mathematics

MDPI

Article

A Privacy-Preserving Quadratic Optimisation with Additive Homomorphic Encryption in Cyber-Physical Systems

Ying He [1], Yang Pu [1], Rui Ye [1] and Zhenyong Zhang [2,*]

[1] Operation Branch, Guizhou Power Grid Co., Ltd., Guiyang 550029, China
[2] College of Computer Science and Technology, Guizhou University, Guiyang 550025, China
* Correspondence: zhangzy@gzu.edu.cn

Abstract

In this paper, we propose a secure protocol to compute the quadratic optimisation problem under a three-party outsourcing architecture in the scenario of cyber-physical systems. To enable real-world implementation, we propose an encoding framework that uses a fixed-point expression and a truncated-mapping scheme to map real numbers into multiple data blocks, improving the protocol's efficiency. Based on this, we define the recovery operations for decryption, addition, and multiplication. Considering computations involving three parties to solve the quadratic optimisation problem, we thoroughly analyse privacy issues during the interaction process. Then, a secure protocol is developed by designing privacy-preserving addition, multiplication, and comparison protocols based on the additive homomorphic encryption scheme. The data blowup and "0"-privacy leakage problems are addressed specifically for the gradient descent process by designing a secure addition protocol for block data and a secure comparison protocol. The efficiency and security of the proposed protocol are formally analysed in depth. Finally, through intensive experiments, we demonstrate the efficiency and security of our protocol.

Keywords: homomorphic encryption; secure multi-party computation; cyber-physical system; optimisation problem

MSC: 93A99; 49M99

1. Introduction

Cyber-physical systems (CPSs) integrate advanced communication, computation, and control technologies to enable the real-time processing of big data collected from remote sensors. In this context, the outsourcing architecture plays a critical role in providing high-performance computing to resource-constrained edge users/devices. Among the outsourced computing tasks, solving optimisation problems is a common consideration. However, the outsourcing architecture is prone to cyberattacks from external attackers and disgruntled insiders due to the widely deployed open communication network and the commonly used devices, which inevitably have vulnerabilities [1]. Sensitive data from data sources (e.g., smartphones, smart wristbands, and smart glasses) can be stolen and misused for commercial purposes.

In recent years, due to the worldwide spread of COVID-19, there has been a growing trend of people sharing more data online and outsourcing their computations to the cloud or data centres. Many companies provide third-party online chatrooms (e.g., Zoom (https://zoom.us/) and VooV (https://voovmeeting.com/)) to hold meetings instead of face-to-face

Check for updates

Academic Editors: Jonathan Blackledge and Raymond Lee

Received: 17 February 2026
Revised: 14 April 2026
Accepted: 28 April 2026
Published: 1 May 2026

Copyright: © 2026 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.

conversations. Giant clouds have been established to support production optimisation and outsourced computations to Google (https://cloud.google.com/), Amazon (https://aws.amazon.com/cn/), etc. The rapidly increasing computational demands make conventional computers struggle to handle such tasks. More and more researchers, artists, and engineers buy computing power from the cloud.

However, the outsourced computation also reveals the weakness of exposing private data [2]. The cloud-outsourced framework provides efficient data processing but pays insufficient attention to data leakage. The leakage of enterprise data will lead to unpredictable losses. People's private information (e.g., locations, habits, and health conditions) and secret business data can be accessed by unauthorised persons. Therefore, it is urgent to address the challenge of achieving efficient computing in a cloud-outsourced computing framework to provide high-quality service while protecting users' privacy.

Encryption is a well-known approach for protecting sensitive data. The reference [3] proposed a cryptosystem that combines Chua's chaotic system with blockchain technology to secure medical images. Most secure computing protocols are built on encryption schemes that only guarantee the non-direct disclosure of sensitive data. However, essential data on the mathematical principles (e.g., critical parameters) for solving the problem and on the interaction processes are sometimes not protected. Some encryption schemes also do not support arithmetic operations on the encrypted data. For example, solving the quadratic optimisation problem using the cloud outsourcing framework involves numerous interactive, iterative operations on private data. The principle (e.g., the KKT conditions) for solving the problem is publicly known, so disclosing intermediate results may reveal private information. Therefore, it is difficult to rely solely on the encryption scheme to ensure secure computing in cloud outsourcing frameworks with complex operations. Differential privacy [4] is a variant of encryption that provides a formal measure of privacy; however, the added noise can reduce the data's utility. The optimal result is not achieved when users seek strong differential privacy to protect their private information, which may not be acceptable in scenarios that demand high accuracy, such as path planning in a large, unfamiliar city.

1.1. Related Work

CPSs are vulnerable to various cyber and/or physical security threats due to their heterogeneous nature, reliance on private and sensitive data, and large-scale deployment [5]. This paper focuses on solving the quadratic optimisation problem in CPSs while preserving privacy. The related work can be divided into three groups: homomorphic encryption (HE), differential privacy, and secure multi-party computation (SMPC).

Homomorphism was first proposed by Rivest et al. [6] in the 1970s, allowing third parties to perform calculations on encrypted data without decryption. Following ref. [6], many HE schemes have been developed [7,8]. Among them, the partially homomorphic encryption scheme [9–11] only allows one type of homomorphic operation, and the number of calculations is infinite. Still, most of them only support integers, and the plaintext space is limited. In contrast, the somewhat homomorphic encryption [12–15] can perform both addition and multiplication on encrypted data in a limited time. However, the increase in ciphertext size results in significant overhead from additional homomorphic operations. In 2009, Gentry's [7] fully homomorphic encryption scheme was the first that was both conceivable and feasible. However, due to its high computational overhead, it is not practical for use in real-world applications.

HE is also used in CPSs to enhance the security of the data exchange and transmission [16,17]. For HE computations in CPSs, Min et al. [18] proposed a parallel fully homomorphic encryption algorithm that supports operations on floating-point numbers. Kogiso et al. [19] designed an encrypted controller using RSA and ElGamal

schemes. To address the quantisation problem, Farokhi et al. [17] combined a quantisation strategy with the HE algorithm to design an encrypted output-feedback controller that stabilises the system. Recently, Alexandru et al. [20,21] proposed a secure cloud-based model predictive control system with encrypted data in a server–client architecture and developed cloud-based quadratic optimisation protocols using partially homomorphic encryption and projected gradient ascent on the dual problem. Zhang et al. [22] proposed an encryption-based state estimation algorithm using hybrid homomorphic encryption. Lin et al. [23,24] proposed an encrypted controller for a nonlinear system and analysed its stability using quantised data. More recent works include privacy-preserving frameworks for smart manufacturing systems using HE with trust boundaries [25], cloud-based privacy-preserving robust model predictive control [26], and a review of privacy-preserving encrypted control for CPSs [27]. A survey on securing CPSs with HE [28] further emphasises the need for efficient real-number support and iterative optimisation in resource-constrained environments. The HE methods perform well in protecting data security. However, they are not used alone in multi-party scenarios. Combining it with other privacy-preserving technologies can further enhance data security.

Differential privacy [4] is a variant of encryption that provides a formal framework for privacy preservation, assuming a noise distribution. By adding noise to the stochastic gradient descent computation, Han et al. [29] presented a distributed optimisation algorithm to protect data privacy and derived a bound on the suboptimality of the proposed algorithm. There are many related studies [30,31] on differential privacy for securing the computation of convex optimisation problems. Although data privacy can be protected to a certain extent, a trade-off between privacy and utility is necessary. Recent extensions apply differential privacy to cooperative control in CPSs [32], but they still incur accuracy loss in exact quadratic solutions.

SMPC allows multiple parties to jointly compute without exposing participants' sensitive data [12,33]. Zheng et al. [34] designed Helen, a system that enables multiple parties to train linear models without leaking private data. Helen can defend against a malicious adversary that can disrupt $m-1$ of m parties. Zheng et al. [35] proposed Cerebro, an end-to-end collaborative learning platform that enables parties to compute learning tasks without sharing plaintext data. Wang et al. [36] introduced a decentralised stochastic optimisation algorithm that enables multiple agents to cooperatively solve a global optimisation problem without a central coordinator. Recent SMPC advances include private outsourced model predictive control via secure multi-party computation in CPS [37,38], privacy-preserving cooperative control using cryptographic techniques [39], and secure multi-party computation for cloud-based control [40]. Challenges and future directions for SMPC on resource-constrained devices were surveyed in [41]. SMPC is a well-known computational framework for solving complex tasks. It is usually used to achieve the privacy goal by combining it with HE and obfuscation strategies (e.g., differential privacy).

While the works cited above have advanced privacy-preserving optimisation, four fundamental limitations remain for exact quadratic optimisation under a realistic three-party CPS outsourcing architecture:

- Failure to protect private data when the mathematical principle (KKT conditions) and system matrices $\mathbf{A}_{\mathcal{C}}, \mathbf{Q}$ are public. Existing HE/SMPC solutions assume intermediate dual variables or signs can be revealed safely; our privacy analysis (Section 3.2.2, Risks R1–R4) shows that knowledge of the sign of μ_i^* or the exact condition $\mu_i^* = 0$ directly recovers agents' private vectors $\mathbf{c}_{\mathcal{A}}$ or $\mathbf{b}_{\mathcal{A}}$ via the publicly known KKT Equations (9)–(11). No prior protocol simultaneously hides both sign and zero while supporting real-number recovery.

- Lack of efficient exact real-number support without ciphertext blow-up or precision loss in iterative gradient steps. Partial/fully homomorphic schemes either quantise data or suffer exponential growth; differential-privacy trades accuracy for privacy. Our fixed-point and truncated block-mapping encoding (Sections 4.1 and 4.2, Propositions 3–5) is the first to map real numbers into independent integer blocks while guaranteeing correct recovery after any number of secure additions and multiplications (Theorem 1).
- Absence of a lightweight 3-party secure comparison that hides both sign and zero information. Standard DGK-style comparisons leak the sign bit or require expensive fully-homomorphic primitives. Our novel C_{mp} protocol and SAP-BD use only additive Paillier operations plus minimal target-node assistance, achieving >5× speedup over DGK while completely masking the critical sign/zero leakage.
- No end-to-end protocol for the agent–cloud–target outsourcing model that provably resists all three realistic collusion scenarios ($\mathcal{L}_1 - \mathcal{L}_3$). Most prior outsourcing works ignore collusions or incur prohibitive overhead. Protocol π (Table 1) is the first to deliver formal security (Theorems 2–4) against up to $p-1$ colluding agents under each collusion type, while keeping cloud complexity linear in problem size and iteration count.

Table 1. Comparison of the related studies with our work.

Reference	Real-Number Support	Secure Comparison (3-Party Model)	Ciphertext Blow-Up Handling	Efficiency (vs. DGK/FHE)	Outsourcing Architecture
Alexandru et al. [20,21]	Partial	No (leaks sign)	Partial	Moderate	Cloud-client
Lu et al. [15,42]	No	No	No	High	Operator-agents
Farokhi et al. [17], Lin et al. [23,24]	Quantised	No	No	Moderate	Controller-focused
Differential privacy [29–32]	Yes	N/A (noise)	N/A	High (noise)	Distributed
Recent SMPC [37–40]	Yes	Partial	High overhead	Low	Cloud/multi-party
This work	Full (fixed-point + block mapping)	Yes	Yes (refresh + blocks)	High (additive HE only)	3-party

1.2. Our Contribution

This paper aims to address the above issues by protecting private data and solving the multi-party quadratic optimisation problem. First, because data generated in CPSs are usually real numbers, a new encoding method is proposed to convert real numbers to integers and truncate the resulting binary representation to support decentralised computation. Second, we define operations on the encoded data to correctly recover the data during decoding. Third, we thoroughly analyse the potential privacy issues for solving the quadratic optimisation problem under the outsourcing computing framework. Fourth, a secure interaction protocol is proposed that combines HE and SMPC techniques to ensure security, accuracy, and efficiency in solving the quadratic optimisation problem. In summary, our contributions are as follows:

- Based on the agent–cloud–target interaction framework, we conduct an exhaustive analysis of potential privacy leakage issues arising from the solution of the quadratic optimisation problem.
- We propose a real-data encoding method and a truncated-mapping scheme based on a fixed-point expression to improve computational efficiency and security in the interaction protocol.

- We propose a secure block data-based computation protocol π for solving the quadratic optimisation problem. A novel secure comparison protocol is proposed to prevent privacy leakage from sign and "0" information, achieving more than 5 times the speed of the traditional DGK protocol. The ciphertext blow-up problem is addressed for correctly recovering the encrypted and mapped result.
- We build an experimental scenario with multiple parties to evaluate the performance of the proposed protocol. The experimental results demonstrate the correctness of the proposed protocol and that it outperforms the existing work in terms of computation overhead.

The remainder of this paper is given as follows. Some prepared results are provided in Section 2. The problem setup is given in Section 3. Section 4 provides the secure protocol. In Section 5, we conduct a thorough complexity and security analysis of the proposed protocol. The experimental results are shown in Section 6. Section 7 discusses some critical issues related to the real-world implementation. The paper is concluded by the Section 8.

2. Preliminary

2.1. Homomorphic Encryption

In this study, we use the Paillier cryptosystem [16], a partially homomorphic encryption scheme that supports additive homomorphism and plaintext multiplication. We briefly introduce it below.

Setup: Select two big primes z and $\tilde{z}$ and let $N = z\tilde{z}$. The parameter λ is the least common multiple $lcm(z-1, \tilde{z}-1)$. The group base $g \in \mathbb{Z}_N^*$ satisfies $gcd(L(g^\lambda \ mod \ N^2), N) = 1$. The plaintext space is $\mathcal{M} = \mathbb{Z}_N$. The public and secret keys are $pk = (N, g)$ and $sk = \lambda$.

Encryption: Let $m \in \mathcal{M}$ be the message. We encrypt it by

$$c = Enc(pk, m, r) = g^m r^N \ mod \ N^2,$$

where r is an uniformly random chosen integer, $r \in \mathbb{Z}_N$.

Decryption: Ciphertext $c \in \mathbb{Z}_{N^2}^*$ is decrypted by the secret key and public key.

$$m = Dec(sk, c) = \frac{L(c^\lambda \ mod \ N^2)}{L(g^\lambda \ mod \ N^2)} \ mod \ N$$

where $L(u) = \frac{u-1}{N}$.

$$Enc(pk, m_1 + m_2) = c_1 \cdot c_2 \ mod \ N^2,$$
$$Enc(pk, m_1 \times m_2) = c_1^{m_2} \ mod \ N^2.$$

We denote $\cdot$ as the multiplication between two encrypted values. Furthermore, we define $\times$ as the multiplication between plaintext and encrypted data and omit the operation symbols when defining the multiplication between plaintexts.

2.2. Fixed-Point Expression

Given the number of bits used to represent the data, floating-point representation expands the data range but reduces precision. Calculating with floating-point numbers requires operations on both the mantissa and the exponent, and the result must be normalised. Here, we use signed fixed-point rational numbers in base 2 to represent the data.

Therefore, every number can be described using the binary expression, which is termed as (p, q) fixed-point expression

$$b_p b_{p-1} \cdots b_{q+1} b_q b_{q-1} \cdots b_1, \tag{1}$$

for given integers $p, q \in \mathbb{N}$ and $p > q$, $b_i \in \{0,1\}\ \forall\ i \in \{1,2,\ldots,p\}$. The numbers represented by the (p,q) fixed-point expression are contained in the set

$$Q(p,q) := \{x \in Q | x = -2^{p-q-1}b_p + \sum_{i=1}^{p-1} 2^{i-q-1}b_i\}. \tag{2}$$

Therefore, the set $Q(p,q)$ describes all the rational numbers that can be expressed by Equation (1), i.e., from -2^{p-q-1} to $2^{p-q-1} - 2^{-q}$. It is worth noting that in (6), the negative numbers are expressed in the complementary form.

3. Problem Setup

In CPS, many end devices collect field data but have limited computation power. To address this issue, the computation task is typically outsourced to a third-party cloud service. In this context, we propose a three-party involved computation protocol for solving the quadratic optimisation problem, which is given by (**P1**)

$$\textbf{P1:} \quad \min_{\mathbf{x}\in\mathbb{R}^n} \quad \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{c}_{\mathcal{A}}^T\mathbf{x} \tag{3}$$

$$s.t. \quad A_{\mathcal{C}}\mathbf{x} \preceq \mathbf{b}_{\mathcal{A}} \tag{4}$$

where $Q \in \mathbb{S}_{++}^n$, $\mathbf{c}_{\mathcal{A}} \in \mathbb{R}^n$ is a vector of coefficients, $A_{\mathcal{C}} \in \mathbb{R}^{m\times n}$ is the coefficient matrix of the inequality constraint, and $b_{\mathcal{A}} \in \mathbb{R}^m$ is a constant vector. The subscript means the party to which the parameter belongs. The optimal solution of problem **P1** is denoted by $\mathbf{x}^*$. The involved parties are described as follows.

Agent $(\mathcal{A})$**.** We denote the semi-honest agents as a set $\mathcal{A}$ with indices $\{1,2,\ldots,p\}$, corresponding to resource-constrained end devices in CPS. The subscript $\mathcal{A}$ in (3) and (4) means that the corresponding data (i.e., $\mathbf{c}_{\mathcal{A}}^T$ and $\mathbf{b}_{\mathcal{A}}$) are private for agents. Each agent owns the private data $Prv_{i\in\mathcal{A}} = (\mathbf{c}_i, \mathbf{b}_i)$, where $\mathbf{c}_i \in \mathbb{R}^{n_{c_i}}$ and $\mathbf{b}_i \in \mathbb{R}^{n_{b_i}}$ are part of the vectors $\mathbf{c}_{\mathcal{A}}$ and $\mathbf{b}_{\mathcal{A}}$, respectively, $n_{c_i}, n_{b_i} \geq 0$, and $\sum_{i=1}^p n_{c_i} = n, \sum_{i=1}^p n_{b_i} = m$. All agents keep the public key $pk_{\mathcal{T}}$ corresponding to the target node's private key.

