Secure and Private Approximated Coded Distributed Computing Using Elliptic Curve Cryptography

Houming Qiu[1,2] and Kun Zhu[1,2(✉)]

[1] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China
{hmqiu56,zhukun}@nuaa.edu.cn
[2] Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing, China

Abstract. In large-scale distributed computing systems, coded computing has attracted considerable attention since it can effectively mitigate the impact of stragglers. Nonetheless, several emerging issues seriously restrict the performance of coded distributed systems. First, the presence of colluding workers collude results in serious privacy leakage issues. Second, few existing works consider security issues in data transmission. Third, the number of required results to wait for increases with the degree of polynomial functions. In this paper, we propose a secure and private approximated coded distributed computing (SPACDC) scheme that addresses the aforementioned issues simultaneously. The SPACDC scheme ensures data security during the transmission process by leveraging a proposed matrix encryption algorithm based on elliptic curve cryptography. Unlike existing coding schemes, our SPACDC scheme does not impose strict constraints on the minimum number of results required to wait for. Furthermore, the SPACDC scheme provides information-theoretic privacy protection for raw data. Finally, extensive performance analysis is provided to demonstrate the effectiveness of the proposed SPACDC scheme.

Keywords: Coded distributed computing · Distributed system · Security · Privacy · Stragglers · Collaborative computing

1 Introduction

In recent years, distributed computing has emerged as an effective paradigm widely applied to handle numerous computation tasks involving massive amounts of data in various fields, including big data analysis [1], signal processing [2] and machine learning (ML) [3]. Distributed computing involves dividing a computation task owned by a master into subtasks that are then assigned to multiple workers. The workers utilize their computing resources to perform the subtasks

This work was supported by National Natural Science Foundation of China (62071230).

H. Gao et al. (Eds.): CollaborateCom 2023, LNICST 562, pp. 357–374, 2024.
https://doi.org/10.1007/978-3-031-54528-3_20

and subsequently return the subtask results to the master. The waiting time for the master to receive all subtask results is known as computation latency. Leveraging multiple workers to perform the parallel computation of the subtasks significantly reduces the computation latency [4]. Therefore, distributed computing has received growing attention, especially in large-scale computing scenarios [5].

However, a critical performance bottleneck in traditional distributed computing systems is the requirement of waiting for computation results from all workers. Consequently, the extremely-slow or faulty workers become the "stragglers", inevitably leading to unpredictable latency [6]. Coded distributed computing (CDC) is proposed as a solution to alleviate the impact of stragglers by carefully embedding "computation redundancy" into computation tasks through coding techniques. With CDC, the master can recover the final result successfully even in the presence of missing results from some workers. This motivates numerous studies on CDC for large-scale distributed computing systems, which suffers from the effects of stragglers [7–12].

Recently, another emerging issue of data privacy is highlighted by the existence of colluding workers in CDC systems. Colluding workers refer to some curious worker nodes, which collaborate with each other to access information from the assigned data. In [10], the authors considered the data privacy issue on high-dimensional matrix multiplication in CDC systems by adding random matrices. However, the recovery threshold proves to be too large. Moreover, most of the existing works primarily focus on data security at workers, while ignoring the concern of data transmission security. In practical scenarios, data is susceptible to eavesdropping during the transmission process [13]. Therefore, ensuring secure data transmission in CDC systems is imperative.

In this paper, we propose a secure and private approximated coded distributed computing (SPACDC) scheme based on elliptic curve cryptography (ECC) to jointly address the aforementioned issues. To the best of our knowledge, this is the first proposed work that addresses secure data transmission in CDC systems. The main contributions of this paper are listed as follows:

- We design a secure and private approximated coded distributed computing (SPACDC) scheme, which provides resiliency against stragglers, and enables information-theoretic privacy protection. Particularly, the SPACDC scheme imposes no strict constraints on the minimum number of required results returned from workers.
- We propose a matrix encryption algorithm based on ECC for CDC systems, which is able to guarantee data security during the transmission process.
- Then, the theoretical proof for the information-theoretic privacy of input data guaranteed by the SPACDC scheme is demonstrated.
- Finally, extensive performance analysis further demonstrate the low complexity of the SPACDC scheme.

The rest of the paper is organized as follows. Section 2 presents the system model and problem formulation of CDC systems. Then, in Sect. 3, we introduce

the proposed matrix encryption algorithm. In Sect. 4, we describe the proposed SPACDC scheme. Finally, we conclude this paper in Sect. 5.

The notations frequently used in this paper are listed in Table 1.

2 System Model and Problem Formulation

In this section, we first describe a distributed system model with encrypted communication. Then, we introduce some relevant definitions used in this paper. Finally, we formulate the secure and private large-scale computing problem in CDC systems.

Table 1. Main notations in this paper

Notation	Description
$\mathbf{C}_i$	Ciphertext of $\mathbf{X}_i$
G	A generator point
K	Number of submatrix blocks
$\mathcal{K}$	set of indexes of the fastest k^* workers
$\mathbf{M}_i$	A confidential matrix of worker W_i
N	Number of workers
$\mathcal{N}$	set of indexes of the N workers
$\mathcal{P}$	Set of colluding workers
pk_M	Public key of the master
pk_{W_i}	Public key of worker W_i
$\mathcal{S}$	Set of stragglers
S	Number of stragglers
sk_M	Private key of the master
sk_{W_i}	Private key of worker W_i
s_{K_i}	Share key
T	Number of colluding workers
W_i	Worker node with index i
$\mathbf{X}$	Input large-scale dataset
$\tilde{\mathbf{X}}_i$	Encoded submatrix of $\mathbf{X}$ assigned to worker W_i
$\mathbf{Y}$	Final result
$\tilde{\mathbf{Y}}_i$	Computed subresult of worker W_i
$\mathbf{Z}_i$	Random matrix

2.1 System Model

We consider a master-worker distributed computing system consisting of a master and N worker nodes W_i for $i \in \mathcal{N}$. We assume that there are S straggling

workers and T colluding workers among N worker nodes. Let $\mathcal{P}$ be the set of the indexes of the T colluding workers. The master is interesting in approximately evaluating a multivariate polynomial $f : \mathbb{V} \to \mathbb{U}$ over a large-scale dataset $\mathbf{X}$, i.e., $\mathbf{Y} \approx f(\mathbf{X})$, where $\mathbb{V}$ and $\mathbb{U}$ are two real matrix spaces, $\mathbf{X} \in \mathbb{F}^{m \times d}$, m and d are positive integer. $\mathbb{F}$ denotes a sufficiently large field. Particularly, our system may encounter the presence of eavesdroppers during data transmission. As shown in Fig. 1, we describe the whole computation process by the following steps.

