computational latency. Therefore, an efficient distributed fault-tolerant computation offloading scheme is urgently needed.

To solve the above problems, this paper proposes a cloud-blockchain integrated privacy-preserving distributed computation offloading scheme (FCOS):

(i) A redundant distributed computing method is designed. By introducing a redundancy mechanism into computation offloading, the scheme not only improves computational efficiency, but also effectively mitigates the impact of single-point failures, enhancing the robustness and reliability of the overall system.

(ii) Smart contracts are adopted to guarantee verifiability and fairness of computation offloading. Leveraging blockchain smart contract technology, the scheme ensures verifiability throughout the computation offloading process without increasing the computational overhead of intelligent vehicles, achieves fairness for both offloaded tasks and verification processes, and prevents interference from potential malicious nodes.

(iii) Privacy protection is realized via homomorphic encryption and blind factor techniques. At the intelligent vehicle terminal, homomorphic encryption and low-cost blind factor technology are applied to preserve the privacy of computing tasks and requests. This scheme effectively protects user data privacy while maintaining high efficiency in computation and communication.

(iv) Theoretical analysis and experimental validation are conducted. Theoretical analysis proves the theoretical advantages of FCOS in terms of computational efficiency, system reliability and privacy protection. Experimental results show that compared with existing technologies, FCOS reduces the average computation time by 90.4% for polynomial calculations within 5000 iterations. When the single-point failure rate ranges from 0% to 60%, the average computation time is reduced by 92.8% to 96.7%.

2. Related Work

The rapid development of the Internet of Vehicles has driven the prosperity of on-board applications, which consume an increasing amount of computing resources from intelligent vehicles. Constrained by the limited computing resources of vehicle terminals, intelligent vehicles struggle to execute computationally intensive tasks. To tackle this challenge, the Vehicular Edge Computing (VEC) framework has been introduced into the Internet of Vehicles. By offloading communication, computation and storage tasks to Mobile Edge Computing (MEC) servers, VEC enables lightweight computing for intelligent vehicle terminals.

2.1 Verifiable Computation

Third-party MEC servers operate outside the supervision of data owners and may return incorrect computation results due to economic incentives or external malicious attacks. To address this problem, Gennaro et al. [22] first proposed the concept of Verifiable Computation (VC) in 2010. VC allows data owners to outsource heavy computational tasks to high-performance cloud servers and verify the integrity of computation results with low overhead. Due to its broad application prospects, extensive research has been conducted by scholars worldwide [6]. According to computation types, verifiable computation can be categorized into general verifiable computation [23] and specific verifiable computation [24]. General verifiable computation supports arbitrary functions, and is typically built upon probabilistically checkable proofs, zero-knowledge proofs and quadratic span programs.

However, the verification overhead of these general methods is still far higher than the cost of local computation, limiting practical deployment and keeping them mostly at the theoretical research stage. Specific verifiable computation consists of protocols tailored for certain types of computations, such as verifiable polynomial computation protocols [25], verifiable matrix computation protocols [26], and verifiable set computation protocols [27]. Compared with general verifiable computation, specific verifiable computation features easier implementation, higher efficiency and simpler deployment.

2.2 Cloud-Blockchain Hybrid Computation Offloading Schemes

Although verifiable computation can guarantee the integrity of computation results, it cannot ensure fair interests for both users and cloud servers. Even if cloud servers return correct computation results, users may default and refuse to make payments; alternatively, cloud servers may return falsified results while claiming validity and demanding payment from users. Blockchain has been incorporated into vehicular edge computation offloading to

improve security, scalability, decentralized coordination and service accountability [10]. However, executing computationally intensive tasks directly on-chain can introduce latency, throughput and Gas-cost problems. Fair-payment approaches therefore combine outsourced cloud execution with blockchain-mediated verification and settlement [28]. The core idea is to outsource heavy computations to high-performance remote cloud servers, while smart contracts verify the correctness and integrity of the computation results. Similarly, a variety of cloud-blockchain hybrid computation offloading schemes have been developed for matrix, polynomial, set and other types of calculations.

Nevertheless, existing schemes suffer from two major drawbacks: privacy leakage and excessive latency. Outsourced computations and requests are submitted in plaintext, so semi-trusted cloud servers may leak sensitive private data. Furthermore, the inherent transparency of blockchain exacerbates the risk of privacy disclosure. Due to the high mobility of vehicles and the highly dynamic nature of vehicular networks, the centralized computing paradigm adopted by existing cloud-blockchain schemes cannot effectively mitigate single-point failures within the VEC framework using conventional approaches, failing to fully satisfy the strict low-latency requirements of vehicular networks. Therefore, an efficient privacy-preserving distributed fault-tolerant computation offloading scheme is urgently required.

2.3 Polynomial-Oriented Computation Offloading Schemes

Polynomial operations are ubiquitous in both scientific research and real-world applications, serving as a fundamental basis for matrix operations, set operations and other computations. Recently, an increasing number of cloud-blockchain hybrid computation offloading schemes have been designed for polynomial calculations.

Campanelli et al. [29] revisited zero-knowledge contingent payments and developed blockchain payment mechanisms for verifiable digital services; this work was not designed as a polynomial-specific VEC offloading protocol. Dong et al. [30] proposed smart counter-collusion contracts for verifiable cloud computing, using game-theoretic incentives and multiple cloud providers to discourage collusive cheating; its focus is general verifiable cloud computing rather than a dedicated polynomial offloading construction. These mechanisms provide useful insights into payment fairness and collusion resistance, but they do not directly address dual-privacy distributed polynomial recovery in highly dynamic vehicular edge environments.

Huang et al. [31] proposed Bitcoin-based fair payment for outsourced fog computation using commitment-based sampling and a semi-trusted third party for dispute resolution, which introduces an additional trust assumption rather than a trusted initialization authority. Guan et al. [4] adopted partial sampling, where smart contracts recalculate partial intermediate results of Horner's rule to verify the correctness and integrity of computation results. With partial sampling, verification accuracy is positively correlated with the sampling size.

Since untrusted cloud servers may leak outsourced polynomial data and the public transparency of blockchain can further increase privacy exposure, Song et al. [18] proposed blockchain-based secure outsourcing for polynomial multiplication. However, that construction does not target private vehicular query requests or distributed fault tolerance in highly dynamic VEC environments. More broadly, the above studies address complementary aspects such as payment fairness, verification, collusion deterrence or outsourced-data confidentiality, but they do not jointly provide private vehicular query protection, verifiable distributed recovery and fault tolerance for highly dynamic VEC environments. Recent secure-outsourcing research further highlights three issues that are directly relevant to FCOS. First, depth-aware arithmetization for homomorphic encryption shows that exponentiation, univariate functions, equality tests, comparisons, and Boolean primitives can be systematically represented by arithmetic circuits, providing a route from polynomial outsourcing to a broader class of functions [32]. Second, recent multi-party private-set computation explicitly removes non-collusion assumptions and demonstrates that collusion assumptions must be stated as part of the security model rather than treated implicitly [33]. Third, recent vehicular-edge studies jointly optimize task processing and limited server resources, emphasizing that latency, resource competition, and mobility-aware deployment should be evaluated in addition to pure computation time [34]. These developments motivate the revised FCOS analysis below, which explicitly separates function generality, cloud-server collusion, algebraic recovery threshold, and network-level deployment latency.

Scope beyond polynomial evaluation. FCOS directly supports bounded integer polynomial functions because its privacy and recovery mechanisms are constructed around polynomial coefficient encoding and interpolation. Matrix-vector and matrix-matrix computations can be decomposed into batches of inner-product or polynomial blocks, while set operations can be represented through characteristic-polynomial or private-set primitives. More general non-polynomial functions require an arithmetic-circuit representation or a hybrid extension using FHE/TEE/MPC techniques; therefore, the present paper does not claim native support for arbitrary functions. This distinction has been added to avoid overstating generality and to clarify a concrete extension path for matrix, set, and perception-related workloads.

3. Preliminaries

3.1 Homomorphic Encryption

Homomorphic Encryption (HE) is a form of encryption that allows direct computation on two ciphertexts without

decryption, and the resulting ciphertext corresponds to the ciphertext obtained by performing the same operation on the original plaintexts [35]. For any valid function f and plaintext m, it satisfies the following property:

$$f(\mathrm{Enc}(m)) = \mathrm{Enc}(f(m)) \tag{1}$$

Full Homomorphic Encryption (FHE) supports unlimited addition and multiplication operations on encrypted data [36] and can compute arbitrary functions yet suffers from low efficiency which limits practical deployment [37]. Partial Homomorphic Encryption only supports a limited set of operations (such as addition or multiplication) and achieves higher computational efficiency.

Paillier homomorphic encryption [38] is an additive homomorphic public-key cryptosystem proposed by Paillier, featuring additive homomorphism and scalar multiplication homomorphism. It consists of the following three algorithms:

(i) (Paillier.pk,Paillier.sk)←Paillier.KeyGen(1^{λ}) Randomly select two prime numbers p and q, such that $\gcd(pq,(p-1)\cdot(q-1))=1$; compute $N=p\cdot q$ and $k=\mathrm{lcm}(p-1,q-1)$, where lcm denotes the least common multiple; randomly select an integer $g\in\mathbb{Z}^{*}_{N^2}$, such that $\gcd\left(L\left(g^k \bmod N^2\right),N\right)=1$, where the function $L(x)=(x-1)/N$; compute $\mu=\left(L\left(g^k \bmod N^2\right)\right)^{-1}$; set the public key Paillier.pk=(N, g) and the private key Paillier.sk=(k, μ).

(ii) c←Paillier.Enc(Paillier.pk,m): For any plaintext message $m\in\mathbb{Z}_N$, pick a random number $r\in\mathbb{Z}^{*}_{N}$, and compute the ciphertext $c=g^m r^N \bmod N^2$.

(iii) m←Paillier.Dec(Paillier.sk,c): Given ciphertext c, recover the plaintext

$$m=\frac{L\left(c^k \bmod N^2\right)}{L\left(g^k \bmod N^2\right)} \bmod N = L\left(c^k \bmod N^2\right)\cdot\mu \bmod N. \tag{2}$$

The additive homomorphism property of Paillier encryption is expressed as follows:

(i) Given the ciphertexts c0 and c1 corresponding to messages m0 and m1, the ciphertext for m0+m1 can be computed by:

$$Paillier.Enc(Paillier.pk, m_0+m_1)=c_0\cdot c_1 \tag{3}$$

(ii) Given a constant k and the ciphertext c0 corresponding to message m0, the ciphertext for k·m0 can be computed by:

$$Paillier.Enc(Paillier.pk, k\cdot m_0)=\underbrace{c_0\cdots\cdots c_0}_{k\ \text{times}}=c_0^k \tag{4}$$

3.2 Lagrange Interpolation

Lagrange interpolation is a polynomial interpolation method that constructs a polynomial function passing exactly through a set of known points in the two-dimensional plane. Given k sample points of a polynomial function:

$$(x_0,y_0),(x_1,y_1),\ldots,(x_{k-1},y_{k-1}) \tag{5}$$

Assuming all distinct x_j values are unique, the unique polynomial of degree k−1 derived from the Lagrange interpolation formula is:

$$L(x):=\sum_{j=0}^{k-1} y_j \prod_{\substack{i=0\\ i\neq j}}^{k-1}\frac{x-x_i}{x_j-x_i} \tag{6}$$

Formally, given a set w of k distinct sample points, Lagrange interpolation is defined as:

$$L(x)=LagrangeForm(w) \tag{7}$$

From the above formula, the coefficient vector of the polynomial function can be expressed as:

$$\mathbf{D}=\begin{bmatrix} d_0 & d_1 & \ldots & d_{n-1}\end{bmatrix}^T \tag{8}$$

4. Definition of the FCOS Model

4.1 Threat Model

The cloud-blockchain integrated privacy-preserving distributed computation offloading scheme for the Internet of Vehicles involves five entities: service providers, intelligent vehicles, Road Side Units (RSUs), cloud servers, and the blockchain network. The overall architecture is illustrated in Figure 2.