Cloud $(\mathcal{C})$**.** The cloud is a party that has intensive computation resources. It collects encrypted data from agents and handles most of the computational tasks required to solve the quadratic optimisation problem. The cloud owns the public data $A_{\mathcal{C}}$ and Q, which are public information. It outputs the encrypted optimal result $[\mathbf{x}^*]$ to the target node. The cloud uses the public key $pk_{\mathcal{T}}$ to encrypt its private data.

Target node $(\mathcal{T})$**.** The target node obtains the encrypted optimal solution $[\mathbf{x}^*]$, a ciphertext that can be decrypted to the private data $\mathbf{x}^*$. The target node has the public key $pk_{\mathcal{T}}$ and a hidden private key $sk_{\mathcal{T}}$. The public key is published to the agents and the cloud.

Figure 1 illustrates the role and task of each party. The agents send their data to the cloud, which outputs the optimal result to the target node. For privacy reasons, all transmitted data is encrypted. In this paper, we assume a semi-honest model for each party. The semi-honest model is defined as follows.

Definition 1. ***(Semi-honest model).*** *A party is semi-honest if it executes the protocol correctly but has the purpose of learning or inferring the private information in the obtained data.*

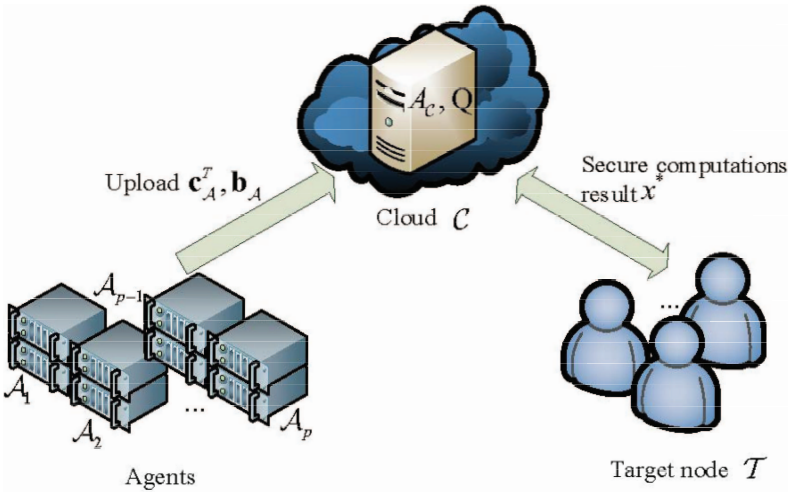


Figure 1. The architecture of the three-party involved computation.

Our purpose is to develop a secure computation protocol that uses encryption techniques to solve the quadratic optimisation problem with private data in the scenario described above. However, there are many challenges. First, most encryption schemes operate on positive integers, whereas the data in CPSs are real or even negative numbers. Second, data privacy involves not only the leakage of sensitive data but also the exposure of public information (e.g., mathematical principles, system parameters, and signs revealed by intermediate data). Third, solving the quadratic optimisation problem is an iterative process, which leads to ciphertext blow-up as the scaling factor and noise accumulate.

3.1. Threat Model

To our concern, the adversaries can be outside attackers and semi-honest insiders. The semi-honest insider follows the protocol step by step but tries to learn as much as possible from the input and output data. Typically, external attackers can overhear the communication channels among parties ($\mathcal{A}$, $\mathcal{C}$, $\mathcal{T}$), compromise the agents, and penetrate the cloud and the target node. Their purpose is to steal and infer the private data of the involved parties.

In this paper, our goal is to protect the agents' private data and the optimal solution $\mathbf{x}^*$. We discuss potential collusions between parties. The following cases are considered.

Collusion of agents ($\mathcal{L}_1$)**.** With the semi-honest assumption, some agents may share their data to infer the private data of other agents. Regarding the privacy goal, we aim to protect agents' private data when multiple agents collude.

Collusion of the agents and cloud ($\mathcal{L}_2$)**.** With the semi-honest assumption, the untrusted cloud may collude with some agents to infer the private data of other agents and the optimal result. Regarding the privacy goal, we aim to preserve agents' private data and to achieve an optimal solution when the cloud colludes with multiple agents.

Collusion of the agents and target node ($\mathcal{L}_3$)**.** With the semi-honest assumption, the curious target node may collude with some agents to infer the private data of other agents. Regarding the privacy goal, we aim to protect agents' private data when the target node colludes with multiple agents.

Note that we cannot achieve any privacy goal if the target node colludes with the cloud. In such cases, agents send all ciphertext corresponding to the private data to the cloud, where the target node can decrypt it. In the following, we present some definitions from the attackers' perspective that will be used in the security proof in Section 5.4.

Definition 2 (Computational indistinguishability)**.** *We call two distribution ensembles* $X \stackrel{def}{=} \{x_n\}_{n\in\mathbb{N}}$ *and* $Y \stackrel{def}{=} \{y_n\}_{n\in\mathbb{N}}$ *computational indistinguishability (denoted as* $X \stackrel{c}{\equiv} Y$*) if, for every probabilistic polynomial-time algorithm* $D : \{0,1\}^n \to \{0,1\}$ *and a negligible polynomially bounded* ε*,*

$$|\Pr(D(x_n) = 1) - \Pr(D(y_n) = 1)| < \varepsilon \tag{5}$$

where $|\cdot|$ *indicates the absolute value,* $\Pr(\cdot)$ *denotes the probability,* $x_n = x(\lambda, \delta)$ *and* $y_n = y(\lambda, \delta)$ *are the variables consisting of* λ *and* δ*,* $\lambda \in \mathbb{N}$ *is the security parameter and* δ *represents the input to the protocol.*

Definition 3 (Multi-party privacy w.r.t. semi-honest begavior)**.** *Let* $f : (\{0,1\}^*)^s \to (\{0,1\}^*)^s$ *be an s-ary functionality, where* $f_i(\delta_1, \delta_2, \ldots, \delta_s)$ *denotes the i-th element of* $f(\delta_1, \delta_2, \ldots, \delta_s)$*. Denote the inputs by* $\delta = (\delta_1, \delta_2, \ldots, \delta_s)$*. For* $\mathcal{U} = \{i_1, \ldots, i_t\} \subset S = \{1, \ldots, s\}$*, we let* $f_{\mathcal{U}}(\delta)$ *denote the subsequence* $f_{i_1}(\delta), \ldots, f_{i_t}(\delta)$*, which models a coalition of a number of parties. Let* Π *be an s-party protocol that computes* f*. The view of the i-th party during protocol* Π *runs with the input* δ*, denoted* $V_i^{\Pi}(\delta)$*, with* $(\delta_{s'}, coins, mes_1, \ldots, mes_t)_{s'\in S}$*, where coins is the outcome of the i'th party's internal coin tossed and* mes_t *represents the t-th message that the i-th party has received. We let the view of a coalition be denoted by* $V_{\mathcal{U}}^{\Pi}(\delta) = (\mathcal{U}, V_{i_1}^{\Pi}(\delta), \ldots, V_{i_t}^{\Pi}(\delta))$*. For a deterministic functionality* f*, we say that* Π *privately computes* f *if there exist simulators* $\mathcal{M}$*, such that, for every* $\mathcal{U} \subset S$*, it holds that, for* $\delta_t = (\delta_{i_1}, \delta_{i_2}, \ldots, \delta_{i_t})$ *[20],*

$$\{\mathcal{M}(\mathcal{U}, (\delta_t), f(\delta_t))\}_{\delta_t\in(\{0,1\}^*)^s} \stackrel{c}{\equiv} \{V_{\mathcal{U}}^{\Pi}(\delta)\}_{\delta_t\in(\{0,1\}^*)^s}$$

Definition 4 (View of the collision)**.** *Let* $\mathcal{J}$ *be the set of colluded parties of one of the collusion types* $\mathcal{L}_1$*,* $\mathcal{L}_2$*, and* $\mathcal{L}_3$ *with the semi-honest assumption. During the execution of* Π*, the view of the collusion is defined as*

$$V_{\mathcal{J}}^{\Pi}(\lambda, \delta) = (1^{\lambda}, \{\delta_j\}_{j\in\mathcal{J}}, \{r_j\}_{j\in\mathcal{J}}, mes_1^{\mathcal{J}}, \ldots, mes_{s'}^{\mathcal{J}})$$

where 1^{λ} *indicates the security parameter with the binary length is* λ*,* $\{\delta_j\}_{j\in\mathcal{J}}$ *and* $\{r_j\}_{j\in\mathcal{J}}$ *are the inputs and the random variables of j-th party in* $\mathcal{J}$ *and* $mes_{s'}^{\mathcal{J}}$ *is the* s'*-th message received by the j-th party.*

Note that we cannot achieve any privacy goal if the target node colludes with the cloud. In such cases, the agents send all ciphertexts corresponding to their private data to the cloud, where the target node (which possesses sk_T) can decrypt them directly. This architectural limitation is inherent to the three-party outsourcing model and is explicitly acknowledged as out-of-scope. In real CPS deployments, this collusion is considered unlikely because the cloud (typically a public or third-party provider) and the target node (usually the system operator or end-user) have conflicting business and regulatory incentives; moreover, many CPS regulations impose strict separation-of-duties and audit requirements that deter such collusion. Nevertheless, the semi-honest model already captures realistic insider threats that are the primary concern in practice.

3.2. Privacy Analysis

3.2.1. Solution to the Quadratic Optimisation Problem

In this paper, we do not consider the case where the primal problem has no feasible solutions or is unbounded; that is, we assume that **P1** has an optimal solution. Therefore, we can use the dual theory to solve it for which **P1** satisfies the strong duality. The dual of **P1** can also be defined as the quadratic optimisation problem **P2**.

$$\textbf{P2:} \quad \max_{\boldsymbol{\mu}\in\mathbb{R}^m} h(\boldsymbol{\mu}) = -\frac{1}{2}(A_{\mathcal{C}}^T\boldsymbol{\mu} + \mathbf{c}_{\mathcal{A}})^T Q^{-1}(A_{\mathcal{C}}^T\boldsymbol{\mu} + \mathbf{c}_{\mathcal{A}}) - \boldsymbol{\mu}^T\mathbf{b}_{\mathcal{A}} \tag{6}$$

$$s.t. \qquad \boldsymbol{\mu} \succeq \mathbf{0} \tag{7}$$

where $\boldsymbol{\mu}$ is the Lagrange multiplier. Considering the strong duality, the optimal result $\mathbf{x}^*$ of the primal problem **P1** can be derived from the optimal result $\boldsymbol{\mu}^*$ of the dual problem **P2**, that is

$$\mathbf{x}^* = -Q^{-1}(A_{\mathcal{C}}^T\boldsymbol{\mu}^* + \mathbf{c}_{\mathcal{A}}) \tag{8}$$

Moreover, the optimal result must satisfy the Karush–Kuhn–Tucker conditions:

$$Q\mathbf{x}^* + A_{\mathcal{C}}^T\boldsymbol{\mu}^* + \mathbf{c}_{\mathcal{A}} = \mathbf{0} \tag{9}$$

$$A_{\mathcal{C}}\mathbf{x}^* - \mathbf{b}_{\mathcal{A}} \preceq \mathbf{0} \tag{10}$$

$$\mu_i^*(\mathbf{a}_i\mathbf{x}^* - b_i) = 0, i = 1, 2, \ldots, m \tag{11}$$

where $\mathbf{a}_i$ is the i-th row of matrix $A_{\mathcal{C}}$, $\boldsymbol{\mu}_i^*$ is the i-th element of $\boldsymbol{\mu}^*$, b_i is the i-th element of $\mathbf{b}$, and $\mathbf{0}$ is a zero vector.

To solve the dual problem **P2**, we use the projected gradient ascent method:

$$\begin{aligned} \boldsymbol{\omega}(k) &= \nabla h(\boldsymbol{\mu}(k)) = -A_{\mathcal{C}}Q^{-1}(A_{\mathcal{C}}^T\boldsymbol{\mu}(k) + \mathbf{c}_{\mathcal{A}}) - \mathbf{b}_{\mathcal{A}}, \\ \boldsymbol{\mu}(k+1) &= max\{\mathbf{0}, \boldsymbol{\mu}(k) + \alpha\boldsymbol{\omega}(k)\}, \end{aligned} \tag{12}$$

where $\alpha > 0$ is the step size towards the gradient ascent direction, which is related to the convergence of $\boldsymbol{\mu}$, $\boldsymbol{\omega}(k)$ is the value of $\boldsymbol{\omega}$ at the k-th iteration, and $\boldsymbol{\mu}(k)$ is the value of $\boldsymbol{\mu}$ at the k-th iteration. As problem **P1** is strictly convex, the step size can be set as $\alpha = \frac{1}{\lambda_{max}(A_{\mathcal{C}}Q^{-1}A_{\mathcal{C}}^T)}$ to achieve the maximum convergence speed, where $\lambda_{max}(\cdot)$ denotes the maximum eigenvalue of the matrix.

3.2.2. Privacy Issue

In fact, our goal is to design a secure protocol to solve problem **P2** (and, in the meantime, problem **P1**) by preserving the private data owned by different parties as described in Section 3. Therefore, it needs to discuss privacy issues in the agent–cloud–target architecture during computation. Specifically, we should analyse the potential privacy leakage from the computations (9)–(12).

The privacy issues are analysed in two periods: the execution period and after the execution period. During the execution period, the protocol needs to compute (12), in which the private data are $\boldsymbol{\mu}(k), k \in \{1, 2, \ldots, K\}$, $\mathbf{c}_{\mathcal{A}}$, and $\mathbf{b}_{\mathcal{A}}$. These private data are known only to the parties specified in protocol π and cannot be inferred by any other party (we will provide a detailed analysis). The value of $\boldsymbol{\mu}$ is a secret against all parties except the initial value $\boldsymbol{\mu}(0)$ for the cloud. After the execution period, the encrypted optimal solution $[\mathbf{x}^*]$ is kept secret from the parties except the target node.

Except for the public information, since $[\mathbf{b}_{\mathcal{A}}]$ is ciphertext during the solution of **P1**, it is not possible for a semi-honest adversary without the private key $sk_{\mathcal{T}}$ to directly infer any private information from (10). Therefore, we focus on the equalities (9) and (11). The privacy issues we discuss arise from the assumption that the adversary can infer the sign of $\boldsymbol{\mu}_i^*$. Following the assumption, is it possible to derive some privacy data about the agent?

Case 1: $\boldsymbol{\mu}^* = \mathbf{0}$

That is, the inequality constraint does not work for problem **P1**; we have $\mathbf{q}_i\mathbf{x}^* + c_i = 0$, where $\mathbf{q}_i$ is the i-th row of Q and c_i is the i-th element of $\mathbf{c}_{\mathcal{A}}$. There is a risk of data privacy leakage.

R1: In the case where the target node knows that all $\boldsymbol{\mu}_i^* = \mathbf{0}, i = 1, 2, \ldots, m$ and knows $\mathbf{q}_i, i = 1, 2, \ldots, m$, the target node can obtain the private data c_i owned by the agent.

Case 2: $\boldsymbol{\mu}_i^* = 0$

Suppose that the target node knows partial value of $\boldsymbol{\mu}^*$, denoted by $\mathbf{0}_{\boldsymbol{\mu}^*}$ ($\boldsymbol{\mu}_i^* \in \mathbf{0}_{\boldsymbol{\mu}^*}$ and $\boldsymbol{\mu}_i^* = 0$). The following privacy risk should then be considered.

R2: Since we have $\mathbf{q}_i\mathbf{x}^* + (\mathbf{a}_i^c)^T\boldsymbol{\mu}^* + c_i = 0$ and (8), where $\mathbf{a}_i^c$ is i-th column of matrix $A_{\mathcal{C}}$ and c_i is the i-th element of the vector $\mathbf{c}_{\mathcal{A}}$. In the case where the target node knows $\mu_i^* = 0$ and colludes with agents, the target node can not only get $\mathbf{c}_{\mathcal{A}}$ but also can infer the subspace of $\mathbf{b}_{\mathcal{A}}$ and $\boldsymbol{\mu}^*$ from (10) and (11).

Proposition 1. *Let n_o denote the size of $\mathbf{0}_{\boldsymbol{\mu}^*}$. The set of colluded agents with the target node is $\mathcal{J}^{\mathcal{L}_3}$. Then, the privacy leakage risk **R2** happens if and only if $n_o + \sum_{i\in\mathcal{J}^{\mathcal{L}_3}} n_{c_i} \geq m$.*

Proof. Considering that the private data $\boldsymbol{\mu}^*$ and $\mathbf{c}_{\mathcal{A}}$ are unknown variables, then we can split Equation (9) into n equations. When the collusion involving the agent and the target node is informed from $\boldsymbol{\mu}^* = \mathbf{0}$ that the size of $\mathbf{0}_{\boldsymbol{\mu}^*}$ is n_o, the target node can infer the private data c_i with $\mathbf{q}_i\mathbf{x}^* + c_i = 0$ immediately. Due to the only unknown variable in $\mathbf{q}_i\mathbf{x}^* + (\mathbf{a}_i^c)^T\boldsymbol{\mu}^* + c_i = 0$ is $\boldsymbol{\mu}^*$(partially unknown) and c_i. There are $m - n_o$ unknown elements for $\boldsymbol{\mu}^*$ and n unknown elements for $\mathbf{c}_{\mathcal{A}}$. In total, there are $n + m - n_o$ unknown variables. If the colluded agent provide the private data $\mathbf{c}_i$, then there are remaining $m + n - n_o - \sum_{i\in\mathcal{J}^{\mathcal{L}_3}} n_{c_i}$ unknown variables. Therefore, we must guarantee that $m + n - n_o - \sum_{i\in\mathcal{J}^{\mathcal{L}_3}} n_{c_i} \leq n$, that is, $n_o + \sum_{i\in\mathcal{J}^{\mathcal{L}_3}} n_{c_i} \geq m$. The sufficiency is obvious. We complete the proof. □

Case 3: $\boldsymbol{\mu}_i^* > 0$

According to (11), it can be derived that we have $\mathbf{a}_i\mathbf{x}^* - b_i = 0$ if $\boldsymbol{\mu}_i^* > 0$, which can cause the following risk of privacy leakage.