1) ***Data Process:*** The master receives the input dataset $\mathbf{X}$. Then, the dataset $\mathbf{X}$ is encoded by N encoding functions into N encoded matrices, is given by

$$\tilde{\mathbf{X}}_i = \mathrm{g}_i(\mathbf{X}) \text{ for } i \in \mathcal{N}. \tag{1}$$

where $\mathcal{N} \triangleq \{0, 1, \ldots, N-1\}$, $\tilde{\mathbf{X}}_i \in \mathbb{F}^{\frac{m}{K} \times d}$, and K is a positive integer. Encoding function g_i such that $\mathrm{g}_i : \mathbb{F}^{m \times d} \to \mathbb{F}^{\frac{m}{K} \times d}$. To protect data security, the master encrypts encoded matrix $\tilde{\mathbf{X}}_i$ into ciphertext $\mathbf{C}_i$ using the proposed encryption Algorithm 1. After that, the master sends $\mathbf{C}_i$ to worker W_i for $i \in \mathcal{N}$.
2) ***Task Computing:*** Each worker W_i decrypts the received encrypted data $\mathbf{C}_i$ to obtain the original data $\tilde{\mathbf{X}}_i$ by its private key. Then, worker W_i performs the computational task $\tilde{\mathbf{Y}}_i = f(\tilde{\mathbf{X}}_i)$. After completing the task, worker W_i encrypts the computed result $\tilde{\mathbf{Y}}_i$ into ciphertext $\tilde{\mathbf{C}}_i$ by Algorithm 1 and then return it back to the master. Note that some workers may fail to complete the task or return $\tilde{\mathbf{Y}}_i$ to the master slower than others.
3) ***Result Recovering:*** The master collects the encrypted data $\{\tilde{\mathbf{C}}_i\}_{i \in \boldsymbol{\mathcal{K}}}$ returned from the fastest $k^* = |\mathcal{K}|$ ($\mathcal{K} \subseteq \mathcal{N}$) workers. Then, the master decrypts the encrypted data $\{\tilde{\mathbf{C}}_i\}_{i \in \boldsymbol{\mathcal{K}}}$ to obtain original computed results $\{\tilde{\mathbf{Y}}_i\}_{i \in \boldsymbol{\mathcal{K}}}$ by Algorithm 1. After then, the master recovers the final result $\mathbf{Y}$ by using decoding functions $\{\hbar_i\}_{i \in \mathcal{K}}$, is given by

$$\mathbf{Y} = \hbar_{\mathcal{K}}\left(\{\tilde{\mathbf{Y}}_i\}_{i \in \mathcal{K}}\right). \tag{2}$$

2.2 Related Definitions

Definition 1 ***(Recovery Threshold** [8]**).*** *The recovery threshold is the minimum number of returned results of worker nodes which can be utilized to decode the final desired result. The lower the recovery threshold, the fewer subtask results returned by the worker nodes are required for the master to recover the final desired result.*

Definition 2 ***(Straggling Workers** [14]**).*** *The straggling workers are also called stragglers, refer to some worker nodes fail to compute the subtasks or return computed results to the master extremely slowly.*

Definition 3 ***(Colluding Workers** [15]**).*** *The colluding workers denote some worker nodes are curious and honest. They collude with each other to obtain information from assigned data from the master.*

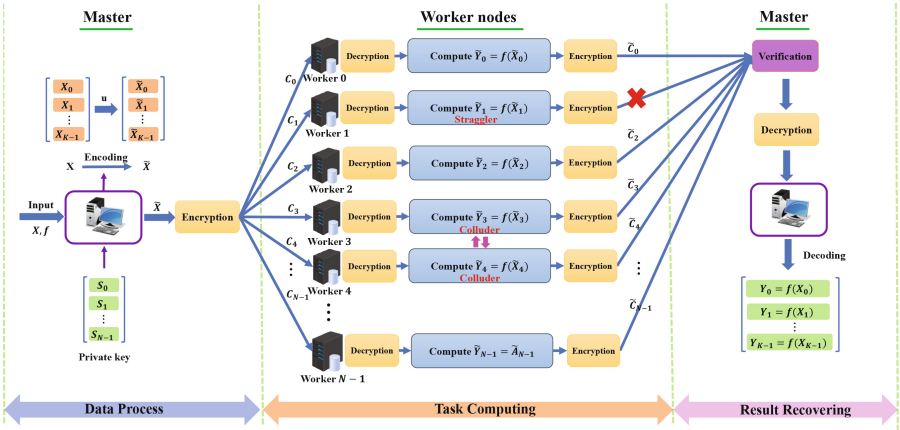


Fig. 1. An overview of the coded distributed computing system using elliptic curve cryptography is shown. In the system, the master want to evaluate a multivariate polynomial over a dataset $\mathbf{X} = [\mathbf{X}_0^T, \mathbf{X}_1^T, ..., \mathbf{X}_{K-1}^T]^T$. Firstly, the master encodes the sub-matrices $\{\mathbf{X}_i\}_{i=0}^{K-1}$ into $\{\tilde{\mathbf{X}}_i\}_{i=0}^{N-1}$. To provide secure data transmission, each encoded sub-matrices $\tilde{\mathbf{X}}_i$ is encrypted as $\tilde{\mathbf{C}}_i$ and then sent to worker W_i for $i \in \mathcal{N}$. Each worker W_i computes $\tilde{\mathbf{Y}}_i = f(\tilde{\mathbf{X}}_i)$ and encrypts the computed result before sending it back to the master node. The master decrypts the returned result $\tilde{\mathbf{C}}_i$ to obtain the result $\tilde{\mathbf{Y}}_i$. Finally, the master computes $\mathbf{Y} = \hbar_{\mathcal{K}}\big(\{\tilde{\mathbf{Y}}_i\}_{i\in\mathcal{K}}\big)$ to obtain approximated $f(\mathbf{X}_i)$ for $i \in \mathcal{N}$.

Definition 4 ***(Privacy Constraint** [10]**).*** *The privacy constraint of the CDC system for encoded data $\tilde{\mathbf{X}}_{\mathcal{P}}$ that assigned to colluding worker W_i for $i \in \mathcal{P}$ is specified by:*

$$\mathbf{I}(\tilde{\mathbf{X}}_{\mathcal{P}}; \mathbf{X}) = 0, \tag{3}$$

where $\mathbf{I}(\bullet;\bullet)$ denotes mutual information.

Definition 5 ***(Elliptic Curve** [16]**).*** *The elliptic curve over finite field $\mathbb{F}_q$ is the set of solutions of the following on-singular Weierstrass equation:*

$$y^2 = x^3 + ax + b, \tag{4}$$

where a and $b \in \mathbb{F}$ are the coefficients of the elliptic curve, and satisfies:

$$4a^3 + 27b^2 \neq 0. \tag{5}$$

Definition 6 ***(Berrut's Rational Interpolation** [7,8,17]**).*** *For a function f, n distinct points $a < x_0 < x_1 < \cdots < x_{n-1} < b$ and the corresponding evaluations $f_0, f_1, \ldots, f_{n-1}$. The Berrut's rational interpolant of f can be written in the form*

$$r(x) \triangleq \sum_{i=0}^{n-1} f_i l_i(x), \tag{6}$$

where $l_i(x)$ *for* $i \in \{0, 1, \ldots, n-1\}$ *are the basic functions are defined as*

$$l_i(x) \triangleq \frac{\frac{(-1)^i}{x-x_i}}{\sum_{i=0}^{n-1} \frac{(-1)^i}{x-x_i}}. \tag{7}$$

2.3 Problem Formulation

In this paper, we consider the problem of coded distributed computing in a master-worker distributed architecture consisting of a master nodes and N worker nodes. Among the N worker nodes, there are S straggling workers and T colluding workers. Notably, communication links between the master and workers are susceptible to eavesdropping, leading to unreliable communication. The presence of stragglers and colluding workers inevitably cause high computation latency and privacy issues, respectively.