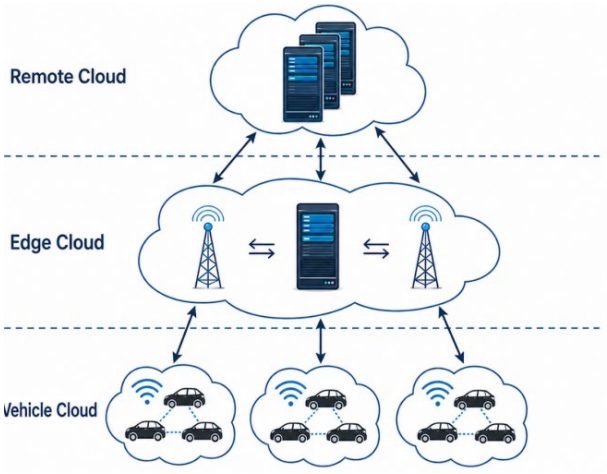


Figure 2. FCOS architecture diagram

Service Provider: An honest owner of computationally intensive services, who outsources these heavy tasks to cloud servers to provide computing services for intelligent vehicles in the Internet of Vehicles. The service provider gains revenue by offering services, and poor-quality or erroneous services will reduce its income. During protocol execution, the service provider cannot manipulate the computation process maliciously and always follows the protocol honestly in our model. Some of its business data contains trade secrets and requires strict privacy protection.

Intelligent Vehicle: A benign smart vehicle equipped with sensors, communication modules, on-board computing units and other devices. It can initiate service requests and conduct vehicular network communications. Intelligent vehicles request services while driving. Some vehicle data involves sensitive user privacy and cannot be directly uploaded in plaintext.

RSU (Road Side Unit): Honest and reliable edge servers deployed alongside road infrastructure, with local computing, storage and communication capabilities, enabling wireless communication with intelligent vehicles and remote cloud servers. RSUs have advanced security features to defend against malicious attacks, data tampering and information leakage. Encryption technologies are adopted to secure communications and data transmissions, protecting traffic information and user privacy.

Cloud Server: A distributed server cluster with large storage and computing resources. Cloud nodes are controlled by a static probabilistic-polynomial-time adversary that may corrupt and collude with up to $t_c < n$ cloud servers. Corrupted servers may share their local computation templates, encoded requests, encoded function data, public ledger information, timing observations, and returned results, and may either return malformed results or remain unavailable. They do not obtain the Paillier secret key, the vehicle-local blinding state, or unreleased RSU verification randomness. Collusion between cloud servers is therefore covered; collusion with the trusted RSU validator set or compromise of the service provider is outside the present security scope and is explicitly listed as a limitation.

Blockchain Network: A permissioned RSU-validator network that stores task commitments, verification evidence, and settlement records and executes the verification logic. The security analysis assumes an honest validator majority and deterministic finality of the permissioned chain. Cloud servers can observe public task and settlement metadata, but protected verification randomness is not exposed through the cloud-server interface before the corresponding result is fixed. A blockchain-majority corruption, validator-key compromise, or adaptive corruption of RSUs is not covered by the current proof.

Adversary and channel assumptions. Service providers, intelligent vehicles, and RSUs follow the protocol honestly in the current model. Vehicle-RSU, RSU-cloud, and RSU-validator channels are assumed to provide authenticated and confidential transport; denial-of-service may delay or suppress messages but does not alter authenticated messages without detection. The cloud adversary is static during one task instance. The privacy theorem therefore protects function coefficients and vehicle requests against a coalition of cloud servers under these assumptions, but it does not claim security if the RSU secret key or the vehicle-local blinding state is compromised.

4.2 Formal Definition

The proposed scheme consists of eight algorithms: the initialization algorithm, computation encoding algorithm, input request encoding algorithm, offline initialization algorithm, distributed computation algorithm, verification algorithm, recovery algorithm, and result decoding algorithm. The formal definition of the cloud-blockchain integrated privacy-preserving distributed computation offloading scheme for the Internet of Vehicles is: FCOS=(SetUp, EncodeComp, EncodeI, InitOffline, DisComp, Verify, Recover, DecodeR).

Admissible function and integer domain. To make the decoding and privacy claims mathematically well-defined, let $f(x)=\sum_{j=0}^{K-1} a_j x^j$ be an integer polynomial with input $x \in [0, B_x]$ and coefficient bounds $|a_j| \le B_o$. Let $B_f = \max_{x\in[0,B_x]} f(x)$, and restrict the current exact-decoding theorem to tasks for which $0 \le f(x) \le B_f$. The vehicle samples the blinding factor r uniformly from a finite admissible interval $R=[R_{min}, R_{max}]$ of size L, where $R_{min} > B_f$. In addition, the Paillier modulus N and the selected r must satisfy $B_{eval} = \max_{x\in[0,B_x], r\in R} |f(x+r)| < N/2$, so that homomorphic evaluation does not wrap around the Paillier plaintext modulus. Parameter sets violating this condition must be rejected, rescaled, decomposed into smaller blocks, or evaluated through a CRT/FHE extension. The

high-degree curves in Section 7 are therefore interpreted as operation-cost microbenchmarks unless the bounded-domain condition is separately satisfied.

The formal description of each algorithm is given as follows:

(i) Initialization algorithm: $\mathrm{SetUp}\left(F,1^{\lambda};B_x,B_a,R_{\min},R_{\max}\right)\rightarrow(\mathrm{pk},\mathrm{sk})$. A randomized algorithm executed by the RSU. It generates the Paillier key pair and publishes the admissibility bounds together with Paillier.pk, i.e., $\mathrm{pk}=\left\{Paillier.pk,B_x,B_a,R_{\min},R_{\max},N\right\}$; the secret key is sk=Paillier.sk. The task-specific EncodeI procedure additionally checks B_eval<N/2 before accepting an instance.

(i) Computation encoding algorithm: $\sigma_f\leftarrow\mathrm{EncodeComp}(f,pk)$. A randomized algorithm executed by the service provider. It takes a function f∈F and the public key *pk* as input and outputs the encoded function σ_f.

(ii) Input request encoding algorithm: $\mathrm{EncodeI}(x,\mathrm{pk})\rightarrow\left(\sigma_x,\mathrm{st}_x\right)$. A randomized algorithm executed by the intelligent vehicle. It samples a fresh blinding factor r from the admissible finite interval R, computes $\sigma_x=x+r$, outputs the encoded request σ_x, and stores stx=r only in vehicle-local session state. The value r is not transmitted to cloud servers or written to the blockchain.

(iii) Offline Initialization Algorithm: $\left(\left\{\sigma_f^{(i)}\right\}_{i\in[n]},\pi\right)\leftarrow\mathrm{InitOffline}\left(\sigma_f,n\right)$. A randomized algorithm executed by RSUs. It takes the encoded function σ_f and the number *n* of cloud servers in the cluster as input, and outputs computation templates $\left\{\sigma_f^{(i)}\right\}_{i\in[n]}$ and verification parameter π.

(iv) Distributed Computation Algorithm: $\sigma_w^{(i)}\leftarrow\mathrm{DisComp}\left(\sigma_f^{(i)},\sigma_x\right)$. A deterministic algorithm executed by cloud servers. It takes the computation template $\sigma_f^{(i)}$ and the encoded input σ_x as input and outputs the distributed computation result $\sigma_w^{(i)}$.Verification Algorithm: $\{0\ or\ 1\}\leftarrow\mathrm{Verify}\left(\sigma_w^{(i)},\Delta_i,\pi\right)$.

(v) A deterministic algorithm executed by the blockchain network. It takes the distributed computation result $\sigma_w^{(i)}$, the intermediate vector Δ_i, and the verification parameter π as input. It outputs 1 if verification passes; otherwise, it outputs 0.

(vi) Recovery Algorithm: $\sigma_y\leftarrow\mathrm{Recover}\left(\sigma_w,\sigma_f,\sigma_x,sk\right)$. A deterministic algorithm executed by RSUs. It takes the set of s distributed computation results $\sigma_w=\left\{\sigma_w^{(0)},\sigma_w^{(1)},\ldots,\sigma_w^{(s-1)}\right\}$, the encoded function σ_f, the encoded input σ_x and the private key *sk* as input, and outputs the encoded result σ_y, where s≤n.

(vii) Result Decoding Algorithm: DecodeR(σy,stx)→y. A deterministic algorithm executed by the intelligent vehicle. It takes the encoded result σy and the vehicle-local session state stx=r as inputs and outputs y=σy mod r. The public key is not sufficient for decoding the blinding layer; the secret blinding state r is required.

4.3 Definition of Correctness and Security

A secure and efficient cloud-blockchain integrated privacy-preserving distributed computation offloading scheme generally needs to satisfy five core properties: correctness, unforgeability of computation results, privacy of outsourced computations, privacy of computation requests, and optimality of the recovery threshold. Formal definitions of these properties are given below.

Definition 1. Correctness. For every admissible bounded task satisfying Section 4.2, if all honest protocol algorithms execute as specified and at least q verified cloud results are available, Verify accepts valid results, Recover reconstructs σ_y, and the requesting vehicle recovers y=f(x) using its local session state stx=r. Formally:

$$\Pr\left[\mathrm{Verify}=1\wedge\mathrm{DecodeR}\left(\mathrm{Recover}(\cdot),\mathrm{st}_x\right)=f(x)\right]\geq1-\mathrm{negl}(\lambda)\tag{9}$$

where negl(λ) denotes a negligible function and the probability is taken over protocol randomness for admissible parameters. In the revised notation, DecodeR takes the vehicle-local state st_x=r rather than the public key.

Definition 2. Unforgeability of Computation Results. Unforgeability targets a static coalition of at most tc semi-trusted cloud servers. The coalition may coordinate forged or inconsistent outputs and share all information available to corrupted cloud nodes but does not know the RSU secret key or unreleased verification randomness. A successful forgery is a coalition-generated result that differs from the correct distributed result yet passes Verify with non-negligible probability.

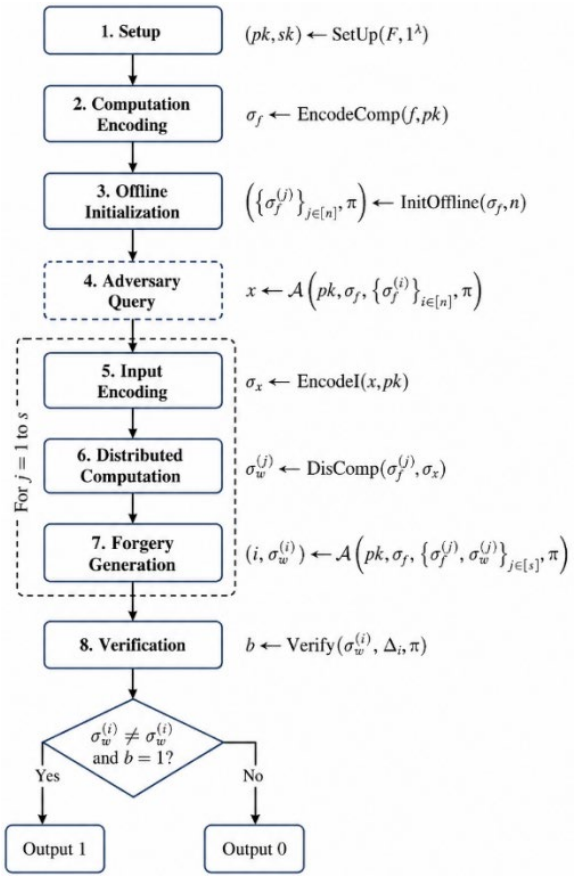


Figure 3. Unforgeability security experiment for the proposed scheme

$\mathrm{Exp}_{\mathsf{A}}^{\mathrm{Privacy}-x}[F, \lambda]$ defines whether an adversary A, given the public key *pk*, the encoded function σ_f, the computation templates $\{\sigma_f^{(i)}\}_{i\in n}$, the distributed computation results $\sigma_w^{(j)}$, and the verification parameter π, can forge a distributed computation result $\sigma_w^{(j)}$ that passes verification when $\sigma_w^{(i)} \neq \sigma_w^{(j)}$.