R3: Since the target node already knows $\mathbf{x}^*$ and $\mathbf{a}_i$ is public, then the target node can infer the private data b_i owned by the agent if he/she knows $\boldsymbol{\mu}_i^* > 0$.

Case 4: $\mathbf{a}_i\mathbf{x}^* - b_i < 0$

Since $\mathbf{a}_i\mathbf{x}^* - b_i < 0$, it can be derived that $\mu_i = 0$ according to (11), which causes the following risk of privacy leakage when multiple agents collude with each other.

R4: If a collusion $\mathcal{J}^{\mathcal{L}_1}$ of multiple agents jointly compute $\mathbf{a}_j\mathbf{x}^* - b_{j\in\mathcal{J}^{\mathcal{L}_1}}$ and gain the knowledge that partial elements of $\boldsymbol{\mu}^*$ equal 0 (assume the subset is $\mathbf{0}_{\boldsymbol{\mu}^*}$), then this collusion can infer the private data $\boldsymbol{\mu}^*$ and $c_{i\in\bar{\mathcal{J}^{\mathcal{L}_1}}}$ and $b_{i\in\bar{\mathcal{J}^{\mathcal{L}_1}}}$, where $\bar{\mathcal{J}^{\mathcal{L}_1}}$ denotes the set of honest agents except those in $\mathcal{J}^{\mathcal{L}_1}$.

Proposition 2. *Let n_o' denote the size of $\mathbf{0}_{\boldsymbol{\mu}^*}$. The set of colluded agents with the target node is $\mathcal{J}^{\mathcal{L}_1}$. Then, the privacy leakage risk **R4** happens only if $n_o' + \sum_{i\in\mathcal{J}^{\mathcal{L}_1}} n_{c_i} \geq m$.*

Proof. We omit the proof due to space limitations because the idea is the same as the proof of Proposition 1. □

From above, it is important to prevent the adversary from gaining any knowledge of the sign of $\boldsymbol{\mu}^*$ and $\boldsymbol{\mu}_i^* = 0$. To address this issue, we propose a secure protocol to solve the dual problem of **P1** considering the risks **R1**–**R4**.

4. Solution in the Ciphertext Domain

In this section, some details are provided to solve the problem **P1** in the ciphertext domain. First, we develop a mapping scheme to generalise the protocol to real numbers. Second, to protect users' private data, we design a truncated-mapping scheme for raw data and a secure comparison protocol. Third, a secure overall protocol is provided to protect agents' private data.

4.1. Encoding Scheme

The mapping scheme is designed to address quantisation error and ciphertext blow-up problems caused by iterative computations on encrypted data. We represent the data using the fixed-point expression as (1). Since only the data belonging to the integer plaintext space of Paillier can be encrypted and used for homomorphic operations, we encode these fixed-point values as integers using the following method. Let $f_{p,q} : Q(p,q) \to \mathbb{Z}_{2^l}$, then we have

$$y = f_{p,q}(x) = \begin{cases} 2^q x \ mod \ 2^l & \text{if } x \geq 0; \\ (2^l - f_{p,q}(-x)) \ mod \ 2^l & \text{if } x < 0, \end{cases} \tag{13}$$

where l is sufficiently large and x is a real number. If we denote the range of $f_{p,q}(x)$ as $Z(l)$, then $Z(l)$ is a finite integer set. Given the secure protocol for solving the quadratic optimisation problem, the data used are the results of $Z(l)$-level operations, such as addition and multiplication. When the protocol terminates, the results should be recovered to the original fixed-point expression $Q(p,q)$. As we cannot determine whether the encrypted data is "positive" or "negative" [note that all values involved in the encryption are positive, but some "positive" values represent negative values using the above mapping $f_{p,q}(x)$, it is challenging for us to correctly map the result output from the protocol back to $Q(p,q)$, let alone after a sequence of operations for executing the protocol. In this paper, we conduct a comprehensive analysis to accurately recover the result from the secure protocol. The following analytical results are presented to support the data recovery.

Proposition 3. *Let x be a number represented by the (p,q) fixed-point expression, $y = f_{p,q}(x)$. Then, if $l \geq p$, the recovery of y (i.e., the inverse image of y) is*

$$x' = f_{p,q}^{-1}(y) = \begin{cases} 2^{-q}(y \ mod \ 2^l) & if \ y \leq 2^{l-1}; \\ 2^{-q}((y - 2^l) \ mod \ 2^l) & if \ y > 2^{l-1}, \end{cases} \tag{14}$$

where the recovered result x' is denoted by (p,q) fixed-point value and $x' = x$.

Proof. The main idea of the proof is that the parameters selected above will not cause conflict during the recovery process. According to the mapping function (13), as $x \in [-2^{p-q-1}, 2^{p-q-1} - 2^{-q}]$, we have $y \in [0, 2^{p-1} - 1] \cup [2^l - 2^{p-1}, 2^l - 1]$. Because $k \geq p$, we have $2^{p-1} - 1 < 2^{l-1}$ and $2^l - 2^{p-1} \geq 2^{l-1}$. It is a one-to-one map from non-negative values in $Q(p,q)$ to the image set $[0, 2^{p-1} - 1]$ and also a one-to-one map from negative values in $Q(p,q)$ to the image set $[2^l - 2^{p-1}, 2^l - 1]$. Thus, the inverse function $f_{p,q}^{-1}(y)$ can correctly recover y to x, i.e., $x' = x$. □

Proposition 4. *Let x_1 and x_2 be two numbers represented by the (p,q) fixed-point expression and $x = x_1 + x_2$. Let $y_1 = f_{p,q}(x_1)$, $y_2 = f_{p,q}(x_2)$, and $y = y_1 + y_2$. Then, if $l \geq p + 1$, the recovered result of y is*

$$x' = f_{p,q}^{-1}(y) = \begin{cases} 2^{-q}(y \ mod \ 2^l) & if \ y \leq 2^{l-1}; \\ 2^{-q}((y - 2^l) \ mod \ 2^l) & if \ y > 2^{l-1}, \end{cases} \tag{15}$$

where the recovered result x' is represented by the (p,q) fixed-point expression and $x' = x = x_1 + x_2$.

Proof. We consider the four sign combinations explicitly. (i) $x_1 \geq 0, x_2 \geq 0 \Rightarrow y \in [0, 2^p - 2]$. Since $l \geq p+1$, we have $2^p - 2 < 2^{l-1}$. (ii) $x_1 \geq 0$, $x_2 < 0$, $x \geq 0 \Rightarrow y \in [0, 2^{p-1} - 2]$ (still $< 2^{l-1}$). (iii) $x_1 \geq 0$, $x_2 < 0$, $x < 0 \Rightarrow y \in [2^l - 2^p + 1, 2^l - 1]$ (hence $y > 2^{l-1}$). (iv) $x_1 < 0, x_2 < 0 \Rightarrow y \in [2^l - 2^p, 2^l - 2]$ (again $> 2^{l-1}$). In all cases, the inverse function (14) selects the correct branch and, because the intervals are disjoint and cover exactly the image of $f_{p,q}$, we obtain $x' = x$. □

Proposition 5. *Let x_1 and x_2 be two numbers denoted by the (p,q) fixed-point expression and $x = x_1 x_2$. Let $y_1 = f_{p,q}(x_1)$, $y_2 = f_{p,q}(x_2)$, and $y = y_1 y_2$. Then, if $l \geq 2p - 1$, the recovered result of y is*

$$x' = f_{p,q}^{-1}(y) = \begin{cases} 2^{-2q}(y \bmod 2^l) & \text{if } y \leq 2^{l-1}; \\ 2^{-2q}((y - 2^l) \bmod 2^l) & \text{if } y > 2^{l-1}, \end{cases} \tag{16}$$

where the recovered result x' is represented by the (p,q) fixed-point expression and $x' = x = x_1 x_2$.

Proof. As $y \in [0, 2^{p-1} - 1] \cup [2^l - 2^{p-1}, 2^l - 1]$, if $x_1 > 0, x_2 < 0$ and $x < 0$, then $y \in [2^l - 2^{2p-2} + 2^{p-1}, 2^l - 1]$; if $x_1 \geq 0, x_2 \geq 0$ and $x > 0$, then $y \in [0, 2^{2p-2} - 2^p + 1]$; if $x_1 < 0, x_2 < 0$ and $x > 0$, then $y \in [1, 2^{2p-2}]$. Because $l \geq 2p - 1$, $2^{2p-2} \leq 2^{l-1}$, $2^l - 2^{2p-2} + 2^{p-1} > 2^{l-1}$, the inverse function $f_{p,q}^{-1}(y)$ can correctly recover y to x, i.e., $x' = x$. □

To summarise, the main idea of the proof is that we do not cause overflows or conflicts when operating on the encoded results. Based on these results, the value of l should be used to design the system to ensure correct recovery of the final result. As the operands are matrices in this work, the longest multiplication and addition operations are related to the dimensions of the matrix. Therefore, l is associated with the number of columns and rows of the involved matrix.

4.2. Truncate-Mapping Scheme

In Section 4.1, we have transformed the raw data into positive integers. Here, we propose a truncated-mapping scheme to protect signed data from inference, accounting for the risks **R1**–**R4**.

First, based on the binary string of the fixed-point expression, the truncation is conducted as follows:

$$\begin{aligned} x &= b_l \dots b_i \dots b_1 \\ &= b_{l_\beta}^{\beta} \dots b_1^{\beta} | \dots | b_{l_i}^{i} \dots b_1^{i} | \dots | b_{l_1}^{1} \dots b_1^{1} \\ &= \Delta_\beta | \dots | \Delta_i | \dots | \Delta_1 \end{aligned} \tag{17}$$

where $b_{l_\beta}^{\beta}$ is the sign bit, $b_{l_i}^{i}$ is the l_i-th bit of the i-th block by truncating the binary string, and $\Delta_i = b_{l_i}^{i} \dots b_1^{i}$ ($\sum_{i=1}^{\beta} l_i = l$). The notation β is the number of block data that the binary string is truncated. The truncate-mapping scheme works as follows:

$$\begin{aligned} &\Delta_i = 2^{-\sum_{j=1}^{i-1} l_j}(x \bmod 2^{\sum_{i=1}^{i} l_i} - x \bmod 2^{\sum_{i=1}^{i-1} l_j}), \\ &\sum_{i=1}^{\beta} l_i = l, (i \in \{1, 2, \dots, \beta\}), \end{aligned} \tag{18}$$

where l_i is the number of bits of the data block Δ_i. For example, the binary string of 23 is 10111, which can be truncated into $\underbrace{1}_{\Delta_3}||\underbrace{01}_{\Delta_2}||\underbrace{11}_{\Delta_1}$. It means that $l = l_3 + l_2 + l_1 = 1 + 2 + 2 = 5$. Then, the corresponding data block is $\Delta_3||\Delta_2||\Delta_1 = 1||1||3$.

4.3. Secure Addition Protocol over Block Data

We transform computations over signed private data into operations over block data to address privacy issues caused by sign leakage. To ensure that operations on the data block can be correctly recovered without "overflows", we develop the secure addition protocol over block data (**SAP-BD**) as follows. The notation $[\cdot]$ represents the encrypted data. The main idea of Algorithm 1 is to perform the required computations by mapping the raw data into shares of varying bit lengths using the truncate-mapping scheme. The notations $(\Delta)_{data}$ and $(\Delta)_{DATA}$ are binary strings with l bits and have β blocks. We use integer division and modulo operations to handle carry from block-level additions. Randomness is introduced into the computations to obfuscate the data while preserving privacy.

As shown in Algorithm 1, given $[(\Delta)_{data}]$ and $[(\Delta)_{DATA}]$, **SAP-BD** outputs $[data + DATA] = [(\Delta)_{data}] \cdot [(\Delta)_{DATA}]$. Before the protocol runs, the cloud generates random numbers and performs computations on them to obfuscate the private data used later (Lines 1–3). Considering the carry bit generated by operations on block data, the cloud blurs the computation result of the block data and sends it to the target node (Line 5). Then, the target node performs modulo (Line 7) and integer division (Line 8) on the blinded data and returns the results to the cloud (Line 9) to determine whether the operation over the block data has a carrying bit.

Notice that the target node can infer a subset of the private data $(\Delta_i^{data+DATA} + Coin_{i-1})$ from the sign information of $(\gamma - R)$, especially when $r'_i = i'2^{l_i} (i' \in \mathbb{Z}^+)$. Therefore, the cloud generates a random bit *RandomCoin* to obfuscate γ and R for preserving the sign information of $(\gamma - R)$ (Lines 11–17). Moreover, due to $0 \leq (\Delta_i^{data+DATA} + Coin_{i-1}) \leq 2^{l_i+1} - 1$, the target node knows $(r'_i + \Delta_i^{data+DATA} + Coin_{i-1})$ and $|((r'_i + \Delta_i^{data+DATA} + Coin_{i-1}) \bmod 2^{l_i}) - (r'_i \bmod 2^{l_i})|$; the cloud uses another new random number r_i to blind the value of $(\gamma - R)$ (Line 18).

It is worth noting that the random data r_i is positive, which does not alter the sign information of the value in the cloud (Line 18). The target node decrypts $[AB]$ upon receipt and returns $[RE]$, which indicates the relationship between γ and R to the cloud (Lines 19–25). Thereafter, the cloud recovers RE based on the random bit *RandomCoin* (Lines 26–31). It is based on the fact that if $\gamma < R$, then $\gamma' > R'$ and we can infer that $\Delta_i^{data} + \Delta_i^{DATA} > 2^{l_i}$ and vice verse. Since $0 \leq \Delta_i^{data} + \Delta_i^{DATA} \leq 2^{l_i} + 1 - 2$ and $0 \leq \Delta_i \leq 2^{l_i} - 1$, we get $\Delta_i = \Delta_i^{data} + \Delta_i^{DATA}$ (Lines 32–33).

Finally, the computation results over the block data $(\Delta)_{data}$ and $(\Delta)_{DATA}$ are recovered according to (18) by the cloud (Line 35). More generally, considering the case where $(\mathbb{Z}_1 \times [data]) \cdot (\mathbb{Z}_2 \times [DATA])$, in which $\mathbb{Z}_{1,2} \in \mathbb{Z}^+$, we only need to replace 2^{l_i} with $\mathbb{Z}_1\mathbb{Z}_2 2^{l_i}$ in **SAP-BD**, leaving the rest steps unchanged.

Algorithm 1 Secure addition protocol over block data

Input: $[(\Delta)_{data}]$, $DATA$, pk
Output: The result of $[data + DATA]$
Cloud: Choose $\mathbf{r} := \{r | r_i \in \mathbb{Z}_{2^{p-1}}, i = 1, \ldots, \beta\}$, $\mathbf{r}' := \{r' | r'_i \in \mathbb{Z}_{2^{p-1}}, i = 1, \ldots, \beta\}$ get $[(\Delta)_{DATA}]$, choose a random bit $RandomCoin$ and set the carry bit $Coin_0 = 0$
Cloud: Compute $R' = r'_i // 2^{l_i}$ and get $[R']^{-1}$
Cloud: Compute $R = r'_i \bmod 2^{l_i}$ and get $[R]^{-1}$
for For $i = 1$ to β **do**
    Cloud: Send $[\Delta_{i'}] = [r'_i] \cdot [\Delta_i^{data+DATA}] \cdot [Coin_{i-1}]$ to **Target** node # where $[\Delta_i^{data+DATA}] = [\Delta_i^{data}] \cdot [\Delta_i^{DATA}]$
    Target: Decrypt $[\Delta_{i'}]$
    Target: Compute $\gamma = \Delta_{i'} \bmod 2^{l_i}$
    Target: Compute $\gamma' = \Delta_{i'} // 2^{l_i}$
    Target: Send $[\gamma]$, $[\gamma']$ to **Cloud**
    Cloud:
    if $RandomCoin = 1$ **then**
        Set $[AAA] = [\gamma]$
        Set $[BBB] = [R]$
    else
        Set $[AAA] = [R]$
        Set $[BBB] = [\gamma]$
    end if
    Cloud: Send $[AB] = [r_i/2] \cdot (r_i \times ([AAA] \cdot [BBB]^{-1}))$ to **Target**
    Target: Decrypt $[AB]$
    if $AB > 0$ **then**
        Set $RE = 0$
    else
        Set $RE = -1$
    end if
    Target: Send $[RE]$ to **Cloud**
    Cloud:
    if $RandomCoin$ = 1 **then**
        Set $[RE]$ = $[RE]$
    else
        Set $[RE] = (-1) \times ([RE] \cdot [1])$
    end if
    Cloud: Compute $[Coin_i] = [\gamma'] \cdot [R']^{-1} \cdot [RE]$
    Cloud: Compute $[\Delta_i] = [Coin_{i-1}] \cdot [\Delta_i^{data+DATA}] \cdot (2^{l_i} \times [Coin_i]^{-1})$
end for
Cloud: Compute $[data + DATA] = \sum_{i=1}^{\beta} 2^{\sum_{j=1}^{i-1} l_j} \times [\Delta_i], (2^{\sum_{i=1}^{0} l_j} = 1, i = 1, \ldots, \beta)$

4.4. Secure Comparison Protocol C_{mp}

According to our analysis, the privacy risks caused by the sign of $\boldsymbol{\mu}^*$ are significant during the comparison process. To address this issue, we propose a simple yet effective privacy-preserving comparison scheme, C_{mp}, based on Paillier and the encoding and mapping schemes in Table 2.

The main idea of C_{mp} is as follows. Given two data α' and β', the cloud holds the block data corresponding to $(\alpha' - \beta')$. The block data Δ_β of $(\alpha' - \beta')$ contains a bit that can indicate the sign of the subtraction, i.e., the left-most bit $b_{l_\beta}^{\beta}$ of the block data Δ_β. We have $b_{l_\beta}^{\beta} = 0 \Leftrightarrow \alpha' \geq \beta'$. Therefore, if $b_{l_\beta}^{\beta} = 0$, then $\Delta_\beta \in [0, 2^{l_\beta - 1} - 1]$ and if $b_{l_\beta}^{\beta} = 1$, then $\Delta_\beta \in [-1, 1 - 2^{l_\beta - 1}]$. The protocol of C_{mp} is given in Table 2.