To address these critical issues, we aim to design a secure and private approximated coded distributed computing (SPACDC) scheme using elliptic curve cryptography (ECC), which can ensures data security during the transmission process while reducing the recovery threshold and guaranteeing (information-theoretic) privacy protection of data.

3 Matrix Encryption Algorithm Based on ECC

In this section, we propose a matrix encryption algorithm based on ECC for CDC systems, abbreviated as MEA-ECC. First, we introduce ECC, which is a cost-efficient and high security encryption technique.

3.1 Background of ECC

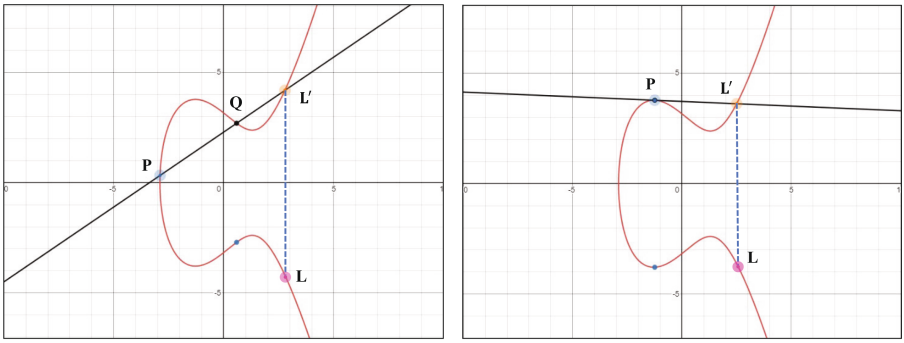
ECC is a type of asymmetric public key encryption algorithm that based on elliptic curve theory. It uses the coordinate points of the elliptic curve over finite field $\mathbb{F}_q$ for cryptographic operations. The elliptic curve equation (4) over a finite field is specified by:

$$y^2 = \{x^3 + ax + b\} \bmod \{q\}, \tag{8}$$

which is called Weierstrass equation, the discriminant:

$$\{4a^3 + 27b^2\} \bmod \{q\} \neq 0. \tag{9}$$

ECC offers two fundamental mathematical operations: point addition and point multiplication. These operations take one or two distinct points on the curve as inputs and generate a new point on the same curve as output. As illustrated in Fig. 2, we given the graphical representation of the point addition and doubling operations.



(a) Point addition (b) Point double

Fig. 2. Two visualized examples of point addition and double operations on elliptic curve $y^2 = x^3 - 5x + 10$. (a) Point addition: given two distinct points, P and Q, draw a line passing through them and intersecting the curve at another point, L'. Then, draw a parallel line to the y-axis passing through L', which intersects the curve at point L. (b)Point double: given a points, P, draw a tangent line passing through P and intersecting the curve at another point, L'. Then, draw a parallel line to the y-axis passing through L', which intersects the curve at point L.

1) ***Point Addition or Doubling:*** For two points, $P(x_1, y_1)$ and $Q(x_2, y_2)$ on the elliptic curve, the addition of the two points P and Q is defined as $P+Q = L(x_3, y_3)$, where

$$x_3 = \{\lambda^2 - x_1 - x_2\} \bmod \{q\}, \tag{10}$$

$$y_3 = \{\lambda(x_1 - x_3) - y_1\} \bmod \{q\}, \tag{11}$$

and

$$\lambda = \begin{cases} \{\frac{y_2 - y_1}{x_2 - x_1}\} \bmod \{q\}, & \text{if } P \neq Q; \\ \{\frac{3x_1^2 + a}{2y_1}\} \bmod \{q\}, & \text{if } P = Q. \end{cases} \tag{12}$$

2) ***Point Multiplication:*** For any point P on the elliptic curve and an integer n, the point multiplication $n \cdot P$ is defined as

$$n \cdot P = \underbrace{P + P + \cdots + P}_{n \text{ times}}, \tag{13}$$

3.2 Matrix Encryption Algorithm Based on ECC

In this section, we introduce the proposed MEA-ECC. The master wants to securely send a confidential matrix $\mathbf{M}_i \in \mathbb{F}^{m \times d}$ to worker W_i for $i \in \mathcal{N}$. The detailed encryption process of the MEA-ECC is described as follows:

1) ***Key generation:*** The master randomly generates an integer $sk_M < q$ as his private key, and then computes the public key $pk_M = sk_M \cdot G$, where G is a generator point. Similarly, worker W_i generates a private key $sk_{W_i} < q$ and a public key $pk_{W_i} = sk_{W_i} \cdot G$, respectively.

Algorithm 1: Matrix Encryption Algorithm

Input: $G, q, k, \mathbf{M}_i$
Output: $\mathbf{C}_i$

[I] **Key generation:** the master generates a private key $sk_M < q$;
The master computes the public key: $pk_M = sk_M \cdot G$;
for $i = 0 : N-1$ **do**
  Worker W_i generates a private key $sk_{W_i} < q$;
  Worker W_i computes the public key: $pk_{W_i} = sk_{W_i} \cdot G$;
end
[II] **Key Exchanging:** the master and worker W_i exchange their public keys pk_M and pk_{W_i};
for $i = 0 : N-1$ **do**
  The master computes share key $s_{K_i} = sk_M \cdot pk_{W_i}$;
  Worker W_i computes share key $s'_{K_i} = sk_{W_i} \cdot pk_M$;
end
$s_{K_i} = sk_M \cdot pk_{W_i} = sk_M(sk_{W_i} \cdot G) = sk_{W_i}(sk_M \cdot G) = sk_{W_i} \cdot pk_M = s'_{K_i}$;
We have $s_K = s'_K$;
[III] **Encryption:** the master generates a random integer k where $1 < k < q$;
Ciphertext point: $\mathbf{C}_i = \{k \cdot G,\ \mathbf{M}_i + \Psi(k \cdot pk_{W_i})\mathbf{I}_{m,d}\}$;
for $i = 0 : N-1$ **do**
  The master sends ciphertext $\mathbf{C}_i$ to worker W_i;
end
[IV] **Decryption:** Worker W_i receives $\mathbf{C}_i$ from the master;
Worker W_i decrypts ciphertext $\mathbf{C}_i$ with private key sk_{W_i};
for $i = 0 : N-1$ **do**
  Decryption: $\mathbf{M}_i + \Psi(k \cdot pk_{W_i})\mathbf{I}_{m,d} - \Psi[sk_{W_i}(k \cdot G)]\mathbf{I}_{m,d} = \mathbf{M}_i + \Psi[k(sk_{W_i} \cdot G) - sk_{W_i}(k \cdot G)]\mathbf{I}_{m,d} = \mathbf{M}_i$;
end
Return: $\mathbf{M}_i$