For any probabilistic polynomial-time adversary A, the scheme satisfies the unforgeability of computation results if:

$$\Pr\left[\mathrm{Exp}_{\mathsf{A}}^{\mathrm{Unforgeability}}[F, \lambda] = 1\right] \leq \mathrm{negl}(\lambda) \tag{10}$$

Definition 3. Cloud-Coalition Privacy. Let A be a static probabilistic-polynomial-time adversary controlling a coalition C of at most t_c cloud servers. The coalition view View_C contains p_k, the encoded function σ_f, the coalition's computation templates and local results, the blinded request σ_x, public blockchain metadata, and messages delivered to corrupted cloud nodes. It excludes Paillier.sk, the vehicle-local session state st_x=r, and unreleased RSU verification randomness. FCOS provides function privacy if equal-length admissible coefficient vectors induce computationally indistinguishable coalition views under Paillier semantic security, and it provides request privacy if admissible inputs induce statistically close blinded-request views.

For r sampled uniformly from the finite interval $R=[R_{min}, R_{max}]$ of size L and any $x_0, x_1 \in [0, B_x]$, the only x-dependent value exposed to the cloud coalition is $\sigma_x=x+r$. Hence the two blinded distributions are shifted finite uniform intervals and satisfy the following bound:

$$\mathrm{Enc}_{\mathrm{pk}}(A_0) \approx_c \mathrm{Enc}_{\mathrm{pk}}(A_1), \quad \Delta\left(\mathrm{View}_C(x_0), \mathrm{View}_C(x_1)\right) \leq |x_0 - x_1| / L \leq B_x / L \tag{11}$$

This definition does not claim privacy after compromise of the RSU secret key, compromise of the requesting vehicle's session state, corruption of a blockchain-validator majority, adaptive cross-session corruption, or excluded cross-role collusion. These boundaries are stated explicitly so that the theorem in Section 6.3 is evaluated under the same adversarial model.

Definition 4. Recovery Threshold and Fault Tolerance. Let K be the number of polynomial coefficients and $q = \sqrt{K}$. The offline decomposition pads the coefficient representation to a q×q block and distributes evaluation templates to n≥q cloud servers with distinct server indices. The recovery threshold is q verified results. The scheme tolerates up to n−q unavailable or rejected servers; more generally, if b Byzantine/invalid results are rejected by Verify and u servers are unavailable or delayed beyond the deadline, successful recovery requires n−b−u≥q.

5. Detailed Construction of the Scheme

5.1 Initialization Algorithm

Initialization Algorithm SetUp(F,1^{λ}): Taking polynomial functions as an example, initialize the function family F and generate a public/private key pair (*pk*, *sk*) according to the security parameter λ. The specific execution steps are illustrated in Figure 4.

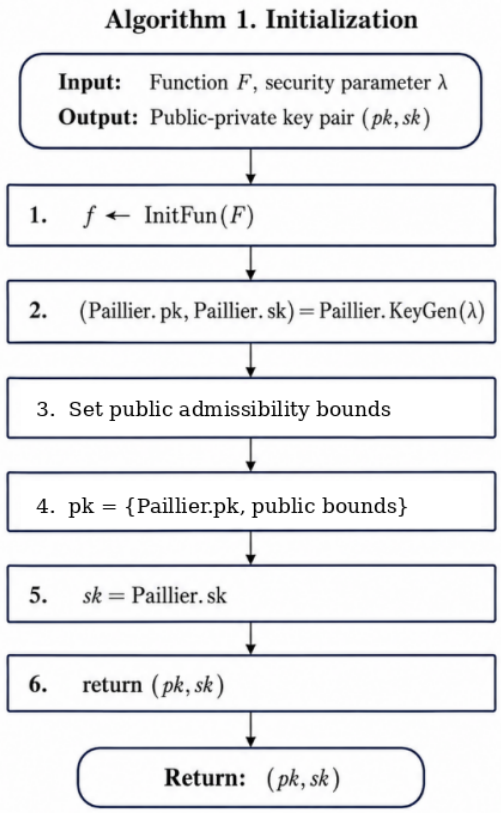


Figure 4. The procedure of Algorithm 1

5.2 Computation Encoding Algorithm

The main function of the EncodeComp(f, pk) algorithm is to encode the function f∈F to protect the data privacy of *f*. The procedures of EncodeComp(f, pk) algorithm is illustrated in Figure 5. The detailed execution steps are as follows:

(i) Without loss of generality, the polynomial function *f* is expressed as:

$$f(x) = a_0 + a_1 x^1 + \cdots + a_{k-1} x^{k-1} \bmod R \tag{12}$$

where $a_i \in \square, i \in [k]$. Let $A = [a_0 a_1 \cdots a_{k-1}]$ denote the coefficient vector of polynomial *f*.

(ii) Use Paillier.Enc(Paillier.pk,A) to encode the coefficient vector A, and obtain the encoded result:

$$\sigma_A = [c_0 c_1 \cdots c_{k-1}] \tag{13}$$

(iii) Output the encoded function:

$$\sigma_f(x) = c_0 + c_1 x^1 + \cdots + c_{k-1} x^{k-1} \tag{14}$$

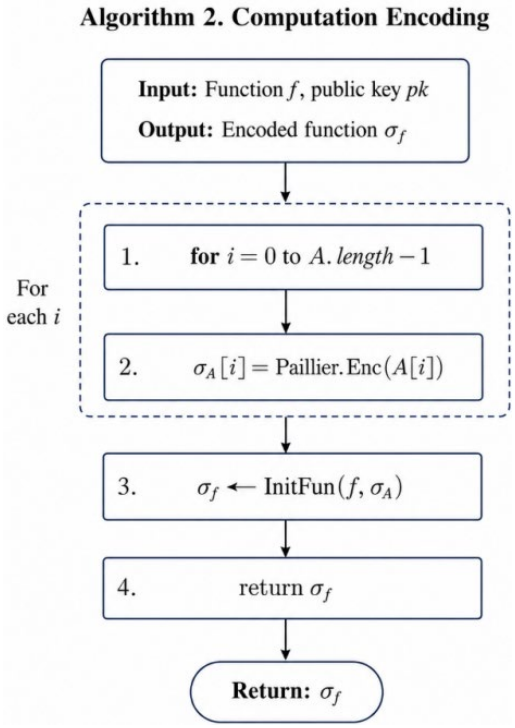


Figure 5. The procedures of Algorithm 2

5.3 Input Request Encoding Algorithm

EncodeI(x, pk): This algorithm blinds the bounded vehicle input x with a fresh vehicle-local random factor r. The public key contains the Paillier public key and the public admissibility bounds, but not r. Let R=[R_{min},R_{max}] be the admissible blinding interval defined in Section 4.2, with |R|=L and R_{min}>B_f. The intelligent vehicle samples r uniformly from R, stores stx=r locally, and never transmits r to the RSU, cloud servers, or blockchain.

(i) Parse the public key pk and the public admissibility parameters (B_x,B_a,R_{min},R_{max},N).

(ii) Uniformly sample r←R=[R_{min},R_{max}] and store st_x=r in vehicle-local session memory. Reject the task instance if the bound B_{eval}<N/2 is not satisfied.

(iii) Encode the input x as follows:

$$\sigma_x = x + r \tag{15}$$

(iv) Output the encoded input σ_x to the offloading protocol and retain st_x=r locally for DecodeR.

5.4 Offline Initialization Algorithm

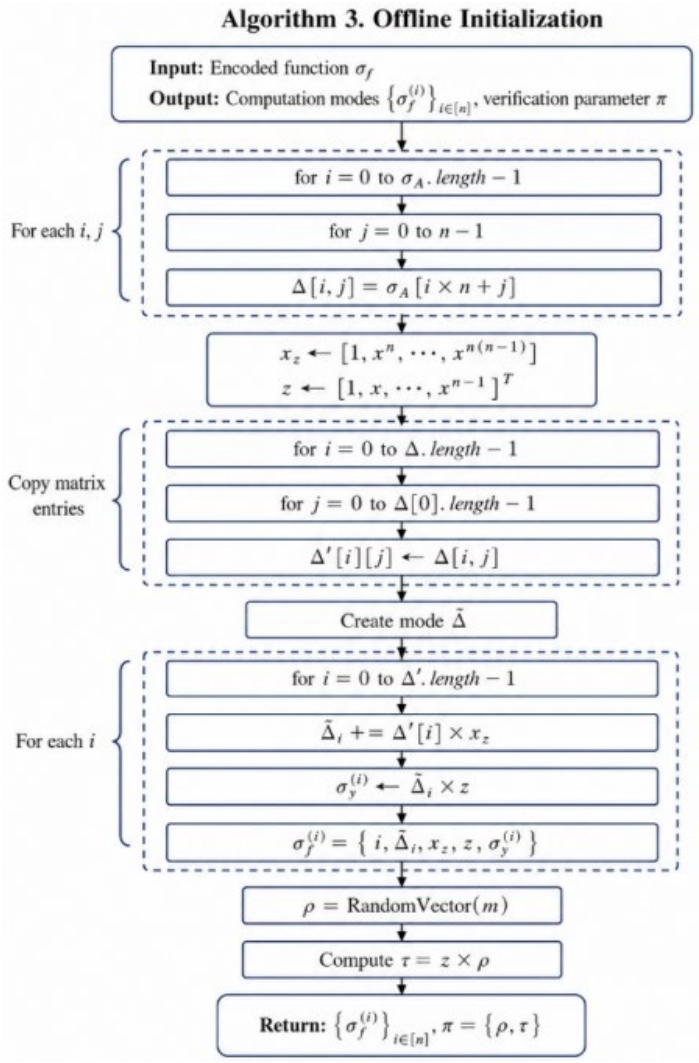


Figure 6. The procedures of Algorithm 3

Offline Initialization Algorithm InitOffline(σ_f,n): This algorithm (as illustrated in Figure 6) zero-pads the K encrypted polynomial coefficients into a q×q block with $q = \sqrt{K}$ and n≥q, then decomposes the encoded block into n computation templates for distributed evaluation and generates verification evidence π. Distinct nonzero server indices are assigned so that any q verified responses form a full-rank q×q Vandermonde system, consistent with Definition 4 and Theorem 3.

The specific execution steps are as follows:

(i) Construct the encoded coefficient matrix Δ from the encoded function σ_f:

(ii)

$$\Delta = \left[c_{i,j}\right] \tag{16}$$

where $c_{i,j} = c_{i\times n+j}$, $i \in [n]$, $j \in [n]$.

(iii) The expression of the encoded function σ_f is then formulated as:

$$\sigma_f(x) = x_z \Delta z \tag{17}$$

where $x_z = \left[1x^n \cdots x^{n(n-1)}\right]$, $z = \left[1x \cdots x^{n-1}\right]^T$. Here, x_i denotes placeholder variables for the user input.