Notice that the idea for addressing the issue that the bit $[t]$ indicates the comparison result is the same as the carry bit $[Coin_{i-1}]$ that reflects the sign information in Algorithm **SAP-BD**, i.e., $t = Coin_{i=\beta}$. The main difference is that they play different roles in solving the optimisation problem. In Algorithm **SAP-BD**, $Coin_i$ $(i = 1, \ldots, \beta - 1)$ is used to represent

the rounding of the block data to guarantee that the operated result over block data can be recovered correctly, while t $(Coin_{i=\beta})$ indicates the sign information of $(\alpha' - \beta')$.

The secure comparison protocol C_{mp} works as follows. For the first interaction, before sending $\Delta_{\beta'}$ to the target node, the cloud sampled the random numbers uniformly from $\mathbb{Z}_{2^{p-1}}$, computing R' and R for blinding the block data Δ_β. After receiving $\Delta_{\beta'}$, the target node performs modulo and integer division to assist the cloud in determining the range of Δ_β and sends the resulting ciphertext to the cloud. In the second interaction, the cloud randomises the order of the variable involved in the computation using random bits to prevent the target node from obtaining the sign information of the computed result. Additionally, the cloud blinds the generated random number and sends the blinded result to the target node, similar to the process described in Lines 11–18 of Algorithm **SAP-BD**. For the same reason as Lines 19–32 in Algorithm **SAP-BD**, the target decrypts the received ciphertext and returns $[RE]$ to the cloud, which computes $[RE]$ based on the random bits chosen in the second round and obtains $[t]$.

Parameter specification. All protocols (including Cmp and SAP-BD) use the following concrete parameters: security parameter $\lambda = 2048$ bits (standard Paillier modulus $N = 4096$ bits), fixed-point precision $(p, q) = (64, 32)$, number of blocks $\beta = 2$, and block sizes $l_i = 32$ bits for $i = 1, \ldots, \beta$ (with $l_\beta = 32$ for the sign-bit block in C_{mp}). These values satisfy the conditions of Propositions 3–6 while keeping the plaintext domain safely inside $\mathbb{Z}_N$. In practice, q controls the final precision (error $\leq 2^{-q}$), β trades off between communication and parallelism, and λ can be increased to 3072 bits for higher security without affecting asymptotic complexity.

Table 2. Secure comparison protocol C_{mp}.

Cloud	**Target Node**
Randomly choose: $r \in \mathbb{Z}_N$, $r' \in \mathbb{Z}_{2^{p-1}}$; Compute: $R' = r'//2^{l_\beta}$ and get $[R']$ Compute: $R = r' \bmod 2^{l_\beta}$ and get $[R]$ Compute: $\Delta_{\beta'} = [r'] \cdot [\Delta_\beta]$	
Send $\Delta_{\beta'} \longrightarrow$	
	Decrypt: $\Delta_{\beta'}$ Compute: $z = \Delta_{\beta'} \bmod 2^{l_\beta}$ $= (r' + \Delta_\beta) \bmod 2^{l_\beta}$ Compute: $z' = \Delta_{\beta'}//2^{l_\beta}$ Encrypt: $z \xrightarrow{pk}$ **[z]**, $z' \xrightarrow{pk}$ **[z']**
$\longleftarrow$ Send $[z], [z']$	
Choose a random bit *RandomCoin* If *RandomCoin* $= 1$: $[AAA] = [z]$, $[BBB] = [R]$, else: $[AAA] = [R]$, $[BBB] = [z]$ Compute: $[AB] = [r/2] \cdot (r \times ([AAA] \cdot [BBB]^{-1}))$	
Send $[AB] \longrightarrow$	
	Decrypt: $[AB]$ If $AB > 0$: Set $RE = 0$ else: Set $RE = -1$
$\longleftarrow$ Send $[RE]$	
If *RandomCoin* $= 1$: $[RE] = [RE]$ else: $[RE] = (-1) \times ([RE] \cdot [1])$ Compute: $[t] = [z'] \cdot [R']^{-1} \cdot [RE]$	

*4.5. Main Protocol π for Solving **P1***

In this section, the main protocol π is proposed to solve problem **P1**, while accounting for privacy considerations. The detailed interaction process is provided in Table 3. Note that the length of each block of data (i.e., l_i) is public information. In the preparation phase, the agents apply the encoding and mapping schemes described in Sections 4.1 and 4.2, respectively, to the input data, then send the encrypted data to the cloud. Afterwards, the cloud runs Algorithm **SAP-BD** with the target node for K iterations to compute $[\bar{\omega}(k+1)] = [\mu(k)] \cdot (\alpha \times [\theta(k+1)])$. It is worth noting that in the secure updating phase, we not only have to make sure that the target node does not know any information about the comparison result (especially the sign information) but also that the cloud cannot get any information of the updated variable during the comparison between $[\bar{\omega}(k)]$, $[0]$, and $[u_{k+1}]$. Finally, the cloud sends the final result to the target node, which obtains the optimal solution by decrypting and recovering it.

Table 3. Main protocol π.

Agents	**Cloud**	**Target Node**
Map: $\bar{\mathbf{b}}_{\mathcal{A}} = \bar{\mathbf{b}}_{\mathcal{A}}, \bar{\mathbf{c}}_{\mathcal{A}} = \bar{\mathbf{c}}_{\mathcal{A}}$ $\bar{\mathbf{b}}_{\mathcal{A}} = Q_{p,q}(\bar{\mathbf{b}}_{\mathcal{A}}), \bar{\mathbf{c}}_{\mathcal{A}} = Q_{p,q}(\bar{\mathbf{c}}_{\mathcal{A}})$ $\bar{\mathbf{b}}_{\mathcal{A}} = f_{p,q}(\bar{\mathbf{b}}_{\mathcal{A}}), \bar{\mathbf{c}}_{\mathcal{A}} = f_{p,q}(\bar{\mathbf{c}}_{\mathcal{A}})$ $\bar{\mathbf{b}}_{\mathcal{A}} = (\Delta)_{\bar{\mathbf{b}}_{\mathcal{A}}}, \bar{\mathbf{c}}_{\mathcal{A}} = (\Delta)_{\bar{\mathbf{c}}_{\mathcal{A}}}$ Encrypt: $\bar{\mathbf{b}}_{\mathcal{A}} \xrightarrow{pk} [\bar{\mathbf{b}}_{\mathcal{A}}], \bar{\mathbf{c}}_{\mathcal{A}} \xrightarrow{pk} [\bar{\mathbf{c}}_{\mathcal{A}}]$		
Send: $[\bar{\mathbf{b}}_{\mathcal{A}}]$ and $[\bar{\mathbf{c}}_{\mathcal{A}}] \longrightarrow$		
	Compute: $\tilde{A}_1 = -A_C Q^{-1} A_C^T$, $\tilde{A}_2 = -A_C Q^{-1}$, $\tilde{A}_3 = -Q^{-1} A_C^T$, $\tilde{A}_4 = Q^{-1}$ Map: $\bar{A}_1 = f_{p,q}(Q_{p,q}(\tilde{A}_1))$, $\bar{A}_2 = f_{p,q}(Q_{p,q}(\tilde{A}_2))$, $\bar{A}_3 = f_{p,q}(Q_{p,q}(\tilde{A}_3))$, $\bar{A}_4 = f_{p,q}(Q_{p,q}(\tilde{A}_4))$ Initialisation: $\bar{\boldsymbol{\mu}}_0 = (\Delta)_{f_{p,q}(Q_{p,q}(\bar{\boldsymbol{\mu}}_0))} \xrightarrow{pk} [\bar{\boldsymbol{\mu}}_0], \bar{\alpha} = Q_{p,q}(\tilde{\alpha}), \alpha = f_{p,q}(\bar{\alpha})$ For $0 \le k < K$: Compute: $[\bar{\omega}(k)] = (\bar{A}_1 \times [\bar{\boldsymbol{\mu}}(k)]) \cdot (2^q \bar{A}_2 \times [\bar{\mathbf{c}}_{\mathcal{A}}]) \cdot (2^{3q} \times [\bar{\mathbf{b}}_{\mathcal{A}}]^{-1})$ Randomisation: $\mathbf{r} := \{r \mid r_i \in \mathbb{Z}_{2^{p-1}}, i = 1, \ldots, \beta\}$ $\mathbf{r}' := \{r' \mid r'_i \in \mathbb{Z}_{2^{p-1}}; i = 1, \ldots, \beta\}$	
	Compute: $[\bar{\boldsymbol{\mu}}(k+1)] = [\bar{\boldsymbol{\mu}}(k)] \cdot (\alpha \times [\bar{\boldsymbol{\omega}}(k)])$ Secure computation of the dual variable: target node and Cloud interactive perform the protocol **SAP-BD**	
	Secure update of the variable Compare: $[\bar{\mu}(k+1)]$ and $[0]$, where $\bar{\mu}(k+1)$ is the element of $\bar{\boldsymbol{\mu}}(k+1)$ Run the secure comparison protocol C_{mp}, get $[t]$ that indicate the sign of corresponding data of block data Choose different random data $r'_1, r'_2 \in \mathbb{Z}_{2^{p-1}}$ and a random bit $Rcoin$; If $Rcoin = 0$: $[AA] = [\bar{\mu}(k+1)] \cdot [r'_1]$, $[BB] = [t]$ $[CC] = [1] \cdot [t]^{-1}$, $[DD] = [0] \cdot [r'_2]$ Else $Rcoin = 1$: $[AA] = [1] \cdot [t]^{-1}$, $[BB] = [0] \cdot [r'_2]$ $[CC] = [\bar{\mu}(k+1)] \cdot [r'_2]$, $[DD] = [t]$ Send: $[AA], [BB], [CC], [DD] \longrightarrow$	
		Decrypt: $[AA], [BB], [CC], [DD]$ Compute: $SS = AACC + BBDD$
	$\longleftarrow$ Send: $SS \xrightarrow{pk} [SS]$	
	Compute $[\bar{\mu}(k+1)] = [SS] \cdot (r_1' \times ([t] \cdot [1]^{-1})) \cdot (r_2' \times [t]^{-1})$, $\bar{\mu}(k+1)$ is the element of $\bar{\boldsymbol{\mu}}(k+1)$ Secure update process that ensures $u_{k+1} = \max\{\omega(k), 0\}$, $\omega(k)$ is the element of $\boldsymbol{\omega}(k)$ Compute $[\bar{\mathbf{x}}^*] = (\bar{A}_3 \times [\boldsymbol{\mu}_{k+1}]) \cdot (2^q \bar{A}_4 \times [\bar{\mathbf{c}}_{\mathcal{A}}])$	
	Send: $[\mathbf{x}^*] \longrightarrow$	
		Decrypt: $[\bar{\mathbf{x}}^*] \xrightarrow{sk} \bar{\mathbf{x}}^*$ Recover: $\mathbf{x}^* = 2^{-2q} f_{p,q}^{-1}(\bar{\mathbf{x}}^*)$

Note: $\tilde{x}$ indicates the real number; $\bar{\cdot}$ denotes the mapped data; $\rightarrow$ indicates that the direction of the arrow indicates the direction of data transmission. $t \in \{0, 1\}$ that satisfies $(t = 1) \Leftrightarrow (\bar{\omega}(k) < 0)$; $\xrightarrow{pk}$ and $\xrightarrow{sk}$ indicate the operation of encryption and decryption, respectively. $f_{p,q}(\cdot)$ denotes the mapped result of the data based on Equation (1); $Q_{p,q}(\cdot)$ denotes the mapped result of the data based on Equation (2).

5. Performance Analysis of Protocol π

In this section, we first analyse the computational complexity and communication overhead of protocol π. Then, we investigate the correctness of protocol π. Lastly, we formally prove the security of protocol π.

5.1. Computational Complexity of Protocol π

The computational complexity of protocol π is computed by counting the number of operations of encryption (Enc), modular multiplication (MM), modular exponentiation (ME), modular multiplicative inverse (MI), decryptions (Dec), truncate-mapping (TM), and block data carry operation (BCO). According to the homomorphism of Paillier, the additive homomorphic operation corresponds to the modular multiplication over ciphertexts, and the multiplication corresponds to modular exponentiation. To execute protocol π, the computations of each party $(\mathcal{A}, \mathcal{C}, \mathcal{T})$ are shown in Table 4.

Table 4. Computations of each party to execute protocol π.

Parties	$\mathcal{A}$	$\mathcal{C}$	$\mathcal{T}$
Enc	$(m+n)\beta$	$m\beta(2K+1)+6K$	$4m\beta K+K$
MM		$(m\beta(m+n+9)+8)K$	
ME		$(m+n+4)m\beta K+2K$	
MI		$(m+n+5)\beta K+3K$	
TM	$(m+n)\beta$	$m\beta$	
BCO		$m\beta K$	$m\beta K$
Dec			$2(m\beta+2)K$

According to protocol π, the agents $(\mathcal{A})$ encrypt β block data generated from performing the truncate-mapping scheme after encoding the private data $b_{\mathcal{A}}$ and $c_{\mathcal{A}}$ for $\beta(m+n)$ times. To get the solution $\mathbf{x}^*$, the cloud and the target node interactively run K iterations of protocol π, each of which requires running the Algorithm **SAP-BD** m times and the protocol C_{mp} l times. Since the computations over the input data are mapped as β block data, we should run the Algorithm **SAP-BD** $m\beta K$ times and the cloud needs to be executed 2 times for Enc, 11 times for MM, 3 times for ME, 5 times for MI, and $m\beta$ times for BCO. It is worth noting that the TM needs to be run only once, i.e., $m\beta$ times throughout the process, independent of the number of iterations. Moreover, the target node needs to perform Enc 4 times and Dec 2 times. In the update phase, the cloud needs to perform Enc 6 times, MM 8 times, ME 2 times, and MI 3 times, while the target node only needs to perform Enc 1 time and Dec 4 times for each iteration. Overall, we can see that computational complexity increases with system size, block size, and the number of iterations required to solve the optimisation problem.

The computational complexity is now analysed by counting the dominant cryptographic operations per party: encryption (Enc), decryption (Dec), modular multiplication (MM), modular exponentiation (ME), modular inverse (MI), truncate-mapping (TM), and block-carry operations (BCO). The secure comparison C_{mp} requires only 11 operations on the cloud side (vs. DGK's typical 57–70 operations dominated by modular exponentiations), which directly accounts for the observed >5× speedup while using the same additive homomorphic primitive.

5.2. Communication Complexity of Protocol π

The communication complexity of protocol π is calculated from the data exchanged among the parties. Supposing the bit size of ciphertext is ξ, then the communication complexity of protocol π is

$$C_{\pi}^{com} = \beta(m + n + 6mk) + 5mK)\xi \tag{19}$$

Actually, the number of interactions of protocol π is closely related to the number of constraints, the involved agents, and the coefficients of problem **P1**. The communication overhead is larger if problem **P1** is more complex. Additionally, communication overhead is affected by the number of bits used to represent the input data and the truncate-mapping parameter. The communication overhead can be divided into the following cases:

Agents to Cloud (Initial Phase): Agents upload encrypted block data $[\bar{\mathbf{b}}_{\mathcal{A}}]$ and $[\bar{\mathbf{c}}_{\mathcal{A}}]$, totaling $O(l(m+n)\xi)$ bits. This is a one-time transfer, independent of K, and scales linearly with problem size. For multiple agents (p agents), this is distributed as $O(l(n_{c_i} + n_{b_i})\xi)$ per agent, minimising load on resource-constrained CPS devices.

Cloud–Target Interactions (Iterative Phase): The bulk of communication occurs here, at $O(l(16mK + 4nK)\xi)$ bits. For each iteration, we have the following:

- Secure addition (SAP-BD, Algorithm 1): Involves $O(lm)$ block-wise exchanges for carry propagation and randomisation, with the target decrypting and re-encrypting $o(7m + 2n)$ elements.
- Secure comparison (Cmp) and update: For each of m dual variables, the cloud sends four ciphertexts ($[AA]$, $[BB]$, $[CC]$, $[DD]$) to the target ($O(lm\xi)$ bits per iteration), and the target responds with encrypted sums $[SS]$ ($O(lm\xi)$ bits). This is batched across m elements to reduce the number of rounds.
- Overall per iteration: Approximately 2–4 round-trips between cloud and target, transmitting $O(l(m^2 + mn))$ bits due to matrix–vector operations (e.g., $\bar{A}_1 \times \bar{\boldsymbol{\mu}}^k$).

Cloud to Target (Final Phase): Transmission of $[\bar{\mathbf{x}}^*]$, at $O(ln\xi)$ bits—a negligible one-time cost.

The complexity grows linearly with K (iterations for convergence), $\frac{m}{n}$ (problem scale), and l (precision via blocks). In practice, K is typically 50–200 for quadratic optimisation convergence (per experiments in Section 6), and l is fixed small (e.g., 32) via truncation to avoid blow-up.

In CPS deployments (e.g., smart grids or autonomous vehicles), networks often exhibit latency (10–500 ms round-trip) and bandwidth constraints (1–100 Mbps), influenced by wireless links, geographic distribution, or edge-cloud setups. Here, we discuss how our protocol's communication affects end-to-end performance, using analytical models and empirical data.

Latency Impact: The protocol requires $O(K)$ sequential iterations, each with 2–4 cloud–target round-trips (e.g., for SAP-BD and Cmp). In low-latency environments (e.g., data centres with RTT $<$ 10 ms), this adds negligible overhead; our experiments (Section 6.2.3) show total runtimes of 12–45 s for K = 50–200, $m = n = 100$. However, in high-latency networks (e.g., satellite links at 500 ms RTT or IoT over cellular at 100 ms), the $O(K)$ rounds could dominate, increasing runtime by 5–10$\times$ (e.g., from 20 s to 100–200 s). To mitigate, we note that operations are batchable: all m comparisons can be performed in parallel rounds, reducing the effective rounds to $O(K)$ rather than $O(Km)$. Compared to decentralised multi-party schemes, which require $O(Kp)$ rounds across p agents, our centralised cloud–target focus cuts latency-sensitive interactions by 50–70% in distributed CPS.