2) ***Key Exchanging:*** The master obtains the share key by computing $s_{K_i} = sk_M \cdot pk_{W_i}$. Similarly, worker W_i obtains the share key by computing $s'_{K_i} = sk_{W_i} \cdot pk_M$. It can be observed that $s_{K_i} = sk_M \cdot pk_{W_i} = sk_M(sk_{W_i} \cdot G) = sk_{W_i}(sk_M \cdot G) = sk_{W_i} \cdot pk_M = s'_{K_i}$.
3) ***Encryption:*** The master sends confidential matrix $\mathbf{M}_i$ to worker W_i for $i \in \mathcal{N}$. The ciphertext point is given by $\mathbf{C}_i = \{k \cdot G,\ \mathbf{M}_i + \Psi(k \cdot pk_{W_i})\mathbf{I}_{m,d}\}$, where function $\Psi(x, y) = x$, i.e., returns the value of the x-coordinate as the output. $\mathbf{I}_{m,d} \in \mathbb{F}^{m \times d}$ denotes the matrix with all elements being 1, and k is a random integer while satisfying $1 < k < q$.
4) ***Decryption:*** Worker W_i decrypts ciphertext $\mathbf{C}_i$ by computing $\mathbf{M}_i + \Psi(k \cdot pk_{W_i})\mathbf{I}_{m,d} - \Psi[sk_{W_i}(k \cdot G)]\mathbf{I}_{m,d} = \mathbf{M}_i + \Psi[k(sk_{W_i} \cdot G) - sk_{W_i}(k \cdot G)]\mathbf{I}_{m,d} = \mathbf{M}_i$.

The specific procedure of matrix encryption algorithm is given in Algorithm 1.

4 Secure and Private Approximated Distributed Computing Using ECC

In this section, we propose a secure and private approximated coded distributed computing scheme based on ECC, called SPACDC scheme. Before providing a general description, we firstly present an illustrating example to reveal the core idea of the SPACDC scheme.

4.1 Illustrating Example

Consider the function $f(\mathbf{X}_i) = \mathbf{X}_i\mathbf{X}_i^T$ in a distributed computing system consisting of a master node and $N = 8$ worker nodes. We set $K = 2$, $S = 1$, and $T = 1$. The master evenly divides the input matrix $\mathbf{X}$ into $K = 2$ submatrices as follows:

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_0' \\ \mathbf{X}_1' \end{bmatrix}, \tag{14}$$

where $\mathbf{X}_0'$ and $\mathbf{X}_1'$ both are with dimension of $\frac{m}{2} \times d$. Thus, the goal of the master is to obtain the computed results of $f(\mathbf{X}_0')$ and $f(\mathbf{X}_1')$.

The master encodes the input data $\mathbf{X} = \begin{bmatrix} \mathbf{X}_0' \\ \mathbf{X}_1' \end{bmatrix}$ using the following encoding function.

$$u'(z) = \frac{1}{(z-1)\phi(z)}\mathbf{X}_0' - \frac{1}{(z-2)\phi(z)}\mathbf{X}_1' + \frac{1}{(z-3)\phi(z)}\mathbf{Z}_0', \tag{15}$$

where $\phi(z) = \frac{1}{z-1} - \frac{1}{z-2} + \frac{1}{z-3}$, $\mathbf{Z}_0' \in \mathbb{F}^{\frac{m}{2} \times d}$ is a random matrix and generated independently of $\mathbf{X}$ by the master. All elements of $\mathbf{Z}_0'$ are selected uniformly independent and identically distributed (i.i.d.) from field $\mathbb{F}$. It can be noted that $u'(1) = \mathbf{X}_0'$, $u'(2) = \mathbf{X}_1'$ and $u'(3) = \mathbf{Z}_0'$. The master selects 8 distinct values $\{\alpha_i'\}_{i=0}^7$ from field $\mathbb{F}$ such that $\{\alpha_i'\}_{i=0}^7 \cap \{1, 2, 3\} = \varnothing$. Then, we obtain the encoded data $\{\tilde{\mathbf{X}}_i' = u'(\alpha_i')\}_{i=0}^7$ by employing encoding function (15).

To protect data security during the transmission process, the master encrypts the encoded data $\{\tilde{\mathbf{X}}_i' = u(\alpha_i)\}_{i=0}^7$ into ciphertext $\{\mathbf{C}_i'\}_{i=0}^7$ by the proposed MEA-ECC. Then, the master assigns encrypted data $\mathbf{C}_i'$ to worker W_i, for $i \in \{0, 1, 2, ..., 7\}$. When the worker W_i receives the encrypted data $\mathbf{C}_i'$, it obtains the original encoded data $\tilde{\mathbf{X}}_i'$ by decrypting $\mathbf{C}_i'$ using the MEA-ECC.

Then, the worker W_i performs the assigned computation task $\tilde{\mathbf{Y}}_i' = f(\tilde{\mathbf{X}}_i')$. Before returning the computed result back to the master, the worker node should encrypt the result $\tilde{\mathbf{Y}}_i'$ into $\tilde{\mathbf{C}}_i'$ by the MEA-ECC. This encryption step is essential to ensure the confidentiality and security of the computed result during the transmission process. Upon receiving the encrypted data $\tilde{\mathbf{C}}_i'$ from the worker W_i, the master decrypts it by MEA-ECC to obtain the original computed result $\tilde{\mathbf{Y}}_i'$. Let $\mathcal{F}'$ be the set of the indexes of the fastest workers that return the computed results back to the master. Hence, we have the interpolation points $(\alpha_i', \tilde{\mathbf{Y}}_i')$ for $i \in \mathcal{F}'$. After then, the master constructs a decoding function $\hbar(z)$

using Berrut's rational interpolant [17], as follows:

$$\hbar(z) = \sum_{i \in \mathcal{F}'} \frac{\frac{(-1)^i}{z-\alpha'_i}}{\sum_{j \in \mathcal{F}'} \frac{(-1)^j}{z-\alpha'_j}} \tilde{\mathbf{Y}}'_i. \tag{16}$$

Having this decoding function, the master approximately computes $f(\mathbf{X}'_0)$ and $f(\mathbf{X}'_1)$ by substituting $z = 1$ and $z = 2$ into Eq. (16), respectively. Hence, the master completes the computation task.