(iv) Decompose the coefficient matrix Δ row-wise to obtain the decomposed expression:

$$\Delta = \left[\Delta_0 \Delta_1 \cdots \Delta_{n-1}\right]^T \tag{18}$$

where $\Delta_i = \left[c_{i,0} c_{i,1} \cdots c_{i,j}\right]^T$. Based on the above expression, let $\tilde{\Delta}_i^{(j)} = \Delta_j^T x_i^{j\alpha}$, where $x_i^{j\alpha}$ denotes the placeholder variable for the cloud server index during the computation of encoded results, and its assignment is completed in the online computation offloading phase, with i∈[n], j∈[n], $\alpha \in \square^+$.

Let $\delta_i = \left\{\Delta_i^{(0)}, \Delta_i^{(1)}, \ldots, \Delta_i^{(n-1)}\right\}$. From Section 1.5, we can derive that:

$$\tilde{\Delta}_i = \prod_{i \in [n]} \delta_i \tag{19}$$

(v) Based on the above initialization, the expression for the intermediate encoded result of vector multiplication is generated as:

$$\sigma_y^{(i)} = \tilde{\Delta}_i z \tag{20}$$

Thus, the decomposed computation template is obtained:

$$\sigma_f^{(i)} = \left\{i, \tilde{\Delta}_i, x_z, z, \sigma_y^{(i)}\right\} \tag{21}$$

(vi) Use the pseudorandom function G to generate a random verification vector ρ of length m, where m is a positive integer.

(vii) Compute the verification evidence for validating the distributed computation results:

$$\tau = z\rho \tag{22}$$

(viii) Output the computation templates $\left\{\sigma_f^{(i)}\right\}_{i \in [n]}$ and the verification parameters π={ρ, τ}.

5.5 Distributed Computation Algorithm

Distributed Computation Algorithm DisComp$\left(\boldsymbol{\sigma}_f^{(i)}, \boldsymbol{\sigma}_x\right)$: This algorithm (as illustrated in Figure 7) computes the distributed computation results using the computation template $\sigma_f^{(i)}$ generated by the offline initialization algorithm. The specific execution steps are as follows:

(i) Parse $\sigma_f^{(i)}$ to obtain the cloud server index i, the intermediate vector $\tilde{\Delta}_i$, the encoded input σ_x, and the expression for the intermediate encoded result $\sigma_y^{(i)}$.

(ii) Substitute the encoded request σ_x into the expression for z to compute $\tilde{z}_i$, then calculate the intermediate encoded result:

$$\sigma_{\tilde{y}}^{(i)} = \tilde{\Delta}_i \tilde{z}_i \tag{23}$$

(iii) Output the distributed computation result $\sigma_w^{(i)} = \left(i, \sigma_{\tilde{y}}^{(i)}\right)$.

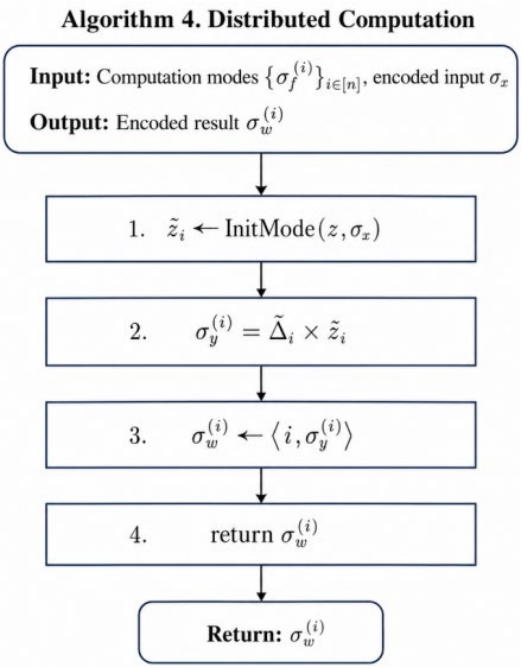


Figure 7. The procedures of Algorithm 4

5.6 Verification Algorithm

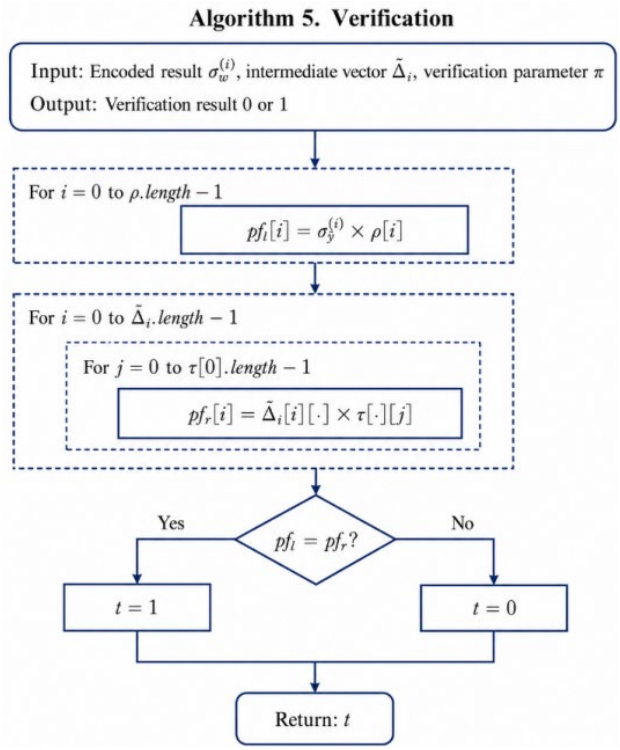


Figure 8. The procedures of Algorithm 5

Verification Algorithm $\text{Verify}\left(\sigma_w^{(i)}, \tilde{\Delta}_i, \pi\right)$: This algorithm (as illustrated in Figure 8) mainly verifies the distributed computation results returned by cloud servers with the verification parameter π. The detailed execution steps are listed below:

(iv) Parse $\sigma_w^{(i)}$ to acquire the intermediate encoded result $\sigma_{\tilde{y}}^{(i)}$, and parse the verification parameter $\pi=\{\rho, \tau\}$.

(v) Calculate the following verification items:

$$pf_l = \sigma_{\tilde{y}}^{(i)} \rho \tag{24}$$

$$pf_r = \tilde{\Delta}_i \tau \tag{25}$$

(vi) Verify whether the following equation holds:

$$pf_l = pf_r \tag{26}$$

(vii) Output 1 if the equation holds; otherwise, output 0.

5.7 Recovery Algorithm

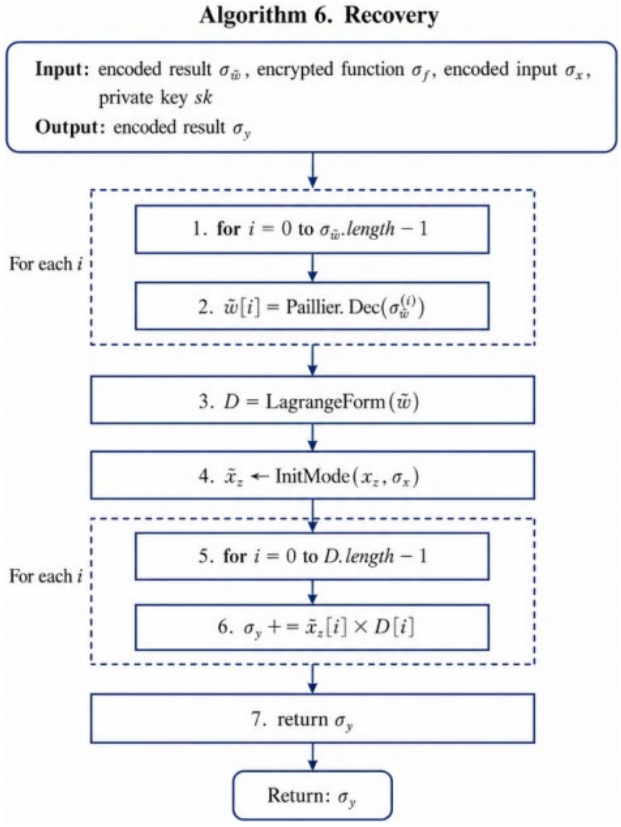


Figure 9. The procedures of Algorithm 6

Recovery Algorithm $\text{Recover}\left(\sigma_{\tilde{w}}, \sigma_f, \sigma_x, sk\right)$: This algorithm (as illustrated in Figure 9) decrypts the distributed computation results $\sigma_{\tilde{w}}$ and computes the final encoded result using the decrypted data. The specific execution steps are as follows:

(viii) (1) Receive the set of distributed computation results returned by the blockchain network: $\sigma_{\tilde{w}} = \left\{\sigma_{\tilde{w}}^{(i)} \mid i \in P, \mid P \mid = s \le n\right\}$, where P denotes the index set of the first s cloud servers that pass verification.

(ix) Call $Paillier.Dec\left(sk, \sigma_{\tilde{w}}\right)$ to decode the distributed computation results $\sigma_{\tilde{w}}$, obtaining:

$$\tilde{w} = \left\{\left(i, \tilde{y}^{(i)}\right) \mid i \in P, \mid P \mid = s \le n\right\} \tag{27}$$

where $\tilde{y}^{(i)}$ is the plaintext corresponding to the verified distributed computation result $\sigma_{\tilde{y}}^{(i)}$.

(x) Utilize the decoded distributed computation results $\tilde{w}$ to obtain the corresponding polynomial coefficient vector via Lagrange interpolation $LagrangeForm(\tilde{w})$:

$$D = \left[d_0 d_1 \ldots d_{n-1}\right]^T \tag{28}$$

(xi) Construct the vector $\tilde{x}_x$:

$$\tilde{x}_x = \left[1 \sigma_x^n \ldots \sigma_x^{n(n-1)}\right] \tag{29}$$

(xii) Compute the encoded result by right multiplying the polynomial coefficient vector D with the vector $\tilde{x}_x$:

$$\sigma_y = \tilde{x}_x \cdot D \tag{30}$$

(xiii) Output the encoded result σ_y.

5.8 Result Decoding Algorithm

Result Decoding Algorithm DecodeR(σ_y,st_x): This algorithm is executed only by the requesting intelligent vehicle. It parses the local session state st_x=r and recovers the exact bounded result from σ_y. The random blinding factor r is generated in EncodeI and remains local to the vehicle throughout the protocol; it is neither a public-key component nor a value that must be transmitted back by the RSU.

Decode the encoded result σ_y to obtain the final output:

$$y = \sigma_y \bmod r \tag{31}$$

Output the final result y.

Remark: FHE (Fully Homomorphic Encryption) can also be used to encode the polynomial coefficient vector A and the input x. We refer to this variant scheme as FCOS-F. Its basic workflow is consistent with the original FCOS scheme. In the EncodeComp and EncodeI algorithms, fully homomorphic encryption is applied to the polynomial coefficient vector A and the input x. In the Recover algorithm, the private key of the FHE scheme is used for direct decryption to retrieve the corresponding result.