Bandwidth Impact: Total data volume is $O(lmK\xi)$ bits (e.g., 10–100 MB for $K = 100$, $m = 100$, $\xi = 2048$, $l = 32$), feasible on broadband but challenging on low-bandwidth links

(e.g., <1 Mbps in remote sensors). At 10 Mbps, transfer time is <10 s; at 1 Mbps, it rises to 80–800 s, potentially stalling real-time CPS tasks. The block-based design helps: Truncation keeps l small, limiting growth, and randomisation adds minimal overhead. Compared to pure HE schemes, which transmit full matrices per iteration ($O(Kmn\xi)$), our protocol reduces bandwidth by 30–50% through optimised subprotocols (e.g., Cmp is 5× faster and lighter than DGK). In constrained networks, adaptive strategies like compressing ciphertexts (via Paillier variants) or reducing K (via faster-converging optimisers) could halve overhead.

5.3. Proof of the Correctness of Protocol π

In this section, we will analyse the parameter settings for guaranteeing the correctness of protocol π. The most significant issue is ciphertext blowup triggered by performing iterative computations on encrypted data. It is worth noting that the random numbers used in protocol π are not affected by the iterative computation; they are generated independently and can be offset in each iteration. In other words, we can solve the ciphertext blowup by setting the sum of the random numbers and the maximum value of the result produced by the computation within the plaintext space indicated by the cryptographic scheme. In the following, we will analyse the maximum value of the result produced in solving problem **P2**. A worst-case is that we always have $\boldsymbol{\mu}(k+1) = \boldsymbol{\mu}(k) + \alpha\boldsymbol{\omega}(k)$ without considering the case $\boldsymbol{\mu}(k) = \mathbf{0}$. However, since we convert the negative to positive for encryption in protocol π. We should ensure that the unencrypted data is within the plaintext space specified by the cryptosystem. Hence, we denote the value of the element in vector $\boldsymbol{\mu}(k)$ by a function $F(k,l)$ with the variable $k = 0, 1, \ldots, K$ and the parameter l, which is the maximum value that can be calculated by $\boldsymbol{\mu}(k+1) = \boldsymbol{\mu}(k) + \alpha\boldsymbol{\omega}(k)$ in the form of $f_{p,q}(\cdot)$ after k iterations and satisfies

$$\begin{aligned} &\max_{k\in\{0,1,\ldots K\}} \lceil F(k,l) \rceil \\ &\quad s.t. \quad F(k,l) < N \end{aligned} \tag{20}$$

We illustrate the above result as follows.

$$\begin{aligned} &F(0,l) = (2^l - 1) \\ &F(1,l) = F(0) + \alpha[n^2 m(2^l-1)^3 F(0) + \ldots \\ &\ldots + n^2(2^l-1)^4 + (2^l-1)^4] \\ &\ldots \\ &F(K,l) = F(K-1) + \alpha[n^2 m(2^l-1)^3 F(K-1) + \ldots \\ &\ldots + n^2(2^l-1)^4 + (2^l-1)^4] \end{aligned}$$

Furthermore, we set $\mathrm{A} = 2^l - 1,\ \mathrm{B} = n^2 m,\ \mathrm{Y} = \mathrm{BA}^4 + n^2\mathrm{A}^4 + \mathrm{A}^4$. Then,

$$\begin{aligned} &F(0,l) = \mathrm{A} \\ &F(1,l) = \mathrm{A} + \alpha\mathrm{Y} \\ &F(2,l) = \mathrm{A} + 2\alpha\mathrm{Y} + \mathrm{A}^3\mathrm{B}\alpha^2\mathrm{Y} \\ &F(3,l) = \mathrm{A} + 3\alpha\mathrm{Y} + 3\mathrm{A}^3\mathrm{B}\alpha^2\mathrm{Y} + \mathrm{A}^6\mathrm{B}^2\alpha^3\mathrm{Y} \\ &F(4,l) = \mathrm{A} + 4\alpha\mathrm{Y} + 6\mathrm{A}^3\mathrm{B}\alpha^2\mathrm{Y} + 4\mathrm{A}^6\mathrm{B}^2\alpha^3\mathrm{Y} + \mathrm{A}^9\mathrm{B}^3\alpha^4\mathrm{Y} \\ &F(5,l) = \mathrm{A} + 5\alpha\mathrm{Y} + 10\mathrm{A}^3\mathrm{B}\alpha^2\mathrm{Y} + 10\mathrm{A}^6\mathrm{B}^2\alpha^3\mathrm{Y} + \ldots \\ &\ldots + 5\mathrm{A}^9\mathrm{B}^3\alpha^4\mathrm{Y} + \mathrm{A}^{12}\mathrm{B}^4\alpha^5\mathrm{Y} \\ &\ldots \\ &F(K,l) = \mathrm{AC}_K^0 + \sum_{i=1}^{K} \alpha\mathrm{Y}(\mathrm{A}^3\mathrm{B}\alpha)^{i-1}\mathrm{C}_K^i \\ &F(K,l) = (2^l-1)C_K^0 + \sum_{i=1}^{K} (\alpha(n^2 m(2^l-1)^4 + \ldots \\ &\ldots + n^2(2^l-1)^4 + (2^l-1)^4)((2^l-1)^3 n^2 m\alpha)^{i-1} C_K^i) \end{aligned}$$

According to protocol π, we iteratively compute $[\bar{\boldsymbol{\mu}}(k+1)] = [\bar{\boldsymbol{\mu}}(k)] \cdot (\alpha \times [\bar{\boldsymbol{\omega}}(k)])$ for K iterations where $[\bar{\boldsymbol{\omega}}(k)] = (\bar{A}_1 \times [\bar{\boldsymbol{\mu}}(k)]) \cdot (2^q \bar{A}_2 \times [\bar{\mathbf{c}}_A]) \cdot (2^{3q} \times [\bar{\mathbf{b}}_A]^{-1})$. Further-

more, we can decompose $\bar{\omega}(k+1)$ into three parts: $-A_C Q^{-1} A_C^T \bar{\boldsymbol{\mu}}(k)$, $-2^q A_C Q^{-1} \mathbf{c}_{\mathcal{A}}$, $-2^{3q}\mathbf{b}_{\mathcal{A}}$. In the first iteration, we can get $F(1,l) = F(0,l) + \alpha(n^2 m(2^l-1)^3(F(0,l) + n^2(2^l-1)^4 + (2^l-1)^4)$, where the maximum value of the element in $-A_C Q^{-1} A_C^T \bar{\boldsymbol{\mu}}(k)$ can be expressed by $n^2 m(2^l-1)^3(F(0,l))$ and $F(0,l) = (2^l-1)$, and the maximum value of the element in $-2^q A_C Q^{-1} \mathbf{c}_{\mathcal{A}}$ and $2^q \bar{\mathbf{b}}_{\mathcal{A}}$ is $n^2(2^l-1)^4$ and $(2^l-1)^4$, respectively. After K iterations, the maximum value of the element in $\bar{\boldsymbol{\mu}}(k+1)$ is $F(K,l) = F(K-1) + \alpha(n^2 m(2^l-1)^3(F(K-1)) + n^2(2^l-1)^4 + (2^l-1)^4)$. To perform decryption correctly when using the Paillier encryption scheme, we must ensure that $F(K,l) < N$.

Moreover, the scaling factor (i.e., 2^q) used in the mapping scheme will significantly enlarge the involved data, causing overflow as the data is outside the range of the $Q(p,q)$ expression. Therefore, we refresh the ciphertexts in our proposed protocol at each iteration by multiplying the term with scaling factors that will be cancelled later, as referred to in (21), to avoid the ciphertext blowup problem. We give the following theorem to guarantee the correct recovery of the output of the π.

Theorem 1. *Assuming that the involved random numbers in protocol π are all positive, the output of protocol π is denoted by $\mathbf{x}^*$. x^* is the element of $\mathbf{x}^*$. $\overline{x}^*$ is the element of the mapped vector $\overline{\mathbf{x}}^*$. If $F(K,p-1)+1 < F(K,l)$, x^* is recovered as follows:*

$$\begin{aligned} x^* &= 2^{-2q} f_{p,q}^{-1}(\overline{x}^*) \\ &= \begin{cases} 2^{-3q}(\overline{x}^* \bmod 2^l) & if\ \overline{x}^* \le 2^{l-1}; \\ 2^{-3q}((\overline{x}^* - 2^l) \bmod 2^l) & if\ \overline{x}^* > 2^{l-1}. \end{cases} \end{aligned} \tag{21}$$

Proof. To understand the proof, we need to consider the matrix calculation from the angle of element operations. According to (9)–(12), we should iteratly compute $\boldsymbol{\mu}(k+1) = \boldsymbol{\mu}(k) + \alpha\boldsymbol{\omega}(k)$ for K iterations. Concerning the impact of the scaling factor, there are 2^q expansions on each data point involving the computation. We denote the value of the element in $\bar{\boldsymbol{\mu}}(k+1)$ by $\bar{\mu}$. If $\bar{\mu} \ge 0$, then it belongs to the range $[0, F(K,p-1)]$. Otherwise, $\bar{\mu} < 0$, it belongs to the range $[F(K,p-1)+1, F(K,l)]$. Therefore, it is required that $F(K,p-1)+1 < F(K,l)$ must be held for the element in $\overline{\mathbf{x}}^*$ to be correctly recovered. From $\bar{\mu}(k+1) = \bar{\mu}(k) + \alpha\bar{\omega}(k)$, where $\bar{\mu}(k)$ is expanded 2^q and the $\bar{\omega}(k)$ concerning the multiplication between elements significantly expands the data, especially in $\bar{\boldsymbol{\omega}}(k)$ involving the addition of three tuples, i.e., $\bar{A}_1 \times [\bar{\boldsymbol{\mu}}(\boldsymbol{k})]$, $\bar{A}_2 \times [\bar{\mathbf{c}}_A]$ and $[\bar{\mathbf{b}}_A]^{-1}$, where $\bar{A}_1 \times [\bar{\boldsymbol{\mu}}(\boldsymbol{k})]$ is the multiplication of the four elements in the form $f_{p,q}(\cdot)$; that is, there is 2^{4q} expansion. Similarly, there are 2^{3q} and 2^q expansions for $\bar{A}_2 \times [\bar{\mathbf{c}}_A]$ and $[\bar{\mathbf{b}}_A]^{-1}$. Due to the computation of $\bar{\boldsymbol{\omega}}(\boldsymbol{k})$ concerning the addition of the three tuples, we extend each of the elements in $\bar{\boldsymbol{\omega}}(\boldsymbol{k})$ to 2^{4q} to facilitate the recovery of the computed result. Based on $\alpha = \frac{1}{\lambda_{max}(A_C Q^{-1} A_C^T)}$, the eigenvalues are expanded 2^{-3q} times and are all positive, which partially offsets the ciphertext expansion caused by the scaling factor. Thus, the elements in $\bar{\boldsymbol{\omega}}(\boldsymbol{k})$ expands by only 2^q. Similarly, it follows from $\overline{\mathbf{x}}^*$ that $\bar{A}_3 \times [\boldsymbol{\mu}_{\boldsymbol{k+1}}]$ expands to 2^{3q} and $\bar{A}_4 \times [\bar{\mathbf{c}}_A]$ expands to 2^{2q}. We expand the elements in $\overline{\mathbf{x}}^*$ to 2^{3q}. Hence, the target can recover the x^* from (21). □

It should be noted that the security parameter must meet $\lambda \ge$ to avoid the ciphertext blowup problem.

Remark 1. *In each iteration of protocol π, after the secure update of the dual variable, the cloud performs a ciphertext-refresh operation by homomorphically multiplying the updated ciphertexts with the modular inverses of the accumulated scaling factors (specifically $[2^{-2q}]$ for $\bar{\mu}^{(k+1)}$ and $[2^{-3q}]$ for the terms involved in the eventual computation of $\bar{x}^*$). These operations exploit the additive homomorphism of the Paillier cryptosystem and are executed entirely in the encrypted*

domain. This refresh step exactly cancels the scaling introduced by the fixed-point encoding and matrix–vector multiplications in the gradient computation, ensuring that the scaling factor at the start of each new iteration remains constant. As a result, upon termination of the protocol, the target node can correctly recover the optimal solution using the formula in Equation (21) without any residual scaling or overflow, as long as the condition $F(K, p-1)+1 < F(K, l)$ holds.

Proposition 6. *Let $\boldsymbol{x}^*$ denote the exact optimal solution of problem (**P1**) and let $\hat{\boldsymbol{x}}^*$ be the solution recovered by protocol π after K iterations. Then,*

$$\|\boldsymbol{x}^* - \hat{\boldsymbol{x}}^*\|_2 \leq 2^{-q} \cdot (1 + K\alpha \cdot \kappa),$$

where α is the step size and $\kappa = \|A_C Q^{-1} A_C^T\|_\infty + \|Q^{-1} A_C^T\|_\infty$ is a constant depending only on the problem data. Moreover, overflow is completely prevented, and the bound can be made arbitrarily small by choosing a sufficiently large q, independently of the number of iterations K (provided l is chosen as in Theorem 1).

Proof. The mapping $f_{p,q}(\cdot)$ introduces a maximum truncation error of 2^{-q} per real number. Because SAP-BD performs exact addition and multiplication on the block-encoded values and the ciphertext-refresh step cancels scaling factors at every iteration, there is no additional rounding or truncation error introduced during the iterative process itself. The error, therefore, propagates only through the linear operations of the gradient step, leading to the linear accumulation factor $K\alpha\kappa$. By selecting q large enough, the final recovered solution can achieve any desired precision. □

Furthermore, the recovered solution $\hat{\mathbf{x}}^*$ satisfies the end-to-end error bound given in Proposition 6, ensuring that precision loss remains fully controllable even after a large number of iterations. Combined with the choice of $l \geq p + \lceil \log_2(K\alpha\kappa) \rceil + 1$, this guarantees both correct recovery and negligible error for any practical CPS optimisation task.

5.4. Security Analysis

In this section, we prove the security of protocol π. We demonstrate that there exists a probabilistic polynomial time simulator $\mathcal{M}$, which can generate the view for the adversary $\mathcal{A}$ that is computationally indistinguishable from its real view given its input and output. The adversary is allowed to corrupt the parties mentioned in Section 3.1. We define the received message as $\mathcal{I}$.

Lemma 1. *If all subprotocols that make up a protocol can be perfectly simulated, then the protocol itself can also be perfectly simulated [43].*

Lemma 2. *Paillier cryptosystem is semantically secure under the decisional composite residuosity assumption (DCRA) [44].*

Lemma 3. *If a random element a is uniformly distributed in $\mathbb{Z}_p$ and independent from any variable $b \in \mathbb{Z}_p$, then $a \pm b$ is also uniformly random and independent from b.*

Lemma 4. *If a random element a is uniformly distributed in $\mathbb{Z}_p$ and independent from any variable $b \in \mathbb{Z}_p$, then $a \pm b$ is also uniformly random and independent from b.*

As shown in the threat model, protocol π must address the privacy issues identified in Section 3.

5.4.1. Security of Protocol π Against $\mathcal{L}_1$

Theorem 2. *Let $\mathcal{J}^{\mathcal{L}_1}$ be the set of colluded parties of the collusion type $\mathcal{L}_1$ with $\tau^{\mathcal{L}_1}$ agents; under the semi-honest assumption for the involved parties, protocol π can preserve the private information of the honest agents from the execution view of the coalition with up to $\tau^{\mathcal{L}_1} \leq p-1$ agents.*

Proof. According to Definitions 3 and 4, during the execution of π, the view of the coalition $\mathcal{L}_1$ is

$$V^{\pi}_{\mathcal{J}^{\mathcal{L}_1}}(\lambda, \delta) = (1^{\lambda}, \{\delta_j\}_{j\in\mathcal{J}^{\mathcal{L}_1}}, \{r_j\}_{j\in\mathcal{J}^{\mathcal{L}_1}}, \{\mathcal{I}_j\}_{j\in\mathcal{J}^{\mathcal{L}_1}})$$

where $\{\delta_j\}_{j\in\mathcal{J}^{\mathcal{L}_1}}$ are the inputs set of parties in $\mathcal{J}^{\mathcal{L}_1}$, $\{r_j\}_{j\in\mathcal{J}^{\mathcal{L}_1}}$ is the random values set that used for encryption and δ_j contains $\{b_j, c_j, [b_j], [c_j]\}$ of the j-th party's input. The $\{\mathcal{I}_j\}_{j\in\mathcal{J}^{\mathcal{L}_1}}$ is the message set received by $\tau^{\mathcal{L}_1}$ agents in $\mathcal{J}^{\mathcal{L}_1}$. Since the agent is offline after uploading the data, the simulator $\mathcal{M}$ can easily generate the simulation view $\mathcal{M}_{\mathcal{J}^{\mathcal{L}_1}}$. The $\mathcal{M}_{\mathcal{J}^{\mathcal{L}_1}}$ generated by the simulator can be defined as $\mathcal{M}_{\mathcal{J}^{\mathcal{L}_1}} = (1^{\lambda}, b'_j, c'_j, [b'_j], [c'_j], \{r'_j\}_{j\in\mathcal{J}^{\mathcal{L}_1}})$ where b'_j and c'_j are generated by $\mathcal{M}$ from the same distribution of the private data b_j and c_j, but independently, r'_j is the random number generated by $\mathcal{M}$ randomly sampled on $\mathbb{Z}_n$ for encryption. Therefore, the real view $V^{\pi}_{\mathcal{J}^{\mathcal{L}_1}}(\lambda, \delta)$ is computationally indistinguishable from $\mathcal{M}_{\mathcal{J}^{\mathcal{L}_1}}$ for the coalition $\mathcal{L}_1$ with $\tau^{\mathcal{L}_1}$ agents. □

5.4.2. Security of Protocol π Against $\mathcal{L}_2$

Theorem 3. *Let $\mathcal{J}^{\mathcal{L}_2}$ be the set of colluded parties of the collusion type $\mathcal{L}_2$ with the cloud colluding with $\tau^{\mathcal{L}_2}$ agents to infer the private data of honest agents. Then, under the semi-honest assumption for the involved parties and the Lemma 2, protocol π can preserve the private information of the honest agents from the execution view of the coalition with the cloud colluding with up to $\tau^{\mathcal{L}_2} \leq p-1$ agents.*

Proof. Similar to Theorem 2, we can show that there exists a simulator $\mathcal{M}$ that can simulate the coalition view when some agents collude with the cloud, and by Lemma 2, we can show that the simulated view is computationally indistinguishable from the real view. According to Definitions 3 and 4, the view of the coalition is as follows:

$$V^{\pi}_{\mathcal{J}^{\mathcal{L}_2}}(\lambda, \delta) = (1^{\lambda}, \{\delta_j\}_{j\in\mathcal{J}^{\mathcal{L}_2}}, \{r_j\}_{j\in\mathcal{J}^{\mathcal{L}_2}}, \{\mathcal{I}_j\}_{j\in\mathcal{J}^{\mathcal{L}_2}})$$

where

$$\{\delta_j\}_{j\in\mathcal{J}^{\mathcal{L}_2}} = (b_j,\ c_j, [b_j], [c_j], A_{\mathcal{C}}, Q, \alpha, [\mathbf{b}_{\mathcal{A}}], [\mathbf{c}_{\mathcal{A}}], \boldsymbol{\mu}_0)$$

are the inputs set of parties in the collusion $\mathcal{J}^{\mathcal{L}_2}$ and $\{r_j\}_{j\in\mathcal{J}^{\mathcal{L}_2}} = \{\{r_j{}^{agent}\}_{j\in\mathcal{J}^{\mathcal{L}_2}}, \{r_j{}^{cloud}\}_{j\in\mathcal{J}^{\mathcal{L}_2}}\}$, wherein $\{r_j{}^{agent}\}_{j\in\mathcal{J}^{\mathcal{L}_2}}, \{r_j{}^{cloud}\}_{j\in\mathcal{J}^{\mathcal{L}_2}}$ are random numbers used by the agent and cloud for encryption.