4.2 General SPACDC Scheme Design

In this section, we present the general description of the proposed SPACDC scheme. The SPACDC scheme is implemented in five steps that are described as follows:

1) Data Process: The master divides the large-scale matrix $\mathbf{X} \in \mathbb{F}^{m \times d}$ into K equal-sized blocks by row, i.e.,

$$\mathbf{X} = [\mathbf{X}_0^T, \mathbf{X}_1^T, ..., \mathbf{X}_{K-1}^T]^T, \tag{17}$$

where $\mathbf{X}_i \in \mathbb{F}^{\frac{m}{K} \times d}$. K is a positive integer and the final block may be padded with zeros if m is not divisible by K. Then, the goal of the master is to approximately compute $\mathbf{Y}_i \approx f(\mathbf{X}_i)$ for $i \in \mathcal{K}$.

To provide data privacy against the colluding workers, the master selects $K + T$ distinct values $\beta_1, \beta_2, \ldots, \beta_{K+T-1}$ from $\mathbb{F}$ and encodes submatrices $\{\mathbf{X}_i\}_{i=0}^{K-1}$ using the following encoding function:

$$u(z) = \sum_{i=0}^{K-1} \frac{(-1)^i}{(z-\beta_i)L(z)} \mathbf{X}_i + \sum_{i=K}^{K+T-1} \frac{(-1)^i}{(z-\beta_i)L(z)} \mathbf{Z}_i, \tag{18}$$

where $L(z) = \sum_{j=0}^{K+T-1} \frac{(-1)^j}{z-\beta_j}$. $\{\mathbf{Z}_i\}_{i=K}^{K+T-1} \in \mathbb{F}^{\frac{m}{K} \times d}$ are random matrices and generated independently of $\mathbf{X}$ by the master. All elements of $\{\mathbf{Z}_i\}_{i=K}^{K+T-1}$ are selected uniformly i.i.d. from field $\mathbb{F}$. The master selects N distinct values $\{\alpha_i\}_{i=0}^{N-1}$ from $\mathbb{F}$ such that $\{\alpha_i\}_{i=0}^{N-1} \cup \{\beta_i\}_{i=0}^{K+T-1} = \varnothing$. Thus, the master obtains the encoded data $\tilde{\mathbf{X}}_i = u(\alpha_i)$ for $i \in \mathcal{N}$. Moreover, it can be verified that $u(\beta_i) = \mathbf{X}_i$ for $i \in \{0, 1, \ldots, K-1\}$.

To ensure the data security during the transmission process, the master encrypts the encoded data $\{\tilde{\mathbf{X}}_i\}_{i=0}^{N-1}$ into ciphertext $\{\mathbf{C}_i\}_{i=0}^{N-1}$ by the proposed MEA-ECC. Then, the master sends encrypted data $\mathbf{C}_i$ to worker W_i for $i \in \mathcal{N}$.

2) Task Computing: After receiving the encrypted data $\mathbf{C}_i$ from the master, worker W_i first decrypts the encrypted data $\mathbf{C}_i$ to obtain the original encoded data $\tilde{\mathbf{X}}_i$ by the MEA-ECC. Then, the worker W_i computes the product $\tilde{\mathbf{Y}}_i = f(\tilde{\mathbf{X}}_i)$. After obtaining the computed result, worker W_i encrypts the result $\tilde{\mathbf{Y}}_i$ into ciphertext $\tilde{\mathbf{C}}_i$ and return back to the master.

3) Result Recovering: The master waits and collects returned encrypted data $\tilde{\mathbf{C}}_i$ from worker W_i for $i \in \mathcal{N}$. By using the MEA-ECC, the master decrypts

Table 2. Comparison of complexity in six coding schemes

Coded Scheme	Encoding Complexity	Decoding Complexity	Communication Complexity		Computational Complexity	Data Security	Data Privacy
			Master to all workers	Workers to master			
Polynomial Codes [9]	$\mathcal{O}(mdN)$	$\mathcal{O}(m^2 \log^2 K^2 \log\log K^2)$	$\mathcal{O}(mdN/K)$	$\mathcal{O}(m^2)$	$\mathcal{O}(dm^2/K^2)$	No	No
MatDot Codes [18]	$\mathcal{O}(mdN)$	$\mathcal{O}(Km^2 \log^2 K \log\log K)$	$\mathcal{O}(mdN/K)$	$\mathcal{O}(Km^2)$	$\mathcal{O}(dm^2/K)$	No	No
SecPoly Codes [10]	$\mathcal{O}(mdN)$	$\mathcal{O}(m^2 \log^2 K^2 \log\log K^2)$	$\mathcal{O}(mdN/K)$	$\mathcal{O}(m^2)$	$\mathcal{O}(dm^2/K^2)$	No	Yes
BACC scheme [7]	$\mathcal{O}(mdN)$	$\mathcal{O}(\lvert\mathcal{F}\rvert)$	$\mathcal{O}(mdN/K)$	$\mathcal{O}(m^2/(\lvert\mathcal{F}\rvert K^2))$	$\mathcal{O}(dm^2/K^2)$	No	No
LCC scheme [11]	$\mathcal{O}(mdN)$	$\mathcal{O}(m^2 \log^2 K \log\log K)$	$\mathcal{O}(mdN/K)$	$\mathcal{O}(m^2/K)$	$\mathcal{O}(dm^2/K^2)$	No	Yes
SPACDC (Our Scheme)	$\mathcal{O}(mdN)$	$\mathcal{O}(\lvert\mathcal{F}\rvert)$	$\mathcal{O}(mdN/K)$	$\mathcal{O}(m^2/(\lvert\mathcal{F}\rvert K^2))$	$\mathcal{O}(dm^2/K^2)$	Yes	Yes

Algorithm 2: SPACDC Algorithm

Input: $\mathbf{X}, N, K, T, m, d$
Output: $\{\mathbf{Y}_i\}_{i=0}^{K-1}$

[I] **Data process:** the master encodes $\mathbf{X}$;
The master divides $\mathbf{X}$ into K submatrices, as shown in Eq. (17);
for $i = 0 : N-1$ **do**
  $\tilde{\mathbf{X}}_i = \sum_{i=0}^{K-1} \frac{(-1)^i}{(z-\beta_i)L(z)}\mathbf{X}_i + \sum_{i=K}^{K+T-1} \frac{(-1)^i}{(z-\beta_i)L(z)}\mathbf{Z}_i$;
  The master encrypts $\tilde{\mathbf{X}}_i$ into $\tilde{\mathbf{C}}_i$ by Algorithm 1;
  The master sends $\tilde{\mathbf{C}}_i$ to worker W_i;
end
[II] **Task Computing:** worker W_i computes $\tilde{\mathbf{Y}}_i = f(\tilde{\mathbf{X}}_i)$;
for $i = 0 : N-1$ **do**
  if W_i has received $\tilde{\mathbf{C}}_i$ from the master **then**
    Worker W_i obtain $\tilde{\mathbf{X}}_i$ by decrypting $\tilde{\mathbf{C}}_i$ using Algorithm 1;
    Worker W_i computes $\tilde{\mathbf{Y}}_i = f(\tilde{\mathbf{X}}_i)$;
    Worker W_i encrypts $\tilde{\mathbf{Y}}_i$ into $\tilde{\mathbf{C}}_i$ by Algorithm 1;
    Worker W_i sends $\tilde{\mathbf{C}}_i$ back to the master;
  end
end
[III] **Result Recovering:** the master recovers $\mathbf{Y}_i$;
The master decrypts returned data $\tilde{\mathbf{C}}_i$;
The master collects points $(\alpha_i, f(u(\alpha_i)))$ for $i \in \mathcal{F}$;
Constructing a rational function $h(z) = \sum_{i\in\mathcal{F}} \frac{\frac{(-1)^i}{z-\alpha_i}}{\sum_{j\in\mathcal{F}} \frac{(-1)^j}{z-\alpha_j}} f(u(z))$;
The master computes $\mathbf{Y}_i = f(\mathbf{X}_i) \approx h(\beta_i)$ for $i \in \{0, 1, \ldots, K-1\}$;
Return: $\mathbf{Y}_i$