5.9 Algorithm Example

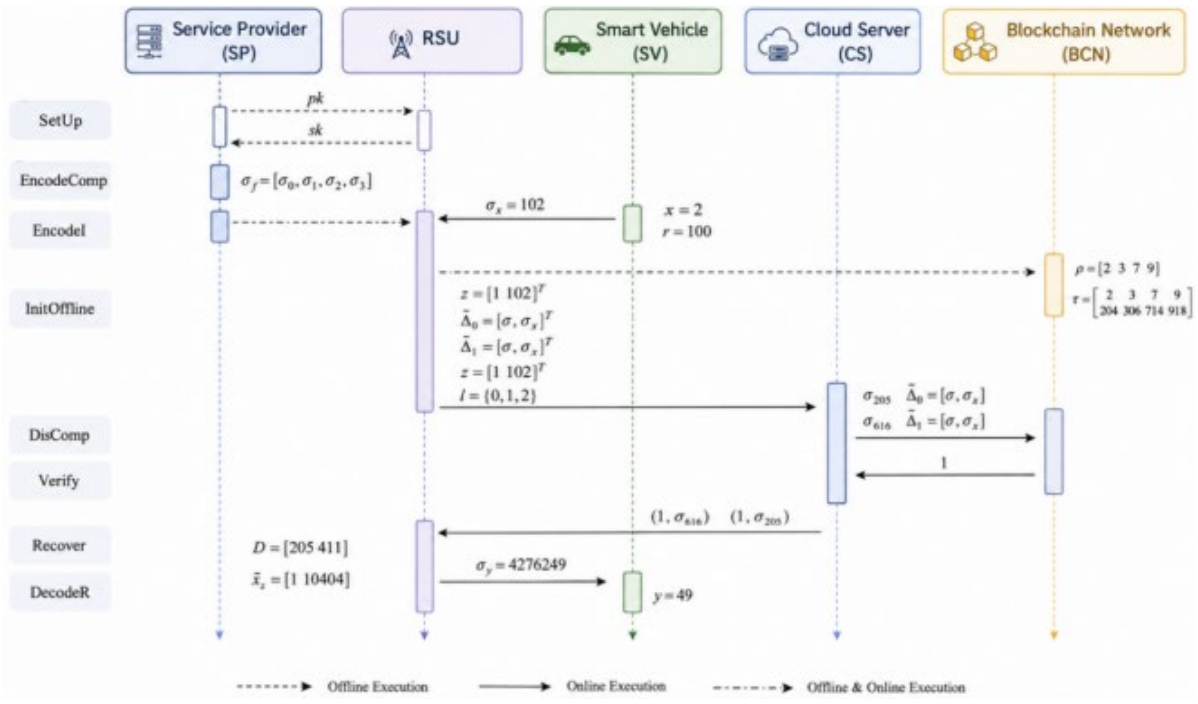


Figure 10. Algorithm flow example

Figure 10 presents a simple and representative example to help understand the details of the proposed scheme (the data used in practical applications and our formal experiments are far larger than this example). In this

example, we adopt the cubic univariate polynomial $f(x)=1+2x+3x^2+4x^3$ as the function f, with input x=2. Under these parameters, the final result can be directly calculated as $f(2)$=49. We select the blinding factor r=100, which satisfies the constraints of the scheme for this example. For simplicity of illustration, σ_1 denotes the ciphertext of coefficient 1 encrypted by the Paillier algorithm.

6. Correctness and Security Analysis

6.1 Correctness

If the scheme satisfies the correctness definition given in Definition 1, the verification algorithm Verify outputs 1, and the encoded result σ_y output by the recovery algorithm Recover can be successfully decoded by the result decoding algorithm to obtain the computation result y=f(x). Therefore, we need to verify that Equation (26) and Equation (31) hold. We first present Lemma 1, which is required for proving Equation (31).

Lemma 1. Polynomial Congruence under Integer Blinding. Let $f(z)=\sum_{j=0}^{K-1}a_j z^j$ have integer coefficients. For any integers x and r>0, f(x+r)≡f(x) (mod r). If, in addition, 0≤f(x)<r and the Paillier evaluation satisfies |f(x+r)|<N/2 so that no plaintext-modulus wraparound occurs, then the centered/plain integer value recovered from the Paillier layer satisfies f(x+r) mod r=f(x).

$$f(x+r)\bmod r = f(x) \tag{32}$$

Proof: By the binomial theorem, $(x+r)^j = x^j + r\cdot q_j(x,r)$ for an integer polynomial q_j. Hence f(x+r)−f(x)=r·Q(x,r) for an integer Q, so f(x+r)≡f(x) (mod r). Exact, rather than merely congruent, decoding follows from the range condition 0≤f(x)<r. Separately, the bound |f(x+r)|<N/2 ensures that the Paillier plaintext recovered before the modulo-r step is the intended integer rather than a value that has wrapped modulo N. Therefore, both a blinding-range condition and a Paillier-modulus condition are necessary; the previous ambiguous expression r≥max(f) has been removed.

We now prove that Equation (26) and Equation (31) hold. According to Equation (23) in the distributed computation algorithm DisComp, Equation (24) in the verification algorithm Verify can be rewritten as:

$$pf_l = \sigma_{\tilde{y}}^{(i)}\rho = \tilde{\Delta}_i\tilde{z}_i\rho \tag{33}$$

Similarly, from Equation (7) and Equation (9), Equation (12) in the verification algorithm Verify can be rewritten as:

$$pf_l = \sigma_{\tilde{y}}^{(i)}\rho = \tilde{\Delta}_i\tilde{z}_i\rho \tag{34}$$

It is clear that Equation (26) holds based on the above derivation. We now analyze the correctness of Equation (31). Based on the additive homomorphism property of the Paillier cryptosystem, each element in Equation (28) of the recovery algorithm can be expressed as $d_i = Paillier.\mathrm{Dec}\left(\Delta_i^T z\right)$. Combined with Equation (29), Equation (30) can be expanded as follows:

$$\begin{aligned}\sigma_y &= \tilde{x}_z\cdot D \\ &= \left[1\sigma_x^n\ldots\sigma_x^{n(n-1)}\right]\cdot D = \left[1\sigma_x^n\ldots\sigma_x^{n(n-1)}\right]\cdot \mathrm{Paillier.Dec}\left(\left[\Delta_0^T z\Delta_1^T z\ldots\Delta_{n-1}^T z\right]^T\right) \\ &= \sum_{i=0}^{n-1}\sigma_x^{ni} z^T\ \mathrm{Paillier.Dec}\left(\Delta_i\right)\end{aligned} \tag{35}$$

Combined further with Equation (12) and Equation (17):

$$\begin{aligned}\sigma_y &= \sum_{i=0}^{n-1}\sigma_x^{ni}\cdot\begin{bmatrix}1 & \sigma_x & \ldots & \sigma_x^{n-1}\end{bmatrix}\cdot\begin{bmatrix}a_{i,0} & a_{i,1} & \ldots & a_{i,j}\end{bmatrix}^T \\ &= \sum_{i=0}^{n-1}\begin{bmatrix}\sigma_x^{ni} & \sigma_x^{ni+1} & \ldots & \sigma_x^{ni(n-1)}\end{bmatrix}\cdot\begin{bmatrix}a_{i,0} & a_{i,1} & \ldots & a_{i,j}\end{bmatrix}^T \\ &= \sum_{i=0}^{n-1}a_{i,0}\sigma_x^{ni} + a_{i,1}\sigma_x^{ni+1} + \cdots + a_{i,j}\sigma_x^{ni(n-1)} = \sum_{i=0}^{n-1}a_i\sigma_x^i\end{aligned} \tag{36}$$

where $i,j\in[n]$, $a_{i,j}=a_{i\times n+j}$. From Equation (15), we have:

$$\sigma_y = f\left(\sigma_x\right) = f(x+r) \tag{37}$$

By Lemma 1:

$$f(x+r)\bmod r = f(x) = y \tag{38}$$

Therefore, Equation (31) holds for admissible bounded tasks satisfying the conditions above. For parameter sets that violate the range or Paillier-modulus bound, FCOS must first rescale/decompose the workload or use a larger-modulus/CRT/FHE variant; the correctness theorem does not cover such parameter sets.

6.2 Unforgeability of Computation Results

Theorem 1. Unforgeability of Computation Results. If the pseudorandom function G is secure, no static probabilistic-polynomial-time coalition of at most t_c corrupted cloud servers can produce a forged distributed result that differs from the correct result and nevertheless passes Verify with non-negligible probability, under the threat-model boundary of Section 4.1.

Proof: Suppose the adversary A returns a forged encoded result $\sigma_{\tilde{w}}^{(i)} \neq \sigma_w^{(i)}$ that passes verification, where $\sigma_{\tilde{w}}^{(i)} = \left\{i, \sigma_{\tilde{y}}^{(i)}\right\}$. Then we have:

$$\begin{cases} \sigma_{\tilde{y}}^{(i)} \rho = \tilde{\Delta}_i \tau \\ \sigma_{\tilde{y}}^{(i)} \neq \sigma_y^{(i)} \end{cases} \tag{39}$$

For the valid encoded result that passes verification, the following equation holds:

$$\sigma_y^{(i)} \rho = \tilde{\Delta}_i \tau \tag{40}$$

Subtracting the two equations yields:

$$\left(\sigma_{\tilde{y}}^{(i)} - \sigma_y^{(i)}\right) \rho = 0 \tag{41}$$

Since $\sigma_{\tilde{y}}^{(i)} \neq \sigma_y^{(i)}$, the vector ρ must be an all-zero vector of length m, i.e., $\rho = \vec{0}_{1\times m}$. The vector ρ is generated by the pseudorandom function G and satisfies $\rho \in \mathsf{F}_q^m$. We further derive:

$$\begin{aligned} \Pr\left(\rho = \vec{0}_{1\times m}\right) &= \prod_{p\in\rho, p\in\mathsf{F}_q} \Pr(p=0) \\ &= q^{-m} \end{aligned} \tag{42}$$

Thus:

$$\Pr\left[\mathrm{Exp}_{\mathsf{A}}^{\text{Unforgeability}}[F,\lambda]=1\right] \leq q^{-m} \tag{43}$$

If a colluding cloud coalition could successfully construct a forged encoded result that passes Verify with non-negligible probability, the same coalition can be represented by a single PPT distinguisher that combines all coalition-visible information. This would yield an efficient procedure contradicting the pseudorandomness assumption on G. Therefore, subject to the stated RSU/validator trust boundary, coordinated cloud-server forgery does not increase the success probability beyond a negligible amount.

6.3 Privacy

Theorem 2. Function and Request Privacy against Colluding Cloud Servers. Assume that Paillier encryption is semantically secure, the RSU secret key is not corrupted, and r is sampled uniformly from the finite interval R=[R_{min},R_{max}] with L=|R|. Then (i) the polynomial coefficients remain computationally hidden from any coalition of at most t_c cloud servers, and (ii) for any two admissible vehicle inputs x0,x1∈[0,B_x], the statistical distance between the blinded-request distributions σ=x0+r and σ=x1+r is at most |x0−x1|/L≤B_x/L. Thus, choosing $L \geq 2^{\lambda_b}$ with $B_x / 2^{\lambda_b}$ negligible yields statistical request privacy with respect to the cloud view.

Proof sketch. For function privacy, all cloud nodes observe Paillier ciphertexts of the coefficient vector and values derived from those ciphertexts without the decryption key. A coalition only aggregates more ciphertexts of the same semantically secure encryption system; under the standard Paillier assumption, replacing the encrypted coefficient vector by encryptions of an equal-length dummy vector is computationally indistinguishable. For request privacy, r is uniform over a finite contiguous interval. The distributions of x0+r and x1+r are uniform over two intervals shifted by |x0−x1|; their statistical distance is exactly the fraction of non-overlapping points and is therefore upper-bounded by |x0−x1|/L. This argument replaces the previous informal statement that x+r is simply "random."

Collusion boundary. The above result covers collusion among cloud servers because they do not possess r or the Paillier secret key. It does not cover collusion between a cloud coalition and the requesting vehicle (which trivially reveals that vehicle's own x and r), compromise of the trusted RSU holding the Paillier secret key, or corruption of a blockchain-validator majority. Adaptive corruption across protocol sessions and side-channel leakage are also outside the current model and are identified as future work.