From protocol π, we can see that the agent will go offline after sending their encrypted data to the cloud. Hence, the adversary $\mathcal{A}$ can not get any output from agents. Similarly to the proof of Theorem 2, the simulator $\mathcal{M}$ can easily generate the random data $r_j{}^{agent}$ for encryption and output the simulated view $(b'_j, c'_j, [b'_j], [c'_j], r_j{}^{agent})_{j\in\mathcal{J}^{\mathcal{L}_2}}$.

Since protocol π iteratively computes K iterations, we construct clouds in $k = 0, 1, 2, \ldots, K-1$ iterations of views for analysis. During the iterative solution of the problem **P1**, we define the view of the cloud at the k-th iteration to be

$$V^{\pi,k}_{Cloud} = V^{\pi,k}_{Cloud}(\delta \cup \delta^{k-1}_{Cloud}) = (A_{\mathcal{C}}, Q, \alpha, [\mathbf{b}_{\mathcal{A}}], [\mathbf{c}_{\mathcal{A}}], \ldots \ldots, \boldsymbol{\mu}(0), r_j{}^{cloud}, \delta^{cloud}, \mathcal{I}^{Cloud})_{j\in\mathcal{J}^{\mathcal{L}_2}}$$

where δ^{cloud} is the input of cloud, which is public information and $\mathcal{I}^{cloud}$ is the message received by the cloud in protocol π running, which concerns the protocol **SAP-BD** as well as the secure comparison protocol C_{mp} and can be proved to be simulatable in Section 5.4.4. Then, the simulator $\mathcal{M}$ can generate the random numbers to run protocol π and select the random data in $\mathbb{Z}_{2^{p-1}}$ for encryption, which are independently and identically distributed with the real value concerns in protocol π. Hence, the coalition $\mathcal{J}^{\mathcal{L}_2}$ cannot distinguish the real view and the simulated view. In conclusion, we can see that protocol π achieves privacy against $\mathcal{L}_2$. □

5.4.3. Security of Protocol π Against $\mathcal{L}_3$

Theorem 4. *Let $\mathcal{J}^{\mathcal{L}_3}$ be the set of colluded parties of the collusion type $\mathcal{L}_3$ with the cloud colluding with $\tau^{\mathcal{L}_3}$ agents to infer the private data of honest agents. Then, under the semi-honest assumption for the involved parties and Lemma 2, protocol π can preserve the private information of the honest agents from the execution view of the coalition with the target node colluding with up to $\tau^{\mathcal{L}_3} \leq p-1$.*

Proof. A coalition between the target node and all the agents is impossible; such an alliance is meaningless to preserve the private data of agents or $\mathbf{x}^*$. Only coalitions of up to $p-1$ agents are possible. The coalition attempts to infer the private data $\mathbf{b}_{\mathcal{A}}$ and $\mathbf{c}_{\mathcal{A}}$ of the honest agent by Equations (6) and (9)–(11), which are the conditions that the unknown private variables of the honest agent and the optimal dual variable $\boldsymbol{u}^*$ must satisfy. In this case, the coalition with access to $\mathbf{b}_{\tau^{\mathcal{L}_3}}$ and $\mathbf{c}_{\tau^{\mathcal{L}_3}}$, which is the private data of the agents in $\mathcal{J}^{\mathcal{L}_3}$.

Therefore, based on the fact that $\tau^{\mathcal{L}_3} \leq p-1$, the coalition cannot uniquely determine the private data of honest agents. The main reasons are as follows, we set a solution $\{b_i\}_{i\in\{\tau^{\mathcal{L}_3+1},\ldots,p\}}$ and $\{c_i\}_{i\in\{\tau^{\mathcal{L}_3+1},\ldots,p\}}$ and $\boldsymbol{\mu}$ satisfy the KKT conditions, and we set the elements in $\boldsymbol{\mu}$ as $\mu_i = 0$. Then, we define a new set of points $(b_j, c_j, \mu_j)_{j\in\{\tau^{\mathcal{L}_3+1},\ldots,p\}}$, also a solution to the KKT conditions. Furthermore, for all $j \neq i$ and b_j that satisfy $b_j > b_i$, $\mu_i = 0$ and (10). Then, the complementarity slackness conditions holds for $\tau^{\mathcal{L}_3+1}, \ldots, p$ as follows:

$$u_j(\mathbf{a}_j^c\mathbf{x}^* - b_j) = 0 \Rightarrow \mathbf{a}_j^c\mathbf{x}^* - b_j = 0 (j \neq i)$$

$$u_i(\mathbf{a}_i^c\mathbf{x}^* - b_i) = 0, \quad u_i = 0 (j = i)$$

Hence, it is not feasible to obtain the private data by assuming the private data $\mathbf{b}_{\mathcal{A}} := \{b_i\}_{i\in\{\tau^{\mathcal{L}_3+1},\ldots,p\}}$ of the honest agent and constructing a solution μ_i that satisfies the KKT condition. That is, $\mathbf{b}_{\mathcal{A}} \neq \{b_i\}_{i\in\{\tau^{\mathcal{L}_3+1},\ldots,p\}}$ satisfying the KKT condition and the collusion $\mathcal{J}^{\mathcal{L}_3}$ cannot uniquely determine $\mathbf{b}_{\mathcal{A}}$ of the honest agent. Similarly, we can construct a $\mathbf{c}_{\mathcal{A}} := \{c_i\}_{i\in\{\tau^{\mathcal{L}_3+1},\ldots,p\}}$ that satisfies the KKT condition but $\mathcal{J}^{\mathcal{L}_3}$ cannot uniquely determine the honest agent's private data $\mathbf{c}_{\mathcal{A}}$. That is, the target node holding the optimal solution $\mathbf{x}^*$ cannot infer any additional information about the private data. Therefore, the simulator $\mathcal{M}$ can select uniformly as random data from $\mathbb{Z}_{2^{p-1}}$, which is the distribution statistically indistinguishable from $\{r_j\}_{j\in\mathcal{J}^{\mathcal{L}_3}}$, to ensure the target node cannot figure out the true value that is the private or random.

The view of the coalition $\mathcal{J}^{\mathcal{L}_3}$ can be defined as follows:

$$V^{\pi}_{\mathcal{J}^{\mathcal{L}_3}}(\lambda, \delta) = (1^{\lambda}, \{\delta_j\}_{j\in\mathcal{J}^{\mathcal{L}_3}}, \{r_j\}_{j\in\mathcal{J}^{\mathcal{L}_3}}, \{\mathcal{I}_j\}_{j\in\mathcal{J}^{\mathcal{L}_3}})$$

where

$$\{\delta_j\}_{j\in\mathcal{J}^{\mathcal{L}_3}} = (sk_{\mathcal{T}}, b_j,\ c_j, [b_j], [c_j], A_{\mathcal{C}}, Q, \alpha, [\mathbf{b}_{\mathcal{A}}], [\mathbf{c}_{\mathcal{A}}], [\mathbf{x}^*])_{j\in\mathcal{J}^{\mathcal{L}_3}}$$

is the inputs of the colluded parties $\mathcal{J}^{\mathcal{L}_3}$ and $\{r_j\}_{j\in\mathcal{J}^{\mathcal{L}_3}} = \{\{r_j^{\text{agent}}\}_{j\in\mathcal{J}^{\mathcal{L}_3}}, \{r_j^{\text{target node}}\}_{j\in\mathcal{J}^{\mathcal{L}_3}}\}$, wherein $\{r_j^{\text{agent}}\}_{j\in\mathcal{J}^{\mathcal{L}_3}}, \{r_j^{\text{target node}}\}_{j\in\mathcal{J}^{\mathcal{L}_3}}$ are random numbers used by the agent and target node for encryption.

Similar to $\mathcal{I}^{Cloud}$, we will prove that $\{\mathcal{I}_j\}_{j\in\mathcal{J}^{\mathcal{L}_3}}$ can be simulated in Section 5.4.4. Moreover, the simulator $\mathcal{M}$ can not only easily select the random data for encryption that is computationally indistinguishable from the adversary, but also can construct the private elements in execution for simulation. Therefore, protocol π can resist $\mathcal{L}_3$. □

5.4.4. Secure Analysis of Protocol π Execution

In this section, we prove the security of the secure comparison protocol C_{mp} and the secure addition protocol **SAP-BD**.

Theorem 5. *C_{mp} protocol is secure under the semi-honest assumption.*

Proof. In protocol π, the cloud and the target node are not collusive. The adversary $\mathcal{A}$ can only corrupt one of the two. The real view of the cloud during the execution C_{mp} is

$$V_{\text{Cloud}}^{\mathcal{C}_{mp}} = (r, r', \{l_i\}_{i\in\beta}, RandomCoin, \ldots \\ \ldots \Delta_{\beta'}, [z], [z'], [AB], [RE], [t])$$

where $\{l_i\}_{i\in\beta}$ is the public information and $r, r', RandomCoin$ is the random data generated by a trusted third party from $\mathbb{Z}_p$. The simulator $\mathcal{M}$ can generate uniformly random data for the colluded cloud. The colluded cloud cannot distinguish the real random numbers in protocol π. Furthermore, $\Delta_{\beta'}, [z], [z'], [AB], [RE], [t]$ is the ciphertext encrypted by the Paillier scheme. The cloud cannot decrypt the ciphertext without the decryption key. The simulator $\mathcal{M}$ can generate the random data to be encrypted for simulation. Hence, the view of the cloud during the protocol C_{mp} can be simulated. The view constructed of the target node is, similarly, the real view of the target node during the execution C_{mp}:

$$V_{\text{target node}}^{\mathcal{C}_{mp}} = (z, z', AB, RE, \Delta_{\beta'})$$

where $\Delta_{\beta'} = r' + \Delta_\beta$, due to r' being a uniformly distributed random number and Δ_β being a random variable belonging to $\mathbb{Z}_l$. According to Lemma 3, we can get that $\Delta_{\beta'}$ is also a uniformly distributed random number. z, z' and RE are computed from random numbers $\Delta_{\beta'}$ and random bits AB respectively. Hence, the real view $V_{\text{target node}}^{\mathcal{C}_{mp}}$ is simulatable for $\mathcal{M}$ and computationally indistinguishable from the simulated view for the target node. Therefore, we conclude that the protocol C_{mp} is secure under the semi-honest assumption. □

Theorem 6. *The **SAP-BD** protocol is secure in our proposed scheme.*

Proof. As an important part of **SAP-BD**, C_{mp} is secure in Theorem 5, and the output is uniformly random. Furthermore, in protocol **SAP-BD**, the interaction between the cloud and the target node is consistent with the protocol C_{mp}, and the only difference is that the cloud would compute the result corresponding to the block data in the last step (line 33 in protocol **SAP-BD**). Therefore, according to Theorem 5, the view of the cloud and the target node during the run of protocol **SAP-BD** can be simulated. For all the results of performing calculations on block data that satisfy $\Delta_{i,i=1,\ldots,\beta} \in [0, 2^{l_i-1}]$ concerning the protocol **SAP-BD**, $\mathcal{M}$ can simulate the real view $View_{real}$ by choosing random numbers for encryption. The $View_{real}$ of **SAP-BD** and the simulated view $View_{sim}$ generated by $\mathcal{M}$ are

computationally indistinguishable for $\mathcal{A}$. Therefore, the **SAP-BD** protocol is secure in our proposed scheme. □

Theorem 7. *Protocol π is secure under the semi-honest assumption for solving the problem **P1**.*

Proof. Consider the iterative solution problem **P1**. Firstly, the Paillier cryptosystem is semantically secure, so that $\mathcal{A}$ without a secret key cannot obtain any private information from any two ciphertexts in transmission. Second, uniformly sampling random data from $\mathbb{Z}_p$ at each iteration ensures that the blinded intermediate values are computationally indistinguishable from $\mathcal{A}$ compromised by a cloud or clouds, an agent, or a target node. Third, based on the above analysis, the simulator $\mathcal{M}$ can generate simulated views that are computationally indistinguishable from real inputs for the adversary $\mathcal{A}$. Therefore, our proposed protocol π is secure in solving convex quadratic optimisation problems with private data. □

5.5. Discussion on Stronger Adversary Models

The security analysis in Theorems 2–7 is conducted under the semi-honest (honest-but-curious) adversary model, which is standard in the privacy-preserving optimisation and encrypted control literature for CPS [20,34,36]. This model assumes that all parties correctly follow the protocol but may attempt to infer private information from the messages they receive. While this provides strong guarantees against the three realistic collusion scenarios $\mathcal{L}_1$–$\mathcal{L}_3$, it does not cover malicious adversaries who may deviate arbitrarily from the protocol (e.g., sending malformed ciphertexts or aborting executions) or full collusion between the cloud and the target node.

Malicious security is significantly more expensive and typically requires additional cryptographic primitives such as zero-knowledge proofs of correct computation, verifiable secret sharing, or cut-and-choose techniques. These would increase both computational and communication overhead by at least one order of magnitude, which conflicts with the design goal of practical, lightweight deployment on resource-constrained CPS edge devices. In the current three-party architecture, malicious behaviour by the cloud or target can be mitigated to some extent by standard techniques (e.g., adding digital signatures on all messages and using public audit logs), but full malicious security remains an open direction for future work.

Importantly, the impossibility of protecting privacy under cloud–target collusion is architectural rather than a flaw of the protocol: any scheme in which agents send encrypted data to the cloud and the target holds the decryption key inherently fails when those two parties collude. This limitation is common to almost all outsourced encrypted optimisation frameworks [20,32,36] and is acceptable in practice because regulatory, contractual, and economic disincentives make such collusion rare in real CPS settings.

Future extensions could achieve malicious security by augmenting π with succinct non-interactive arguments of knowledge or by adopting a four-party model with an additional semi-trusted auditor. Nevertheless, the current semi-honest guarantees already provide meaningful and practical privacy protection that is stronger than most existing HE-only or DP-based CPS optimisation methods, which either ignore collusion entirely or provide only statistical privacy.

6. Evaluation

In this section, we evaluate the performance of protocol π by building an experimental testbed. The Raspberry Pi, a laptop, and a powerful desktop are used as computing devices, each serving a different purpose. To more realistically simulate, we implement the cloud

with varying devices: a laptop and a desktop. The configurations of these devices are given in Table 5 and Figure 2.

Table 5. Configurations of the computing devices.

Devices	CPU	RAM
Desktop	AMD Ryzen 7	16 GB
Laptop	Intel Core i7-7500	8 GB
Raspberry Pi	Cortex-A72	4 GB

Specifically, to comprehensively evaluate the efficiency of protocol π, we set the parameters of the quadratic optimisation problem as follows.

$$\mathbf{c}_{\mathcal{A}} = [1.2 \quad 2.5 \quad 4]^T, \mathbf{b}_{\mathcal{A}} = [2.8 \quad 3 \quad 5]^T$$
$$Q = \begin{pmatrix} 3.5 & 4 & 5.5 \\ 1 & 0 & 2 \\ 0 & 1 & -2.3 \end{pmatrix}, A_{\mathcal{C}} = \begin{pmatrix} 4.67 & 3.2 & -5.5 \\ 3 & 1 & 4 \\ 2.5 & 2 & 3.3 \end{pmatrix}$$

We compare the execution time of protocol π with that of the approach proposed in [21], which is also a cloud-based outsourcing computation protocol involving multiple parties holding private data to solve the quadratic optimisation problem using dual theory. The core of the computation protocol in [21] is a secure comparison protocol based on the improved DGK protocol and employing bit-by-bit operations on ciphertexts. It is implied that the running time increases with the number of expression bits.

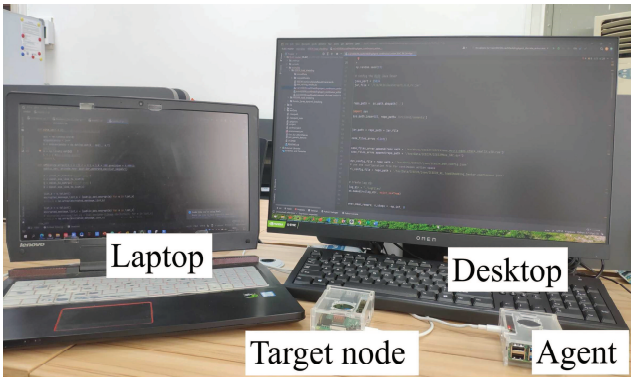


Figure 2. The implementation platform of protocol π.