$\tilde{\mathbf{C}}_i$ to obtain the computed result $\tilde{\mathbf{Y}}_i$. We set $\mathcal{F}$ as the indexes of the faster workers that return the computed results back to the master. Then, we construct a rational function $h(z)$ to approximately interpolates $f(u(z))$ by received points

$(\alpha_i, f(u(\alpha_i)))$ for $i \in \mathcal{F}$, is given by

$$h(z) = \sum_{i \in \mathcal{F}} \frac{\frac{(-1)^i}{z-\alpha_i}}{\sum_{j \in \mathcal{F}} \frac{(-1)^j}{z-\alpha_j}} f(u(z)). \tag{19}$$

Thus, the master is able to obtain approximated results $\mathbf{Y}_i = f(\mathbf{X}_i) \approx h(\beta_i)$ for $i \in \{0, 1, \ldots, K-1\}$.

The specific procedure of the SPACDC scheme is given in Algorithm 2.

5 Result and Complexity Analyses

In this section, we present our main result on information-theoretic privacy of the proposed SPACDC scheme. Then, we give the complexity analysis about the RSPCC scheme.

5.1 Some Theorems of the SPACDC Scheme

Theorem 1. *For a CDC system, each worker W_i for $i \in \mathcal{P}$ ($\mathcal{P} \subseteq \mathcal{N}$) cannot obtain any information about matrix $\mathbf{X}$ from encoded matrices $\tilde{\mathbf{X}}_i$, i.e., the* privacy constraint (3) *is satisfied.*

Proof. We prove Theorem 1 in a manner similar to [15]. For any worker W_i for $i \in \mathcal{N}$, we have the mutual information

$$\begin{aligned}
\mathbf{I}(\mathbf{X}; \tilde{\mathbf{X}}_{\mathcal{P}}) &= \mathbf{H}(\tilde{\mathbf{X}}_{\mathcal{P}}) - \mathbf{H}(\tilde{\mathbf{X}}_{\mathcal{P}}|\mathbf{X}) && \text{(20a)}\\
&= \mathbf{H}(\tilde{\mathbf{X}}_{\mathcal{P}}) - \mathbf{H}(\tilde{\mathbf{X}}_{\mathcal{P}}|\mathbf{X}) + \mathbf{H}(\tilde{\mathbf{X}}_{\mathcal{P}}|\mathbf{X}, \mathbf{Z}_K, \mathbf{Z}_{K+1}, \ldots, \mathbf{Z}_{K+T-1}) && \text{(20b)}\\
&= \mathbf{H}(\tilde{\mathbf{X}}_{\mathcal{P}}) - \mathbf{I}(\tilde{\mathbf{X}}_{\mathcal{P}}; \mathbf{Z}_K, \mathbf{Z}_{K+1}, \ldots, \mathbf{Z}_{K+T-1}|\mathbf{X}) && \text{(20c)}\\
&= \mathbf{H}(\tilde{\mathbf{X}}_{\mathcal{P}}) - \mathbf{H}(\mathbf{Z}_K, \mathbf{Z}_{K+1}, \ldots, \mathbf{Z}_{K+T-1}|\mathbf{X})\\
&\quad + \mathbf{H}(\mathbf{Z}_K, \mathbf{Z}_{K+1}, \ldots, \mathbf{Z}_{K+T-1}|\mathbf{X}, \tilde{\mathbf{X}}_{\mathcal{P}}) && \text{(20d)}\\
&= \mathbf{H}(\tilde{\mathbf{X}}_{\mathcal{P}}) - \mathbf{H}(\mathbf{Z}_K, \mathbf{Z}_{K+1}, \ldots, \mathbf{Z}_{K+T-1}) && \text{(20e)}\\
&\leq \mathbf{H}(\tilde{\mathbf{X}}_{\mathcal{P}}) - \sum_{i=K}^{K+T-1} \mathbf{H}(\mathbf{Z}_i) && \text{(20f)}\\
&= \mathbf{H}(\tilde{\mathbf{X}}_{\mathcal{P}}) - T\frac{md}{K} \log |\mathbb{F}| && \text{(20g)}\\
&\leq \sum_{i=1}^{T} \mathbf{H}(\tilde{\mathbf{X}}_i) - T\frac{md}{K} \log |\mathbb{F}| && \text{(20h)}\\
&= T\frac{md}{K} \log |\mathbb{F}| - T\frac{md}{K} \log |\mathbb{F}| && \text{(20i)}\\
&= 0,
\end{aligned}$$

where (20b) follows the fact that $\tilde{\mathbf{X}}_{\mathcal{P}}$ is a deterministic function of $\mathbf{X}$ and $\{\mathbf{Z}_i\}_{i=K}^{K+T-1}$; (20e) is due to the fact that random matrices $\{\mathbf{Z}_i\}_{i=K}^{K+T-1}$ are independently generated of $\mathbf{X}$ by the master. (20f) and (20h) follow from upper bounding of the joint entropy; (20g) follows i.i.d. uniform random elements of $\{\mathbf{Z}_i\}_{i=K}^{K+T-1}$; (20i) follows from an argument similar to (20g). This completes the proof.