6.4 Optimality of the Recovery Threshold

Theorem 3. Recovery Threshold. Let K be the number of coefficients in the outsourced polynomial and $q = \sqrt{K}$. After zero-padding, the coefficient representation is embedded in a q×q block. Let n≥q cloud servers receive templates indexed by distinct nonzero values $i_1,\ldots,i_n$. Any q verified server responses are sufficient for recovery, whereas fewer than q verified responses are insufficient for unique recovery in the general case.

Proof. Select any q verified responses with distinct indices $i_1,\ldots,i_q$. The corresponding interpolation system has a q×q Vandermonde coefficient matrix V with $V_{uv} = \left(i_u\right)^{(v-1)}, u,v = 1,\ldots,q$. Its determinant is $\det(V) = \prod_{1\leq u<v\leq q} \left(i_v - i_u\right)$, which is nonzero because all server indices are distinct. Hence rank(V)=q and the q unknown block coefficients are uniquely determined. Repeating this reconstruction across the q block columns recovers the padded q×q coefficient block and thus the original K coefficients and final encoded result.

Necessity and failure budget. With only q−1 verified responses, the corresponding matrix has rank at most q−1, leaving at least one degree of freedom and therefore multiple coefficient blocks consistent with the observations; unique recovery cannot be guaranteed. Consequently, the recovery threshold is q, not an unspecified number of returned values. With n servers, the maximum number of omission/timeout failures tolerated is f_{max}=n−q. If b returned values fail Verify and u servers do not respond before the deadline, recovery succeeds exactly when n−b−u≥q. This relation also distinguishes invalid results, unavailable nodes, and delayed responses from one another.

7. Performance Analysis

7.1 Theoretical Analysis

This section compares the proposed FCOS scheme with FOPC [4], POCS [18], PPOC [39] and FCOS-F in terms of functionality and performance. We first briefly describe FOPC, POCS and PPOC.

FOPC: (1) The outsourced calculator evaluates the polynomial via Horner's rule and stores the intermediate results of the Horner's rule computation. (2) The verification process samples the Horner's rule computation process and recursively completes the subsequent calculations for the sampled parts.

POCS: (1) Protect the polynomial data before outsourcing. (2) Delegate polynomial computation to cloud resources under blockchain-assisted coordination. (3) Verify the returned result through the blockchain-supported outsourcing process.

PPOC: (1) Encrypt the polynomial and input values via a homomorphic encryption algorithm, then send the ciphertexts to a single server. (2) The server re-encrypts the ciphertexts using homomorphic proxy re-encryption and masking techniques, then forwards them to another server; the two servers jointly perform the computation. (3) Conduct public verification over the ciphertexts, followed by private verification using public verification data and decrypted data.

Table 1. Functionality comparison

Scheme	Verifiable	Distributed	Privacy Protection - Input	Privacy Protection - Coefficient	Security	Fault Tolerance	Cost
FOPC	✓	×	×	×	Low	Low	Low
POCS	✓	×	×	✓	High	Low	Low
PPOC	✓	✓	✓	✓	High	Low	—
FCOS-F	✓	✓	✓	✓	High	High	High
Proposed Scheme	✓	✓	✓	✓	High	High	Low

Note: ✓ = Supported, × = Not Supported.

In terms of functionality, we compare the proposed scheme with the other schemes regarding verifiability of computation results, form of computation offloading, level of data privacy protection, security, fault tolerance and cost. The comparison results are shown in Table 1. Compared with the other schemes, our distributed computation mode improves fault tolerance. Unlike FOPC and POCS, the proposed scheme protects the privacy of both input values and polynomial coefficients, achieving stronger security. Compared with PPOC, our verification algorithm leverages blockchain technology to ensure tamper-resistance and fairness of the verification process, while reducing smart contract overhead via optimized verification algorithm design.

PPOC explicitly uses a two-server construction to address collusion in verification, whereas FCOS protects against a coalition of cloud servers only under the trust boundary defined in Section 4.1: the coalition does not obtain the RSU secret key or vehicle-local blinding state. Recent work that removes non-collusion assumptions in multi-party private computation [33] illustrates why this boundary must be stated explicitly. FCOS therefore should not be interpreted as providing security under arbitrary cross-role collusion.

Table 2. Computational efficiency comparison

Scheme	EncodeComp	Encodel	DisComp/Comp	Verify	Recover	DecodeR
FOPC	—	—	$O(k)$	$O(\sqrt{nk}$	—	—
POCS	$O(\log_2 k)$	—	$O(2k)$	$O(\log_2$	—	—
PPOC	$O(k)^*$	$O(1)^*$	$O(k)$ $+ O(2k)^*$	$O(k)$	$O(\sqrt{k})$ $+ O(\sqrt$	$O(1)^*$
FCOS-F	$O(k)^*$	$O(1)^*$	$O(\sqrt{k})$	$O(m\sqrt{k}$	$O(k)$ $+ O(\sqrt$	$O(1)^*$
Proposed	$O(k)^*$	$O(1)$	$O(\sqrt{k})$	$O(m\sqrt{k}$	$O(k)$ $+ O(\sqrt$	$O(1)$

Note: k denotes the polynomial degree $0 \le n \le 1$; m denotes the row dimension of the verification matrix.
** indicates complexity measured by the number of encryption algorithm rounds.*

In terms of computational efficiency, we compare the algorithms of the proposed scheme with the corresponding algorithms of other schemes, as shown in Table 2. Since the SetUp and InitOffline algorithms can be executed offline and do not affect the overall computational efficiency of the scheme, they are not listed in the table. For the EncodeComp algorithm, POCS protects polynomial data before outsourcing, while PPOC and our scheme encrypt each coefficient via an encryption algorithm, leading to different computational efficiency baselines. The complexity of the EncodeComp algorithm for both PPOC and our scheme can be reduced from O(k)* to O(1)* via multi-thread parallel encryption. Regarding the complexity of the DisComp/Comp algorithm, our

distributed computation framework reduces the complexity of centralized computation, achieving obvious advantages over FOPC and POCS. In contrast, PPOC requires re-encryption during computation to resist collusion attacks in verification, resulting in higher computational complexity. For the Verify algorithm, FOPC has the lowest complexity, yet its verification process is probabilistic. The remaining schemes support deterministic verification. When the polynomial degree is large, POCS achieves lower complexity than our scheme. When the dimension m of the verification matrix satisfies the stated condition, PPOC has higher complexity.

The complexity of the Recover algorithm in our scheme stems from polynomial interpolation, which is higher than PPOC, yet our scheme is not tied to a specific interpolation algorithm. Therefore, its complexity can be further optimized by selecting different interpolation methods, unlike PPOC and FCOS-F which rely on fixed decryption/encryption operations. Finally, for the request encoding algorithms EncodeI and decoding algorithm DecodeR, our blinding-based design also demonstrates notable efficiency advantages.

7.2. Experimental Analysis

The experimental code is implemented in Python 3.6.13 and Solidity 0.8.17, running on an Ubuntu 18.04 system with an Intel (R) Xeon (R) Platinum 8269CY CPU. Unless otherwise specified, the polynomial coefficients and input data in the timing experiments are randomly generated 64-bit integers, and a 2048-bit Paillier modulus configuration is used. These large-degree timing curves are operation-cost microbenchmarks; exact integer decoding is guaranteed only for bounded instances satisfying the admissibility and non-wraparound conditions in Section 4.2. The Solidity measurements report contract execution/Gas cost rather than public-mainnet confirmation latency. No physical 5G/RSU testbed or public-chain confirmation experiment is claimed in the present study.

Comparative Experiments with Existing Schemes

To conduct a comprehensive evaluation, we compare three variants of the proposed FCOS scheme with different privacy protection levels (FCOS with no privacy protection, FCOS-X using only EncodeI, and FCOS-Y using both EncodeI and EncodeComp) against other baseline schemes lacking privacy protection.

Baseline-equivalence clarification. PPOC is treated as the closest security-oriented baseline because it protects both input values and polynomial coefficients and supports verification, whereas FOPC and POCS primarily serve as verifiability/efficiency baselines with weaker privacy coverage. FCOS-F targets the same dual-privacy goal using fully homomorphic encryption and remains in the theoretical complexity comparison, but is omitted from runtime plots because its substantially larger cost would obscure the other curves. The trust assumptions are not identical across all schemes; therefore, the revised text separates functionality coverage from raw runtime comparisons rather than implying perfectly matched security models.

In addition, the excessive computational complexity of fully homomorphic encryption impairs the readability of experimental results, so the results for the FCOS-F scheme are not plotted in the figures. The experimental results are analyzed as follows. First, we analyze the computational overhead of the EncodeComp algorithm with varying secure prime sizes and input data sizes. We test the EncodeComp algorithm using 1024-bit, 2048-bit and 3072-bit secure primes, as well as 64-bit, 128-bit and 256-bit input values. The results are shown in Figure 11.

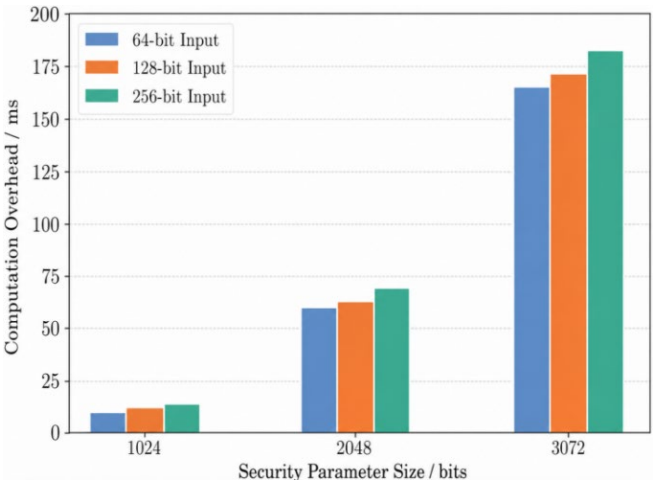


Figure 11. Experimental results of computational overhead for the encodecomp algorithm

The results indicate that the size of the secure prime has a more significant impact on computational overhead. When the secure prime size remains fixed, the increase in overhead caused by larger input data is relatively small. From a data privacy security perspective, larger secure primes offer stronger security but incur substantially higher computational overhead. To balance data security and computational cost, a 2048-bit secure prime is adopted throughout the experiments.

The results of the comparative experiments with existing schemes are shown in Figure 12. We analyze the impact of the privacy-preserving algorithms on the proposed scheme as follows. From Figure 12(a), Figure 12(b) and Figure 12(e), the curves of FCOS-X nearly overlap with those of FCOS, which indicates that the EncodeI algorithm has negligible impact on the computational overhead of the other algorithms. The EncodeComp algorithm has a relatively large overall overhead impact. However, as shown in Figure 12(c), FCOS-Y achieves lower computational overhead than PPOC and POCS for high-degree polynomials,

demonstrating the advantages of the distributed computation framework of our scheme.