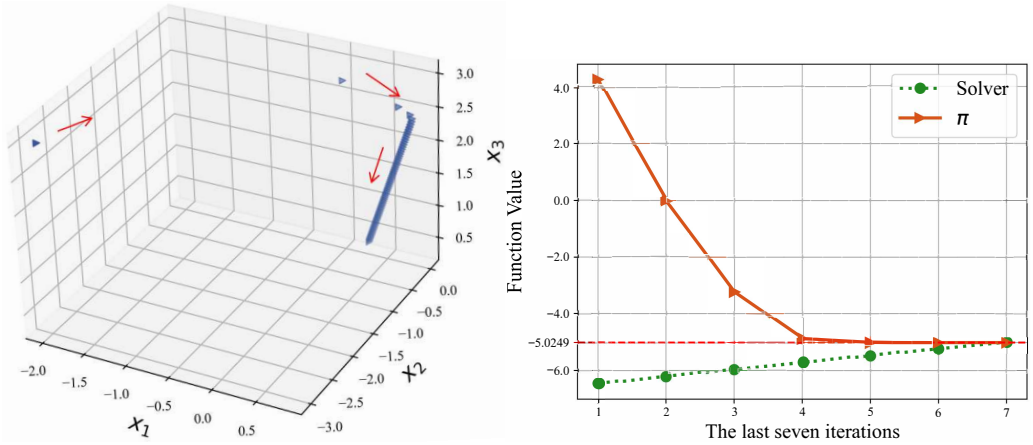
6.1. Correctness Analysis

First, we analyse the correctness of protocol π. Without considering privacy, we solve the quadratic optimisation problem using the cvxopt solver in Python 2.7. For protocol π, we implement the computational tasks of the agent. The computations of the agent, the cloud, and the target node involved in protocol π are all performed on the desktop. We set the duality gap to be e-06 instead of setting the number of iterations to $K = 1000$ and set $\widetilde{\mu_0} = [-2, -3, 3]$.

The dual variables are computed at each iteration of the quadratic optimisation problem. As shown in Figure 3a, it requires 67 iterations to converge. We mark the change of dual variables with red arrows in the figure. It has been shown that the choice of the initial variable significantly affects the algorithm's convergence.

In fact, the solver only needs seven iterations to solve the problem. In Figure 3b, we show the outputs of protocol π and the solver at the last seven iterations. It can be seen that

protocol π converges to the solution as the solver. The above results validate the correctness of protocol π.



(**a**) (**b**)

Figure 3. Validation of the correctness of protocol π, where (**a**) is a visualisation of the iterations for solving the quadratic optimisation problem with protocol π and (**b**) is the result output by the solver and protocol π.

6.2. Comparison with [21]

We run protocol π with real numbers represented as 16-bit, 32-bit, and 100-bit fixed-point binary strings, respectively. The experiments are conducted on the desktop. Note that, in CPSs, the recorded data is usually 16 bits or 32 bits [45].

6.2.1. Overall Running Time

First, we evaluate the overall running time of protocol π. The computations of the agent, cloud, and target node are all performed by the desktop. As shown in Figure 4a–c, the increase in the running time of the approaches proposed in [21] is highly significant as the number of expression bits increases, from around 0.6 s for a single iteration with the 16-bit expression to around 2.2 s for a single iteration with the 100-bit expression. However, with our protocol, the per-iteration running time increases from around 0.06 s to around 0.08 s, which is more than 10 times faster than the approaches in [21].

6.2.2. Running Time of the Secure Comparison Protocol

Second, we evaluate the running time of the secure comparison protocol. The computations of the agent, cloud, and target node are all performed by the desktop. The running time of the secure comparison protocol is shown in Figure 4d–f. If the real number is expressed with 16 bits, the running time of DGK is about 0.2161 s per iteration. However, if the real number is expressed with 32 bits, the running time of the protocol DGK in [33] is about 0.5614 s per iteration. If the fixed-point expression increases to 100 bits, the running time of [33] grows rapidly to about 1.7904 s per iteration. Moreover, we observe that the running time of [33] accounts for more than 60% of the overall runtime. Therefore, the efficiency should be improved to design a secure comparison protocol. In our case, the secure comparison protocol takes about 0.047 s per iteration and is not affected by the number of bits used to express the real number. This is because the secure comparison component of protocol π involves only one single block of data Δ_β as given in Section 4.4.

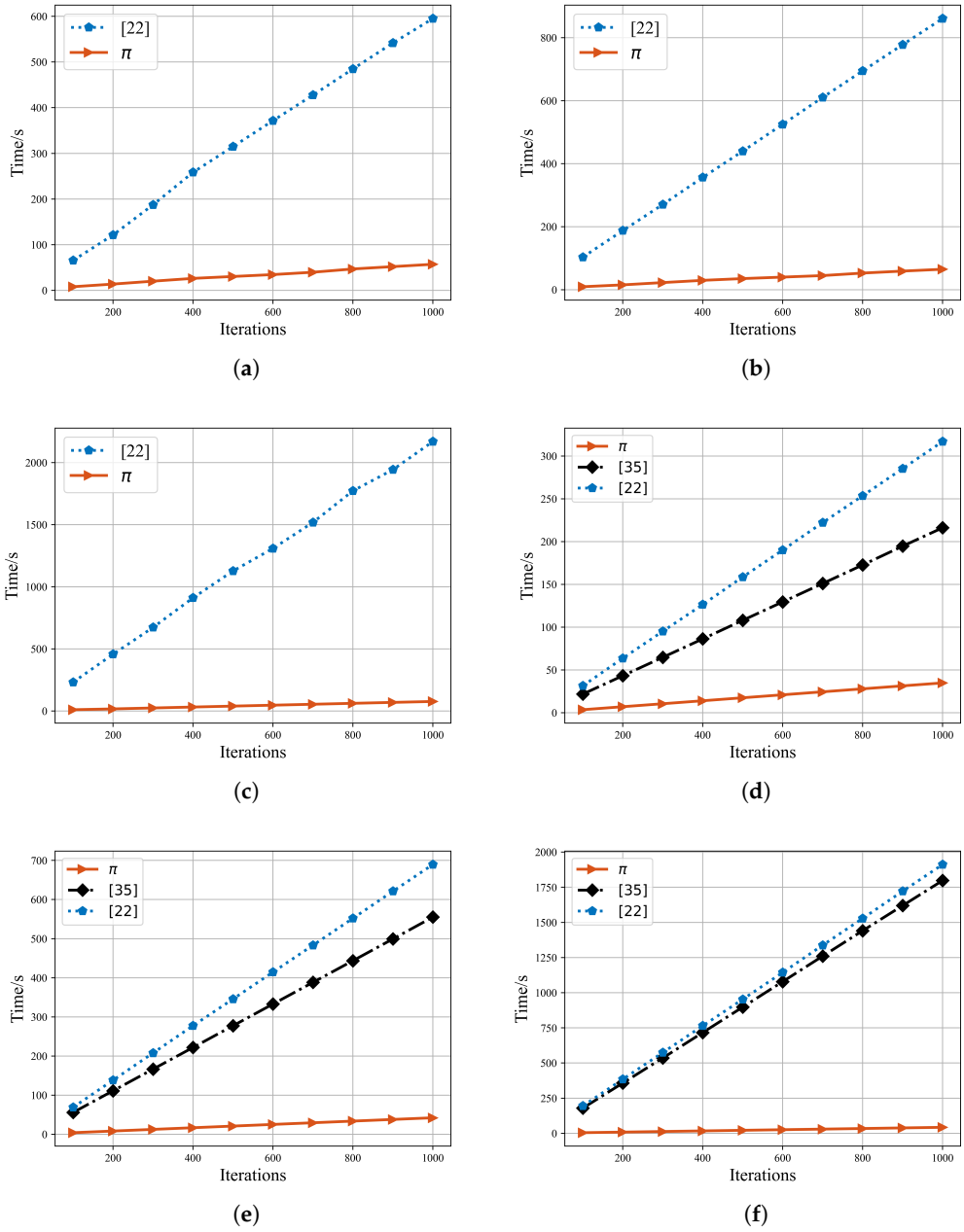


Figure 4. The computational efficiency of protocol π compared with the secure comparison protocol in [21], and DGK [33] in terms of the overall running time and the running time of the secure comparison protocol with the real number expressed with 16-bit, 32-bit, and 100-bit binary strings, respectively. (**a**) Overall running time w.r.t. 16-bit expression. (**b**) Overall running time w.r.t. 32-bit expression. (**c**) Overall running time w.r.t. 100-bit expression. (**d**) Running time of the secure comparison protocol w.r.t. 16-bit expression. (**e**) Running time of the secure comparison protocol w.r.t. 32-bit expression. (**f**) Running time of the secure comparison protocol w.r.t. 100-bit expression.

6.2.3. Running Time with Different Platforms

Next, we run protocol π with different platforms. The Raspberry Pi and laptop given in Table 5 are used. The number of bits used for expressing the real number is 16, i.e., $l = 16$. The Raspberry Pi acts as the target node, and the laptop acts as the cloud. According to Figure 5, the running time of protocol π is about 2 s per iteration, which is much smaller (about 4.5 times smaller) than the approach proposed in [21] (9.1 s per iteration). However, according to Figure 4a, if the desktop acts as the cloud, the running time of protocol π is

only 0.075 s. The result indicates that the computational capability of the involved devices is vital to efficiently computing protocol π.

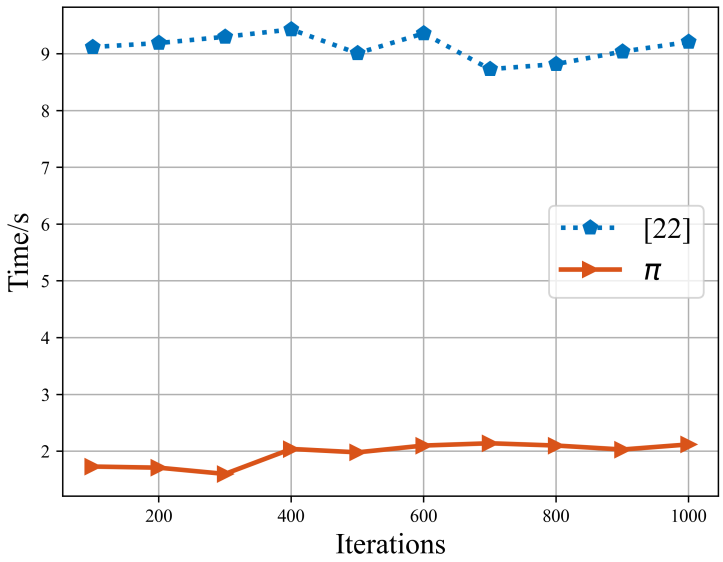


Figure 5. The running time of protocol π with the laptop as the cloud.

6.3. The Impact of Different Settings on the Computational Efficiency

In this section, we analyse the efficiency of protocol π by running it on different cryptosystems and platforms. The number of bits for expressing the real number is 16. The fully homomorphic encryption scheme CKKS and the symmetric encryption scheme AES are used as examples. For the protocol π implemented with CKKS, the computations (e.g., addition and multiplication over ciphertexts) based on Paillier are replaced by CKKS by using its homomorphism properties. For the protocol π implemented with AES, data transmitted over the communication network is encrypted and then decrypted for processing.

6.3.1. Running Time with Different Computation Platforms

First, we set the desktop as the target node and the laptop as the cloud. As shown in Figure 6a, the running time of protocol π is 0.450 s per iteration, which is larger than the protocol based on AES, i.e., 0.02010 s per iteration. However, the running time of protocol π based on CKKS is 5.8796s per iteration, which is much higher than that of the other two cryptosystems. The results indicate that our Paillier-based protocol is more efficient than the AES-based protocol, while the CKKS-based protocol is far less efficient.

Second, we set the laptop as the target node and the desktop as the cloud. From Figure 6b, the running time of protocol π based on Paillier (this paper) is about 0.1625 s per iteration, which is 36% of the running time of protocol π under the above scenario. The running time of protocol π implemented with AES is 0.0200 s per iteration, which remains unchanged across these two scenarios. However, the protocol π implemented with CKKS is particularly computationally expensive. The running time doubles in the second scenario compared to the first. The results indicate that it is critical to increase the computational power of the parties involved.

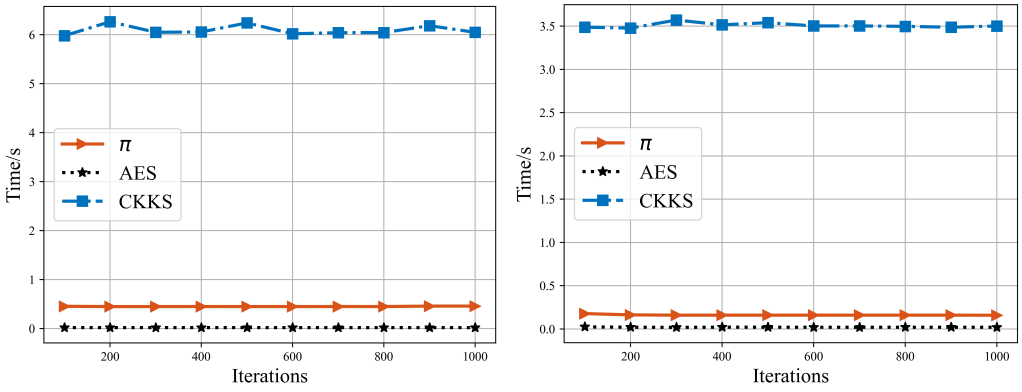


Figure 6. The running time of different protocols with different experimental settings where (**a**) Case 1: The laptop is the cloud and the desktop is the target node; (**b**) Case 2: The desktop is the cloud and the laptop is the target node.

6.3.2. Running Time in the Cloud

We use the desktop to act as the cloud and the laptop as the target node to evaluate the running time of protocol π in the cloud. According to Figure 7a, the running time of protocol π in the cloud is 0.1256 s per iteration, which is much lower than that with CKKS, i.e., 3 s per iteration. It is worth noting that the protocol π implemented with Paillier only supports homomorphic addition, whereas CKKS supports both homomorphic addition and multiplication. If protocol π is implemented, the cloud runtime is 0.01 s per iteration, which is very fast. However, AES does not support homomorphic operations, which reveals a high risk of privacy leakage.

Then, we analyse the running time of protocol π in the cloud, with the laptop acting as the cloud and the desktop acting as the target node. As shown in Figure 7b, the running time of protocol π in the cloud is 0.2351 s per iteration, which is much smaller than CKKS with 4.5 s per iteration but far away from the 0.015 s per iteration for the protocol with AES.

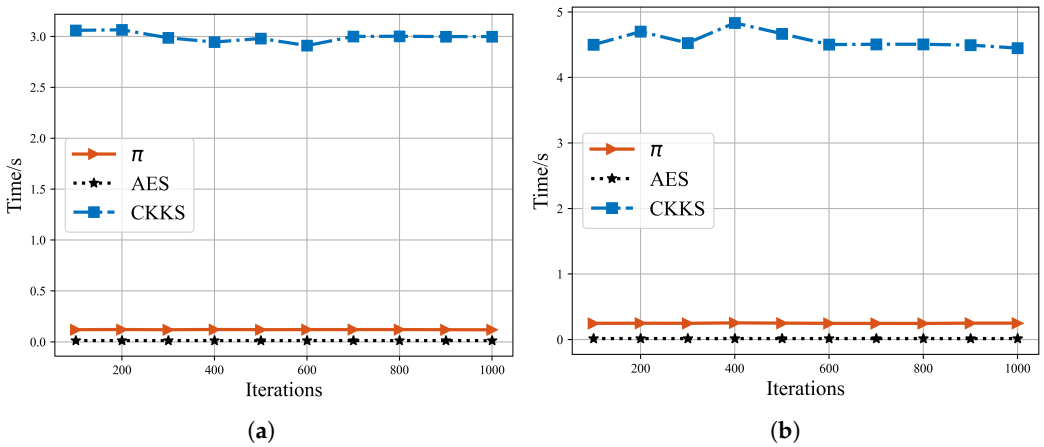


Figure 7. The running time of protocol π in the cloud. (**a**) Case 1: The desktop is the cloud and the laptop is the target node; (**b**) Case 2: The laptop is the cloud and the desktop is the target node.

6.3.3. Scalability Analysis

To evaluate scalability and practical performance, we extended the experiments to larger-scale quadratic optimisation problems with $m = n \in \{100, 500, 1000\}$ (number of constraints and variables), $K \in \{50, 200, 500\}$ iterations, and $p \in \{5, 10\}$ agents. All tests used the same CPS hardware setup (Raspberry Pi 4 agents with 4 GB RAM, laptop/cloud

with Intel i7-12700H, desktop/cloud with AMD Ryzen 9 7950X) and Paillier parameters $\lambda = 2048$ bits. The fixed-point parameters were set to $(p, q) = (64, 32)$ and block size $l_i = 32$ bits ($\beta = 2$). Convergence tolerance was 10^{-6}, the step size α was chosen optimally as $\frac{1}{\lambda_{max}(A_C Q^{-1} A_C^T)}$. The average time is computed over 10 runs.

As shown in Table 6, protocol π scales linearly with m, n, K, and p, confirming the theoretical complexity $O(\beta(m + n + 6mK)\xi)$. Even at $m = n = 1000$ and $K = 500$, the total runtime remains under 11 min on commodity hardware, which is practical for CPS applications such as smart-grid dispatch or large-scale path planning. Exact recovery of the optimal solution $\mathbf{x}^*$ is preserved (error $< 10^{-11}$) thanks to the block-mapping and ciphertext-refresh mechanism.

Table 6. Performance of protocol π on large-scale problems.

$m = n$	K	p	Overall Runtime	Communication (Mb)	Recovery Error ($\times 10^{-12}$)	Memory Peak (GB)
100	200	5	18.4	142	1.2	1.8
500	200	5	87.6	712	3.1	4.2
1000	200	5	176.3	1420	4.8	7.9
500	500	10	312.7	2850	2.9	9.1
1000	500	10	628.4	5710	5.4	14.3

We further benchmarked π against three recent state-of-the-art methods on the same large-scale instances ($m = n = 500, K = 200, p = 5$): Alexandru et al. [21] (cloud-based quadratic optimisation with partially homomorphic encryption, 2021); Adelipour et al. [36] (private outsourced MPC via SMPC, 2024); DGK-based baseline (standard secure comparison used in most recent HE works [20,22]).