Theorem 2. *For a CDC system, each worker W_i for $i \in \mathcal{N}$ cannot obtain any information about the final product $\mathbf{Y}$ from computed sub-product $\tilde{\mathbf{Y}}_i$, i.e.,*

$$\mathbf{I}(\tilde{\mathbf{Y}}_i; \mathbf{Y}) = 0. \tag{21}$$

Proof. For any worker W_i for $i \in \mathcal{N}$, we have the mutual information

$$\begin{aligned}
\mathbf{I}(\mathbf{Y}; \tilde{\mathbf{Y}}_i) &= \mathbf{I}\big(f(\mathbf{X}); f(\tilde{\mathbf{X}}_i)\big) \\
&= \mathbf{H}\big(f(\mathbf{X})\big) - \mathbf{H}\big(f(\mathbf{X})|f(\tilde{\mathbf{X}}_i)\big) && \text{(22a)} \\
&\leq \mathbf{H}\big(f(\mathbf{X})\big) - \mathbf{H}\big(f(\mathbf{X})|f(\tilde{\mathbf{X}}_i), \tilde{\mathbf{X}}_i\big) && \text{(22b)} \\
&= \mathbf{H}\big(f(\mathbf{X})\big) - \mathbf{H}\big(f(\mathbf{X})|\tilde{\mathbf{X}}_i\big) && \text{(22c)} \\
&= \mathbf{H}\big(f(\mathbf{X})\big) - \mathbf{I}\big(f(\mathbf{X}); \mathbf{X}|\tilde{\mathbf{X}}_i\big) - \mathbf{H}\big(f(\mathbf{X})|\tilde{\mathbf{X}}_i, \mathbf{X}\big) && \text{(22d)} \\
&= \mathbf{H}\big(f(\mathbf{X})\big) - \mathbf{I}\big(f(\mathbf{X}); \mathbf{X}|\tilde{\mathbf{X}}_i\big) && \text{(22e)} \\
&= \mathbf{H}\big(f(\mathbf{X})\big) - \mathbf{H}(\mathbf{X}|\tilde{\mathbf{X}}_i) + \mathbf{H}\big(\mathbf{X}|\tilde{\mathbf{X}}_i, f(\mathbf{X})\big) && \text{(22f)} \\
&= \mathbf{H}\big(f(\mathbf{X})\big) - \mathbf{H}(\mathbf{X}) + \mathbf{H}\big(\mathbf{X}|f(\mathbf{X})\big) && \text{(22g)} \\
&= \mathbf{H}\big(f(\mathbf{X})\big) - \mathbf{I}\big(f(\mathbf{X}); \mathbf{X}\big) && \text{(22h)} \\
&= \mathbf{H}\big(f(\mathbf{X})|\mathbf{X}\big) && \text{(22i)} \\
&= 0, && \text{(22j)}
\end{aligned}$$

where (22b) follows from the conditioning reduces entropy; (22e) and (22j) follow from the fact that $f(\mathbf{X})$ is a deterministic function of $\mathbf{X}$; (22g) follows from (3) which has been proved in Theorem 1. This completes the proof.

5.2 Complexity Analysis of the SPACDC Scheme

In this section, we investigate the computational complexities of the proposed SPACDC scheme for encoding, decoding, checking, communication and per-worker computation.

1) ***Encoding Complexity:*** Recalling the encoding phase of the SPACDC scheme, the overall encoding complexity depends on the encoding function $h(z)$ in Eq. (18). We observe that $u(z)$ is the sum of $K+T$ matrices, each of which with dimension $\frac{m}{K} \times d$. Hence, the computational complexity of the encoding function $u(z)$ for each worker is $\mathcal{O}(md(K+T)/K) = \mathcal{O}(md)$. For N workers, the overall encoding complexity of the SPACDC scheme is $\mathcal{O}(mdN)$.

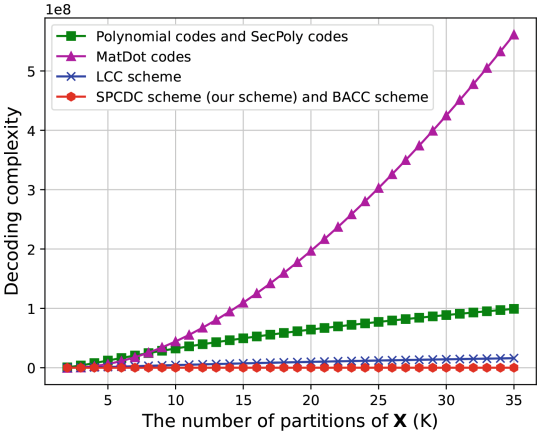


Fig. 3. Comparison of decoding complexity achieved by the Polynomial codes [9], MatDot codes [18], SecPoly codes [10], BACC scheme [7], LCC scheme [11] and SPACDC scheme (our scheme) in distributed computing systems under parameter $m = 1000$ while varying K from 1 to 36.

2) ***Decoding Complexity:*** Note that the result recovering phase of the SPACDC scheme, the master interpolates the polynomial $f(u(z))$ to decode the final result $\mathbf{Y}$ by Eq. (19). Similar to [7], the decoding complexity of the SPACDC scheme is $\mathcal{O}(|\mathcal{F}|)$.
3) ***Communication Complexity:*** The communication complexity of the SPACDC scheme includes two aspects: (1) master-to-worker and (2) worker-to-master. First, the master transmits $\mathcal{O}(md/K)$ symbols to each worker. Hence, the total number of symbols that the master transmits to N workers is $\mathcal{O}(mdN/K)$. Second, each worker returns $\mathcal{O}(m^2)$ symbols to the master. Thus, the total number of symbols that the workers return back to the master is $\mathcal{O}(m^2/|\mathcal{F}|)$.
4) ***Computational Complexity of Each Worker:*** In the task computing phase, worker W_i for $i \in \mathcal{N}$ compute the product $\tilde{\mathbf{Y}}_i = f(\tilde{\mathbf{X}}_i)$, e.g., $\tilde{\mathbf{Y}}_i = \tilde{\mathbf{X}}_i \tilde{\mathbf{X}}_i^T$, where $\mathbf{X}_i$ with dimension $\frac{m}{K} \times d$. Then, we obtain that the computational complexity of each worker is $\mathcal{O}(dm^2/K^2)$.

As shown in Table 2, we give a summary of complexity analysis in terms of encoding, decoding, computation, and communication on the proposed SPACDC scheme and the existing coding schemes (polynomial Codes [9], MatDot codes [18], SecPoly codes [10], BACC scheme [7], and LCC scheme [11]). The complexity comparisons among these coding schemes utilize identical parameter settings. From Table 2, we observe that our SPACDC scheme has the same coding complexity as the other existing schemes. It is noteworthy that the SPACDC scheme achieves security and privacy protection with nearly identical complexity as the other coding schemes. Additionally, the communication complexity of the SPACDC scheme is equal to that of other coding schemes in term of master-to-

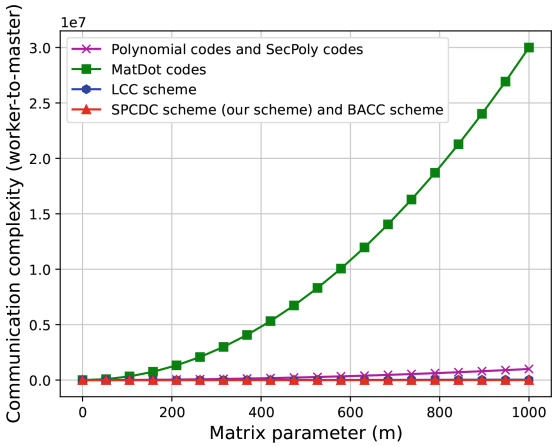


Fig. 4. Comparison of communication complexity (worker-to-master) of the Polynomial codes, MatDot codes, SecPoly codes, BACC scheme, LCC scheme and SPACDC scheme (our scheme) in distributed computing systems under parameter $K = 30$, $|\mathcal{F}| = 10$ while varying m from 1 to 1000.

worker communication. It is mainly determined by the similarity and consistency of the matrix partitioning method of these coding schemes.