We now analyze the results of the comparative experiments with existing schemes. The main complexity of the SetUP algorithm stems from key generation, and all schemes share the same complexity in this regard, so we use our scheme for the following analysis. As shown in Figure 12(a), the EncodeComp algorithm has a notable impact on the computational overhead of the SetUP algorithm; however, this overhead grows linearly with the polynomial degree and remains relatively low overall. In Figure 12(b), the InitOffline algorithm is unaffected by the privacy-preserving algorithms and incurs minor computational overhead. Although our scheme requires an extra run of the InitOffline algorithm, it only needs to be executed offline once when no data updates occur, resulting in a negligible overall impact. From Figure 12(c), for the DisComp algorithm, FCOS and FCOS-X exhibit very similar runtime performance, both far outperforming all other baseline schemes. Compared with PPOC at the same privacy protection level, FCOS-Y reduces the average polynomial computation time by 90.4% within 5000 computation rounds. The overall computational overhead of the Recover algorithm in our scheme is higher than that of PPOC, primarily due to the polynomial interpolation subroutine, which leaves room for further optimization. Finally, the DecodeR algorithm has minimal computational overhead and is unaffected by the EncodeComp algorithm, achieving the expected goal of reducing the computational complexity of smart vehicles while preserving data privacy in the outsourced computation framework.

Specifically, the evaluation metric for the Verify algorithm is Gas consumption (denominated in 10^{-18} Ether), with 256-bit unsigned integers used as input data. As shown in Figure 12(d), the performance varies with polynomial degree. The proposed scheme outperforms FOPC and POCS at low polynomial degrees, and outperforms OPCS at high polynomial degrees. Considering the probabilistic verification of FOPC and POCS, as well as the high overhead of OPCS when computing high-degree polynomials, there exists a trade-off between verification randomness and Gas consumption across these outsourced computation schemes.

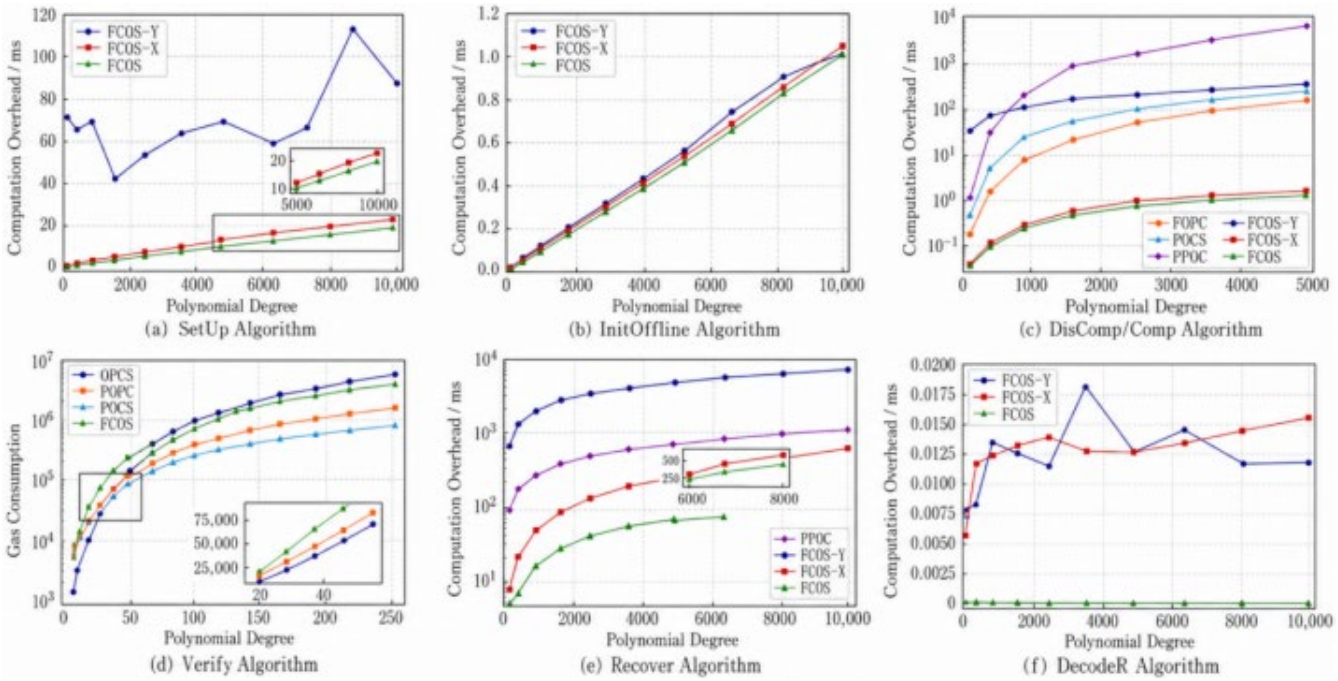


Figure 12. Comparative experimental results of existing schemes and the proposed scheme

The trade-off regarding randomness is relatively complex, so we do not conduct further in-depth research on it here. Below we perform a quantitative analysis of the Gas consumption trade-off between the proposed scheme and OPCS. Based on the Ethereum-to-USD exchange rate as of April 28, 2024 (1 ETH ≈ 3307.94 USD), partial cost data after converting Gas consumption to fiat currency is shown in Table 3. Table 3 lists the computation costs (in US dollars) for four representative polynomials. The data indicates that the break-even polynomial degree for equal costs between the proposed scheme and OPCS lies between 121 and 144.

Therefore, our scheme achieves better cost efficiency for polynomials of degree 144 or higher. Moreover, as the polynomial degree increases, the proposed scheme reduces Gas costs to an increasingly significant extent.

Table 3. Calculation costs

Polynomial Degree	Proposed Scheme / USD $\times 10^{-9}$	Direct Calculation / USD $\times 10^{-9}$	Cost Reduction / USD $\times 10^{-9}$
121	1.8669	1.8146	—
144	2.4004	2.5539	0.154
484	14.065	28.180	14.1
576	18.167	39.847	21.7

Fault Tolerance under Erroneous-Result and Availability Models

We conduct four comparative experiments between the OPCS scheme and the proposed scheme. The probability that a cloud server returns the correct result is set to p={100%,80%,60%,40%}. In the original experiment, an incorrect response is replaced by a random value and is therefore an erroneous-result model. In the revised interpretation, such an erroneous result is counted as a Byzantine/invalid response only after Verify rejects it. Unavailable nodes, timeouts, delayed/straggling nodes, and network disconnections are analytically distinguished through the condition n−b−u≥q and are not conflated with random returned values.

We record the computational overhead required to complete one task as the metric for the existing erroneous-result experiment. This experiment evaluates the cost of obtaining q verified results when some returned values are invalid. It does not by itself establish resistance to coordinated Byzantine collusion or wireless-network disconnection. The security implications of cloud collusion are covered by the threat-model assumptions in Section 4.1, while straggler and timeout behavior should be evaluated in an end-to-end vehicular network testbed.

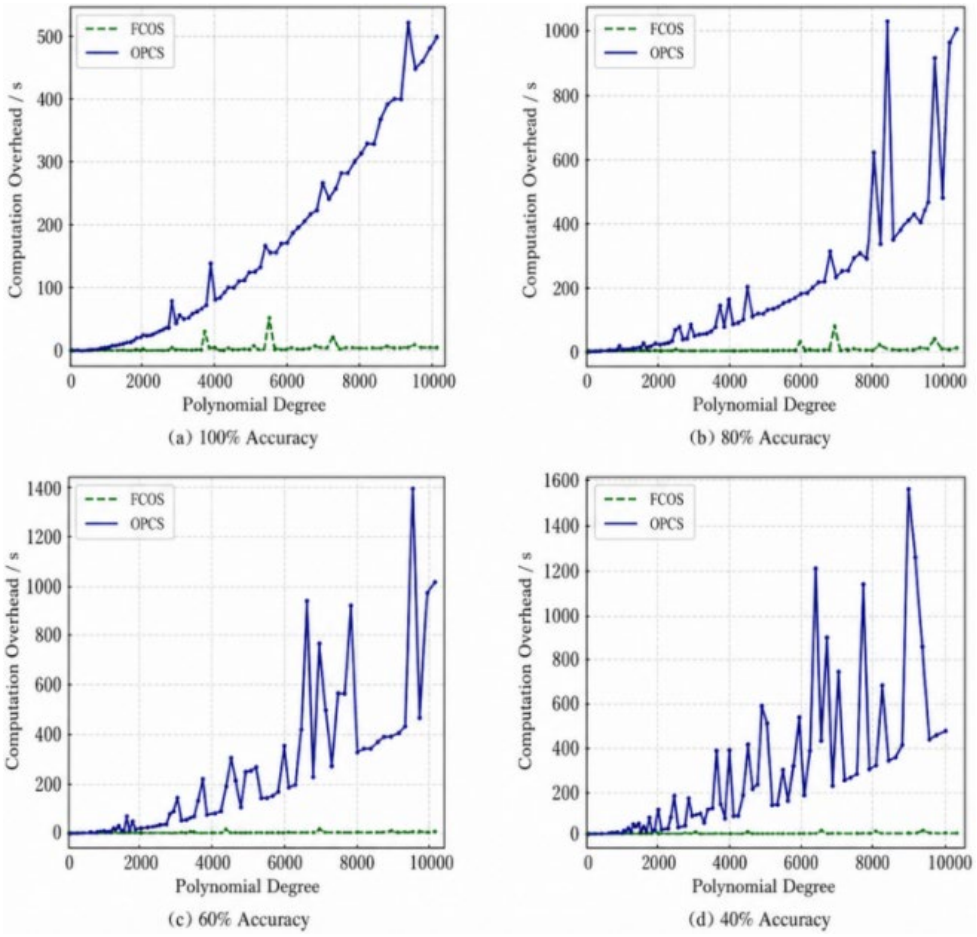


Figure 13. Comparative experimental results of fault tolerance

It can be observed that when computing nodes are not fully trusted, the latency curve of OPCS shows increasing fluctuations as the polynomial degree rises, while the curve of our scheme remains relatively smooth with no obvious

fluctuations. Specifically, under cloud server single-point failure rates of 0%, 20%, 40% and 60%, the proposed scheme reduces the average computation latency by 92.8%, 93.7%, 96.6% and 96.7% respectively compared with OPCS. This indicates that when computational errors exist, our scheme can offset the losses caused by faulty results with much less computation time, which confirms its strong fault tolerance capability.

Redundancy-Cost Trade-off and Deployment-Oriented Latency Accounting

The recovery threshold makes the redundancy-cost trade-off explicit. Let q be the minimum number of verified results and let each cloud server fail independently with probability p_f. The probability of successful recovery is $P_{\text{succ}}\left(n,q,p_f\right)=\sum_{j=q}^{n}\binom{n}{j}\left(1-p_f\right)^j p_f^{n-j}$. If cloud-compute cost is approximately proportional to the number of provisioned servers, a simple normalized resource cost is $C_{\text{norm}}=n/q$. Table 4 illustrates the trade-off for K=64 coefficients (q=8) and p_f=0.2. Increasing n from 8 to 16 doubles normalized compute-resource cost but increases recovery probability from 16.78% to 99.85%.

Table 4. Quantitative redundancy-cost trade-off for K=64, q=8 and pf=0.2

Cloud servers n	Max omission failures n−q	Recovery probability / %	Normalized cloud cost n/q
8	0	16.78	1.00
10	2	67.78	1.25
12	4	92.74	1.50
14	6	98.84	1.75
16	8	99.85	2.00

For an actual vehicular deployment, total service latency should be reported as $T_{\text{service}}=T_{V2R}+T_{R2C}+T_{\text{comp}}+T_{\text{verify}}+T_{\text{recover}}+T_{\text{chain}}+T_{\text{return}}+T_{\text{handover}}$, together with communication volume, concurrent-request throughput, RSU handover effects, and server-straggler statistics. ETSI MEC use cases treat V2X latency as a first-class requirement [40]. The current manuscript measures T_comp, contract execution/Gas, and recovery overhead at protocol/server level, but does not contain physical measurements for the remaining network terms. We therefore explicitly refrain from fabricating end-to-end values and narrow the implementation claim to a protocol prototype; a 5G/RSU emulation or field test with repeated trials, standard deviations/confidence intervals, and blockchain-finality configuration is required before claiming end-to-end vehicular latency.