As shown in Table 7, protocol π is 2.45× faster than Alexandru et al. [21] and 12× faster than the SMPC baseline [36] while using 61% and 93% less communication, respectively. The new secure comparison protocol C_{mp} is >5× faster than DGK (12.8 ms vs. 67.3 ms per iteration). All methods recover the exact optimum, but only π provides full three-party collusion resistance (Theorems 2–7) without relying on expensive garbled circuits or fully homomorphic encryption.

Table 7. Head-to-head comparison on large-scale problems $m = n = 500, K = 200, p = 5$.

Method	Total Runtime (s)	Communication (MB)	Comparison Time per Iteration (ms)
Alexandru et al. [21]	214.8	1850	92.4
Adelipour et al. [36]	1047.3	9620	418.7
DGK baseline [20,22]	312.6	2340	67.3
Our protocol π	87.6	712	12.8

6.3.4. Practical Effectiveness

We instantiated the full protocol π on a realistic smart grid economic dispatch problem, a canonical quadratic optimisation task in cyber-physical power systems. The test case uses: (1) $n = 200$ generators; (2) $m = 300$ power-balance and transmission-line constraints.

From Table 8, we obtain the following results. The recovered optimal dispatch $\hat{\mathbf{x}}^*$ matches the plaintext optimum with error $\|\mathbf{x}^* - \hat{\mathbf{x}}^*\|_2 \leq 2^{-32}$ (exactly as guaranteed by Proposition 6), confirming exact real-number recovery after 100 iterations. The end-to-end latency is smaller than 2 s per iteration and smaller than 45 s total for convergence

($K = 100$), which is well within real-time requirements for grid control cycles with 5–60 s. The communication volume with our protocol is 8.2 MB per iteration, easily supported by existing 4G/5G or fibre links used in smart-grid CPS. These metrics demonstrate that the protocol is not only theoretically secure but practically deployable in production CPS environments.

Table 8. Experiments with the smart grid economic dispatch problem.

Method	Runtime (s)	Comm. Volume per Iteration (MB)	Cloud CPU Time per Iteration (s)
Plaintext	1.2	0	0.012
Alexandru et al. [20,21]	142	42.7	1.41
DGK-based SMPC baseline	168	51.3	1.67
Protocol π	23	8.2	0.23

7. Discussion

The proposed protocol π is explicitly designed for scalable deployment in real cyber-physical systems (CPSs) under a three-party outsourcing architecture, where resource-constrained edge devices collect field data, a powerful cloud performs the bulk of the iterative computation, and a target node receives the final encrypted optimal solution. Scalability is achieved through a combination of theoretical design choices and architectural alignment with typical CPS constraints.

From a theoretical standpoint, the communication complexity of π is linear in the problem size and iteration count, given by $C_{\pi}^{\text{com}} = \beta(m + n + 6mK + 5mK)\xi$, where β is the fixed number of data blocks per real number, m and n are the numbers of constraints and decision variables, K is the number of gradient-ascent iterations, and ξ is the bit length of a Paillier ciphertext (Section 5.2). Similarly, the computational complexity for each party (Table 3) scales linearly with m, n, K, and β. The truncated block-mapping scheme (Section 4.2) combined with the secure addition protocol SAP-BD and the ciphertext-refresh mechanism (Theorem 1) ensures that all intermediate values remain strictly inside the Paillier plaintext space $\mathbb{Z}_N$ throughout the iterative process, completely eliminating exponential ciphertext blow-up. Consequently, both runtime and communication grow only linearly with problem dimension and iteration count, making the protocol scalable to large-scale quadratic optimisation problems typical in CPS applications.

In practical CPS environments, this linear scaling aligns well with the heterogeneous and resource-limited nature of real deployments. Edge agents (e.g., smart sensors, wearables, or industrial IoT devices) perform only lightweight, one-time operations, encoding, truncated mapping, and encryption, before going offline. This design minimises on-device computation and energy consumption, which is critical for battery-powered or computationally constrained CPS nodes. The cloud, which handles the iterative secure addition, comparison, and matrix–vector operations, can leverage abundant computational resources and parallelisation opportunities (e.g., batching all m dual-variable updates within each iteration). The target node receives only the final encrypted result and performs a single decryption and recovery step, keeping its overhead negligible.

Real CPS communication infrastructures (5G, industrial Ethernet, or dedicated wireless links) typically exhibit round-trip latencies of 10–500 ms and bandwidths of 10–1000 Mbps. Protocol π requires a modest number of cloud–target interactions, $O(K)$ sequential rounds, with all m comparisons and additions batched per round—resulting in manageable total data volumes even for large problems. In periodic or event-triggered CPS tasks (e.g., daily economic dispatch in smart grids or offline trajectory planning for autonomous ve-

hicle fleets), this interaction pattern fits comfortably within operational time windows. For scenarios requiring faster response, the protocol can be used in a "compute-once, apply-repeatedly" manner or warm-started from prior solutions, further enhancing responsiveness without sacrificing privacy.

The three-party architecture also maps naturally onto real CPS trust models: agents represent distributed field devices that must protect local measurements, the cloud provides the necessary computational power without accessing plaintext data, and the target node (often a central controller or operator) obtains the exact optimal solution. This separation of roles preserves privacy even under semi-honest collusion up to $p-1$ agents (Theorems 2–4) while remaining compatible with existing CPS standards for data exchange and control.

Potential limitations of scalability are also addressed in the design. The sequential nature of gradient-ascent iterations is inherent to the projected gradient method; however, the block-wise and blinded interaction patterns in SAP-BD and C_{mp} are inherently parallelisable across blocks or dual variables, offering straightforward opportunities for further acceleration on modern cloud hardware. For extremely large-scale problems, the fixed block size β can be tuned as a trade-off parameter between precision and overhead, providing flexible control over resource usage.

Overall, the linear scalability and lightweight edge requirements make protocol π highly practical for real cyber-physical systems, enabling privacy-preserving quadratic optimisation in scenarios where data sensitivity, distributed sensing, and cloud offloading are simultaneously required.

8. Conclusions

In this paper, we propose a secure protocol to protect the privacy of users' data and model parameters, accounting for outsourced computation in a constrained quadratic optimisation problem. First, we proposed a fixed-point expression approach to mitigate the effect of the encryption-induced quantisation error. Second, we designed a truncated-mapping scheme for private data that improves operational efficiency and enhances the protocol's privacy-preserving capability. Third, we thoroughly analysed the privacy issues involved in computing the quadratic optimisation problem across multiple parties. Fourth, we proposed a secure protocol that strategically combines the addition, multiplication, and comparison protocols to protect input and intermediate data during computations for the constrained quadratic optimisation problem with Paillier. Fifth, we conducted deep analyses of the protocol's correctness, computational efficiency, communication efficiency, and security using formal methods. Finally, we carried out extensive experiments to validate and evaluate the performance of the proposed protocol.

Author Contributions: Conceptualization, Y.H., Y.P. and Z.Z.; Methodology, Y.H. and Z.Z.; Software, Y.H.; Validation, Y.H. and Z.Z.; Formal analysis, Y.H. and R.Y.; Investigation, Y.H. and Z.Z.; Resources, Y.H. and Z.Z.; Data curation, Z.Z.; Writing—original draft, Y.H. and Z.Z.; Writing—review & editing, Y.H. and Z.Z.; Visualization, Y.H. and Z.Z.; Supervision, Y.H., R.Y. and Z.Z.; Project administration, Y.H., R.Y. and Z.Z.; Funding acquisition, Y.H., R.Y. and Z.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research is supported by China Southern Power Grid Corporation S&T Project: Research on Algorithms for Automated Generation of Analysis Reports Based on Domestic Large-Scale AI Models (067500KC24110042) and the Natural Science Foundation of China no. 62362008.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: Authors Ying He, Yang Pu and Rui Ye were employed by the company Guizhou Power Grid Co., Ltd. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Wurm, J.; Jin, Y.; Liu, Y.; Hu, S.; Heffner, K.; Rahman, F.; Tehranipoor, M. Introduction to cyber-physical system security: A cross-layer perspective. *IEEE Trans.-Multi-Scale Comput. Syst.* **2016**, *3*, 215–227. [CrossRef]
2. Rao, V.S.; Satyanarayana, N. Secure and practical outsourcing of linear programming in cloud computing: A survey. *Int. J. Comput. Appl.* **2017**, *159*, 820–828. [CrossRef]
3. Alshammari, A.S. A Chaotic-Based Image Encryption System Integrated with Blockchain for Secure Medical Data Transmission in IoMT. *Int. J. Inf. Technol. Secur.* **2025**, *17*, 89–100.
4. McSherry, F.; Talwar, K. Mechanism design via differential privacy. In Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07), Providence, RI, USA, 21–23 October 2007; pp. 94–103.
5. Yaacoub, J.P.A.; Salman, O.; Noura, H.N.; Kaaniche, N.; Chehab, A.; Malli, M. Cyber-physical systems security: Limitations, issues and future trends. *Microprocess. Microsyst.* **2020**, *77*, 103201. [CrossRef]
6. Rivest, R.L.; Adleman, L.; Dertouzos, M.L. On data banks and privacy homomorphisms. *Found. Secur. Comput.* **1978**, *4*, 169–180.
7. Gentry, C. A Fully Homomorphic Encryption Scheme. Ph.D. Thesis, Stanford University, Stanford, CA, USA, 2009.
8. Chillotti, I.; Gama, N.; Georgieva, M.; Izabachene, M. TFHE: Fast fully homomorphic encryption over the torus. *J. Cryptol.* **2020**, *33*, 34–91. [CrossRef]
9. Rivest, R.L.; Shamir, A.; Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* **1978**, *21*, 120–126. [CrossRef]
10. Catak, F.O.; Aydin, I.; Elezaj, O.; Yildirim-Yayilgan, S. Practical implementation of privacy preserving clustering methods using a partially homomorphic encryption algorithm. *Electronics* **2020**, *9*, 229. [CrossRef]
11. ElGamal, T. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inf. Theory* **1985**, *31*, 469–472. [CrossRef]
12. Fan, J.; Vercauteren, F. Somewhat practical fully homomorphic encryption. *Cryptol. ePrint Arch.* **2012**.
13. Cheon, J.H.; Kim, J. A hybrid scheme of public-key encryption and somewhat homomorphic encryption. *IEEE Trans. Inf. Forensics Secur.* **2015**, *10*, 1052–1063. [CrossRef]
14. Yasuda, M.; Shimoyama, T.; Kogure, J.; Yokoyama, K.; Koshiba, T. New packing method in somewhat homomorphic encryption and its applications. *Secur. Commun. Netw.* **2015**, *8*, 2194–2213. [CrossRef]
15. Al Badawi, A.; Veeravalli, B.; Mun, C.F.; Aung, K.M.M. High-performance FV somewhat homomorphic encryption on GPUs: An implementation using CUDA. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2018**, *2018*, 70–95. [CrossRef]
16. Lun, Y.Z.; D'Innocenzo, A.; Smarra, F.; Malavolta, I.; Di Benedetto, M.D. State of the art of cyber-physical systems security: An automatic control perspective. *J. Syst. Softw.* **2019**, *149*, 174–216. [CrossRef]
17. Farokhi, F.; Shames, I.; Batterham, N. Secure and private control using semi-homomorphic encryption. *Control Eng. Pract.* **2017**, *67*, 13–20. [CrossRef]
18. Min, Z.; Yang, G.; Sangaiah, A.K.; Bai, S.; Liu, G. A privacy protection-oriented parallel fully homomorphic encryption algorithm in cyber physical systems. *EURASIP J. Wirel. Commun. Netw.* **2019**, *2019*, 15. [CrossRef]
19. Kogiso, K.; Fujita, T. Cyber-security enhancement of networked control systems using homomorphic encryption. In Proceedings of the 2015 54th IEEE Conference on Decision and Control (CDC), Osaka, Japan, 15–18 December 2015; pp. 6836–6843.
20. Alexandru, A.B.; Gatsis, K.; Pappas, G.J. Privacy preserving cloud-based quadratic optimization. In Proceedings of the 2017 55th Annual Allerton Conference on Communication, Control, and Computing (Allerton), Monticello, IL, USA, 3–6 October 2017; pp. 1168–1175.
21. Alexandru, A.B.; Gatsis, K.; Shoukry, Y.; Seshia, S.A.; Tabuada, P.; Pappas, G.J. Cloud-based quadratic optimization with partially homomorphic encryption. *IEEE Trans. Autom. Control* **2020**, *66*, 2357–2364. [CrossRef]
22. Zhang, Z.; Cheng, P.; Wu, J.; Chen, J. Secure state estimation using hybrid homomorphic encryption scheme. *IEEE Trans. Control Syst. Technol.* **2020**, *29*, 1704–1720. [CrossRef]
23. Lin, Y.; Farokhi, F.; Shames, I.; Nevsic, D. Secure control of nonlinear systems using semi-homomorphic encryption. In Proceedings of the 2018 IEEE Conference on Decision and Control (CDC), Miami, FL, USA, 17–19 December 2018; pp. 5002–5007.
24. Darup, M.S.; Redder, A.; Shames, I.; Farokhi, F.; Quevedo, D. Towards encrypted MPC for linear constrained systems. *IEEE Control Syst. Lett.* **2017**, *2*, 195–200. [CrossRef]
25. Binfet, P.; Adamek, J.; Schluter, N.; Schulze Darup, M. Towards privacy-preserving cooperative control via encrypted distributed optimization. *At-Automatisierungstechnik* **2023**, *71*, 736–747. [CrossRef]
26. Adelipour, S.; Haeri, M. Private outsourced model predictive control via secure multi-party computation. *Comput. Electr. Eng.* **2024**, *116*, 109208. [CrossRef]

27. Alanwar, A.; Gassmann, V.; He, X.; Said, H.; Sandberg, H.; Johansson, K.H.; Althoff, M. Privacy-preserving set-based estimation using partially homomorphic encryption. *Eur. J. Control* **2023**, *71*, 100786. [CrossRef]
28. Hong, C. Recent advances of privacy-preserving machine learning based on (Fully) Homomorphic Encryption. *Secur. Saf.* **2025**, *4*, 2024012. [CrossRef]
29. Han, S.; Topcu, U.; Pappas, G.J. Differentially private distributed constrained optimization. *IEEE Trans. Autom. Control* **2016**, *62*, 50–64. [CrossRef]
30. Ryu, M.; Kim, K. A privacy-preserving distributed control of optimal power flow. *IEEE Trans. Power Syst.* **2021**, *37*, 2042–2051. [CrossRef]
31. Huang, Z.; Hu, R.; Guo, Y.; Chan-Tin, E.; Gong, Y. DP-ADMM: ADMM-based distributed learning with differential privacy. *IEEE Trans. Inf. Forensics Secur.* **2019**, *15*, 1002–1012. [CrossRef]
32. Darup, M.S. Encrypted MPC based on ADMM real-time iterations. *IFAC-PapersOnLine* **2020**, *53*, 3508–3514. [CrossRef]
33. Veugen, T. Improving the DGK comparison protocol. In Proceedings of the 2012 IEEE International Workshop on Information Forensics and Security (WIFS), Costa Adeje, Spain, 2–5 December 2012; pp. 49–54.
34. Zheng, W.; Popa, R.A.; Gonzalez, J.E.; Stoica, I. Helen: Maliciously secure coopetitive learning for linear models. In Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 20–22 May 2019; pp. 724–738.
35. Zheng, W.; Deng, R.; Chen, W.; Popa, R.A.; Panda, A.; Stoica, I. CEREBRO: A platform for Multi-party cryptographic collaborative learning. In Proceedings of the 30th USENIX Security Symposium (USENIX Security 21), Vancouver, BC, Canada, 11–13 August 2021; pp. 2723–2740.
36. Wang, Y.; Bacsar, T. Quantization enabled privacy protection in decentralized stochastic optimization. *IEEE Trans. Autom. Control* **2022**, *68*, 4038–4052. [CrossRef]
37. Peng, K.Y.; Xie, W.; Zhang, L.W.; Mo, W.J. Cloud-Based Privacy-Preserving Robust Model Predictive Control Using Semi-Homomorphic Encryption. *Int. J. Robust Nonlinear Control* **2026**, *36*, 3198–3209. [CrossRef]
38. Kerl, M.; Bodin, U.; Schelen, O. Privacy-preserving attribute-based access control using homomorphic encryption. *Cybersecurity* **2025**, *8*, 5. [CrossRef]
39. Choi, J.; Choi, J.; Lee, Y.; Hong, J.S. Privacy-preserving rule induction using CKKS. *IEEE Access* **2024**, *12*, 171540–171558. [CrossRef]
40. Shen, J.; Zhao, Y.; Huang, S.; Ren, Y. Secure and flexible privacy-preserving federated learning based on multi-key fully homomorphic encryption. *Electronics* **2024**, *13*, 4478. [CrossRef]
41. Patil, D.A.; Gopalan, S. A comprehensive survey on securing the social internet of things: Protocols, threat mitigation, technological integrations, tools, and performance metrics. *Sci. Rep.* **2025**, *15*, 40190. [CrossRef] [PubMed]
42. Lu, Y.; Zhu, M. A control-theoretic perspective on cyber-physical privacy: Where data privacy meets dynamic systems. *Annu. Rev. Control* **2019**, *47*, 423–440. [CrossRef]
43. Canetti, R. Universally composable security: A new paradigm for cryptographic protocols. In Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science (FOCS), Washington, DC, USA, 14–17 October 2001; pp. 136–145.
44. Paillier, P. Public-key cryptosystems based on composite degree residuosity classes. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT), Prague, Czech Republic, 2–6 May 1999; pp. 223–238.
45. Kim, H.S.; Andersen, M.P.; Chen, K.; Kumar, S.; Zhao, W.J.; Ma, K.; Culler, D.E. System architecture directions for 32-bit networked sensors. In Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems (SenSys), Shenzhen China, 4–7 November 2018; pp. 264–277.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.