Downstream IoV Application Mapping

A representative downstream use case is privacy-preserving perception-derived risk scoring. The vehicle can execute sensor preprocessing or a compact perception backbone locally and obtain bounded scalar features such as relative distance, time-to-collision surrogate, lane-deviation score, and object-confidence statistics. A service provider can define a bounded polynomial risk function over a scalar projection or a set of low-dimensional feature blocks; FCOS then protects the service coefficients with Paillier encryption and blinds the vehicle query before distributed cloud evaluation. This mapping keeps raw camera/LiDAR data on the vehicle and offloads only compact derived features. It should be viewed as a workload mapping rather than a complete private CNN-inference system. Full convolutional perception requires matrix/tensor operators and nonlinear activations; recent outsourced-computation work on depth-aware arithmetic circuits [32] provides a possible extension path.

8. Conclusion

Aiming at privacy leakage, high computation latency, and weak fault tolerance in cloud-blockchain computation offloading, this paper proposes FCOS, a privacy-preserving distributed outsourcing framework for bounded integer polynomial workloads. Distributed computation mitigates omission failures when at least $q=\sqrt{K}$ verified results are available; Paillier encryption protects function coefficients, and a vehicle-local finite-domain blinding factor protects requests against cloud-server coalitions under the stated trust boundary. The revised analysis clarifies exact-decoding and Paillier non-wraparound conditions, defines cloud-collusion and blockchain assumptions, and proves the Vandermonde rank condition for the recovery threshold. The current evaluation demonstrates protocol-level computation, verification/Gas, and erroneous-result tolerance, but does not constitute a physical end-to-end V2X deployment. Future work will extend matrix/set and arithmetic-circuit workloads, evaluate adaptive and cross-role collusion, and perform 5G/RSU field or network-emulated experiments covering transmission latency, handover, blockchain finality, concurrency, stragglers, communication volume, and confidence intervals.

Acknowledgements

This is the acknowledgement text. The authors have no acknowledgements to declare.

References

[1] Ahmed M, Raza S, Mirza MA, Aziz A, Khan MA, Khan WU, Li J, Han Z. A survey on vehicular task offloading: classification, issues, and challenges. J King Saud Univ Comput Inf Sci. 2022;34(7):4135-4162.

[2] Raza S, Wang S, Ahmed M, Anwar MR. A survey on vehicular edge computing: architecture, applications, technical issues, and future directions. Wirel Commun Mob Comput. 2019;2019:3159762.

[3] Li S, Xie Y, Shi M, Zheng X, Lu Y. Mobile edge computing empowered energy consumption optimization for multiuser power IoT networks. EAI Endorsed Trans Scalable Inf Syst. 2025;12(3). doi: 10.4108/eetsis.8678.
[4] Guan Y, Zheng H, Shao J, Lu R, Wei G. Fair outsourcing polynomial computation based on the blockchain. IEEE Trans Serv Comput. 2022;15(5):2795-2808.
[5] Sun PJ. Security and privacy protection in cloud computing: discussions and challenges. J Netw Comput Appl. 2020;160:102642.
[6] Yu X, Yan Z, Vasilakos AV. A survey of verifiable computation. Mob Netw Appl. 2017;22(3):438-453.
[7] Abbes S, Rekhis S. A blockchain-based solution for reputation management in IoV. In: Proceedings of the 2021 International Wireless Communications and Mobile Computing Conference (IWCMC); 28 June–2 July 2021; Harbin City, China. Piscataway, NJ: IEEE; 2021. p. 1129-1134.
[8] Rehman M, Khan ZA, Javed MU, Iftikhar MZ, Majeed U, Bux I, Javaid N. A blockchain-based distributed vehicular network architecture for smart cities. In: Proceedings of the Workshops of the 34th International Conference on Advanced Information Networking and Applications; 15–17 April 2020; Caserta, Italy. Cham: Springer; 2020. p. 320-331.
[9] Xiao T, Chen C, Pei Q, Song HH. Consortium blockchain-based computation offloading using mobile edge platoon cloud in Internet of Vehicles. IEEE Trans Intell Transp Syst. 2022;23(10):17769-17783.
[10] Javaid U, Sikdar B. A secure and scalable framework for blockchain-based edge computation offloading in social Internet of Vehicles. IEEE Trans Veh Technol. 2021;70(5):4022-4036.
[11] Narayan G, Haveri P, Rashmi B, Deewan Y. A framework for data provenance assurance in cloud environment using Ethereum blockchain. EAI Endorsed Trans Scalable Inf Syst. 2023;11(2). doi: 10.4108/eetsis.3536.
[12] Bhurani K, Dogra A, Agarwal P, Shrivastava P, Singh TP, Bhandwal M. Smart contracts for ensuring data integrity in cloud storage with blockchain. EAI Endorsed Trans Scalable Inf Syst. 2024;11(6). doi: 10.4108/eetsis.5633.
[13] Khan MMA, Sarwar HMA, Awais M. Gas consumption analysis of Ethereum blockchain transactions. Concurr Comput Pract Exp. 2022;34(4):e6679.
[14] Ozdemir A, Wahby R, Whitehat B, Boneh D. Scaling verifiable computation using efficient set accumulators. In: Proceedings of the 29th USENIX Security Symposium; 12–14 August 2020; Virtual Event. Berkeley, CA: USENIX Association; 2020. p. 2075-2092.
[15] Dorsala MR, Sastry VN, Chapram S. Fair payments for verifiable cloud services using smart contracts. Comput Secur. 2020;90:101712.
[16] Yao Y, Chen W, Chen X, Ding J, Pan S. A blockchain-based privacy-preserving scheme for vehicular trust management systems. In: Proceedings of the 2020 International Conference on Internet of Things and Intelligent Applications (ITIA); 27–29 November 2020; Zhenjiang, China. Piscataway, NJ: IEEE; 2020. p. 1-5.
[17] Xue J, Yan W. Edge computing communication privacy protection method based on federated learning algorithm. EAI Endorsed Trans Scalable Inf Syst. 2026;12(9).
[18] Song M, Sang Y, Zeng Y, Luo S. Blockchain-based secure outsourcing of polynomial multiplication and its application in fully homomorphic encryption. Secur Commun Netw. 2021;2021:9962575.
[19] Cordi C, Frank MP, Gabert K, Helinski C, Kao RC, Kolesnikov V, Ladha A, Pattengale N. Auditable, available and resilient private computation on the blockchain via MPC. In: Cyber Security, Cryptology, and Machine Learning (CSCML 2022), LNCS 13301; 30 June–1 July 2022; Be'er Sheva, Israel. Cham: Springer; 2022. p. 281-299.
[20] Kumari P, Kaur P. A survey of fault tolerance in cloud computing. J King Saud Univ Comput Inf Sci. 2021;33(10):1159-1176.
[21] Malik MK, Goel V, Swaroop A. A new hybrid COA-OOA based task scheduling and fuzzy logic approach to increase fault tolerance in cloud computing. EAI Endorsed Trans Scalable Inf Syst. 2024;11(6).
[22] Gennaro R, Gentry C, Parno B. Non-interactive verifiable computing: outsourcing computation to untrusted workers. In: Advances in Cryptology - CRYPTO 2010, LNCS 6223. 15–19 August 2010; Santa Barbara, CA, USA. Berlin, Heidelberg: Springer; 2010. p. 465-482.
[23] Ahmad H, Wang L, Hong H, Li J, Dawood H, Ahmed M, Yang Y. Primitives towards verifiable computation: a survey. Front Comput Sci. 2018;12(3):451-478.
[24] Fiore D, Gennaro R. Publicly verifiable delegation of large polynomials and matrix computations, with applications. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security; 16–18 October 2012; Raleigh, NC, USA. New York, NY: ACM; 2012. p. 501-512.
[25] Sun J, Zhu B, Qin J, Hu J, Ma J. Confidentiality-preserving publicly verifiable computation schemes for polynomial evaluation and matrix-vector multiplication. Secur Commun Netw. 2018;2018:5275132.
[26] Liu J, Bi J, Li M. Secure outsourcing of large matrix determinant computation. Front Comput Sci. 2020;14(6):146807.
[27] Li Y, Ghosh D, Gupta P, Mehrotra S, Panwar N, Sharma S. Prism: private verifiable set computation over multi-owner outsourced databases. In: Proceedings of the 2021 International Conference on Management of Data; 20–25 June 2021; Virtual Event, China. New York, NY: ACM; 2021. p. 1116-1128.
[28] Zhang Y, Deng RH, Liu X, Zheng D. Blockchain-based efficient and robust fair payment for outsourcing services in cloud computing. Inf Sci. 2018;462:262-277.
[29] Campanelli M, Gennaro R, Goldfeder S, Nizzardo L. Zero-knowledge contingent payments revisited: attacks and payments for services. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security; 30 October–3 November 2017; Dallas, TX, USA. New York, NY: ACM; 2017. p. 229-243.
[30] Dong C, Wang Y, Aldweesh A, McCorry P, Van Moorsel A. Betrayal, distrust, and rationality: smart counter-collusion contracts for verifiable cloud computing. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security; 30 October–3 November 2017; Dallas, TX, USA. New York, NY: ACM; 2017. p. 211-227.
[31] Huang H, Chen X, Wu Q, Huang X, Shen J. Bitcoin-based fair payments for outsourcing computations of fog devices. Future Gener Comput Syst. 2018;78(2):850-858.
[32] Vos J, Conti M, Erkin Z. Efficient batchable secure outsourced computation: depth-aware arithmetization of common primitives for BFV & BGV. In: Proceedings of the 34th USENIX Security Symposium (USENIX Security 25); 13–15 August 2025; Seattle, WA, USA. Berkeley, CA: USENIX Association; 2025. p. 8385-8402.

[33] Dong M, Zhang C, Bai Y, Chen Y. Efficient multi-party private set union without non-collusion assumptions. In: Proceedings of the 34th USENIX Security Symposium (USENIX Security 25); 13–15 August 2025; Seattle, WA, USA. Berkeley, CA: USENIX Association; 2025. p. 2005-2024.
[34] Xie Y, Wu Q, Fan P, Cheng N, Chen W, Wang J, Letaief KB. Resource allocation for twin maintenance and task processing in vehicular edge computing network. IEEE Internet Things J. 2025;12(15):32008-32021.
[35] Marcolla C, Sucasas V, Manzano M, Bassoli R, Fitzek FH, Aaraj N. Survey on fully homomorphic encryption, theory, and applications. Proc IEEE. 2022;110(10):1572-1609.
[36] Zhang J, Cheng X, Yang L, Hu J, Liu X, Chen K. SoK: fully homomorphic encryption accelerators. ACM Comput Surv. 2024;56(12):316.
[37] Mishra PK, Rathee D, Duong DH, Yasuda M. Fast secure matrix multiplications over ring-based homomorphic encryption. Inf Secur J Glob Perspect. 2021;30(4):219-234.
[38] Paillier P. Public-key cryptosystems based on composite degree residuosity classes. In: Advances in Cryptology - EUROCRYPT'99, LNCS 1592. 2–6 May 1999; Prague, Czech Republic. Berlin, Heidelberg: Springer; 1999. p. 223-238.
[39] Song B, Zhou D, Wu J, Yuan X, Zhu Y, Wang C. Protecting function privacy and input privacy in the publicly verifiable outsourcing computation of polynomial functions. Future Internet. 2023;15(4):152.
[40] ETSI. Multi-access edge computing (MEC); use cases and requirements. ETSI GS MEC 002 V3.2.1.; 2024.