Figure 3 shows a comparison of the decoding complexity among the Polynomial codes, MatDot codes, SecPoly codes, BACC scheme, LCC scheme and SPACDC scheme (our scheme) under parameter $m = 1000$ while varying K from 1 to 36. It is evident that the decoding complexities of the SPACDC and BACC schemes are smallest among these coding schemes, while MatDot codes exhibit the highest decoding complexity. The decoding complexities of the Polynomial and SecPoly codes are higher than those of the LCC, BACC and our SPACDC scheme. Both SPACDC and BACC schemes employ Berrut's rational interpolant for decoding the final result, constructing low-degree rational functions to significantly reduce computational complexity. The decoding complexity of the well-known coding scheme LCC is slightly higher than that of our scheme. As the degree of a polynomial function to be evaluated increases, the decoding complexity of LCC scheme will increase proportionally. The decoding complexity of LCC scheme will be much higher than that of our scheme due to a high polynomial degree.

In Fig. 4, we compare the communication complexity (worker-to-master) of Polynomial codes, MatDot codes, SecPoly codes, BACC scheme, LCC scheme and SPACDC scheme (our scheme) under parameter $K = 30$, $|\mathcal{F}| = 10$ while varying m from 1 to 1000. The communication complexity of the SPACDC and BACC scheme are smallest among these coding schemes. The communication complexity (worker-to-master) of MatDot codes is the highest among all coding schemes. In fact, the communication complexity (worker-to-master) is largely related to the dimension of returned result matrices.

As shown in Fig. 5, we present a comparison of the computational complexity of various coding schemes, including Polynomial codes, MatDot codes, SecPoly codes, BACC scheme, LCC scheme, and our proposed SPACDC scheme. The parameters used are $m = 5000$ and $d = 1000$, with varying values of K from 1 to 36. From the results, it is evident that our SPACDC scheme exhibits comparable computational complexity to the other coding schemes, except for MatDot codes. Notably, MatDot codes show higher computational complexity compared to the rest of the coding schemes. This discrepancy is primarily caused by the larger size of the assigned matrices in the workers when employing MatDot codes [18].

As discussed above, our SPACDC scheme exhibits significantly lower complexity compared to existing coding scheme, such as the Polynomial codes, MatDot codes, SecPoly codes, and LCC scheme. Although the SPACDC scheme has the same complexity with BACC scheme, BACC scheme is unable to provide security and privacy protection for data. Therefore, our SPACDC scheme outperforms all other coding schemes.

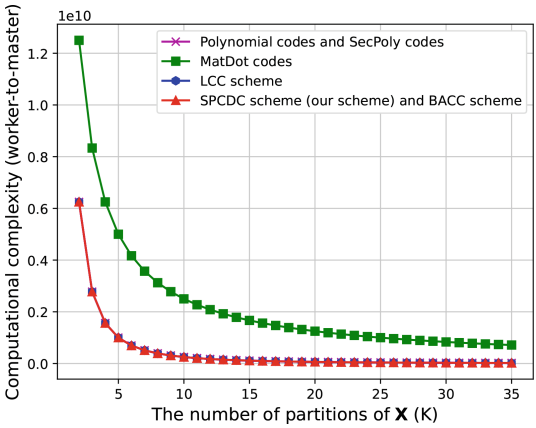


Fig. 5. Comparison of computational complexity of the Polynomial codes, MatDot codes, SecPoly codes, BACC scheme, LCC scheme and SPACDC scheme (our scheme) in distributed computing systems under parameter $m = 5000$, $d = 1000$ while varying K from 1 to 36.

6 Conclusions

In this paper, we proposed a secure and private approximated coded distributed computing (SPACDC) scheme, which jointly addressed the issues of stragglers and colluding workers. It is worth noting that the SPACDC scheme is able to ensure data security during the transmission process using the proposed matrix encryption algorithm, i.e., MEA-ECC. Then, the theoretical proofs for the information-theoretic privacy protection of input data were given. Finally, we provided comprehensive performance analysis to demonstrate the effectiveness of our SPACDC scheme.

References

1. Wu, Z., Sun, J., Zhang, Y., Wei, Z., Chanussot, J.: Recent developments in parallel and distributed computing for remotely sensed big data processing. Proc. IEEE **109**(8), 1282–1305 (2021)
2. Güler, B., Avestimehr, A.S., Ortega, A.: TACC: topology-aware coded computing for distributed graph processing. IEEE Trans. Sig. Inf. Process. Netw. **6**, 508–525 (2020)
3. Guo, Y., Zhao, R., Lai, S., Fan, L., Lei, X., Karagiannidis, G.K.: Distributed machine learning for multiuser mobile edge computing systems. IEEE J. Sel. Top. Sig. Process. **16**(3), 460–473 (2022)
4. Schlegel, R., Kumar, S., Rosnes, E., i Amat, A.G.: Privacy-preserving coded mobile edge computing for low-latency distributed inference. IEEE J. Sel. Areas Commun. **40**(3), 788–799 (2022)
5. Ng, J.S., et al.: A comprehensive survey on coded distributed computing: fundamentals, challenges, and networking applications. IEEE Commun. Surv. Tut. **23**(3), 1800–1837 (2021)
6. Yu, Q., Maddah-Ali, M.A., Avestimehr, A.S.: Straggler mitigation in distributed matrix multiplication: fundamental limits and optimal coding. IEEE Trans. Inf. Theor. **66**(3), 1920–1933 (2020)
7. Jahani-Nezhad, T., Maddah-Ali, M.A.: Berrut approximated coded computing: straggler resistance beyond polynomial computing. IEEE Trans. Pattern Anal. Mach. Intell. **45**(1), 111–122 (2023)
8. Jahani-Nezhad, T., Maddah-Ali, M.A.: Optimal communication-computation trade-off in heterogeneous gradient coding. IEEE J. Sele. Areas Inf. Theor. **2**(3), 1002–1011 (2021)
9. Yu, Q., Maddah-Ali, M.A., Avestimehr, A.S.: Polynomial codes: an optimal design for high-dimensional coded matrix multiplication. In: Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 4406–4416 (2017)
10. Yang, H., Lee, J.: Secure distributed computing with straggling servers using polynomial codes. IEEE Trans. Inf. Forensics Secur. **14**(1), 141–150 (2019)
11. Yu, Q., Li, S., Raviv, N., Kalan, S.M.M., Soltanolkotabi, M., Avestimehr, S.A.: Lagrange coded computing: optimal design for resiliency, security, and privacy. In: The 22nd International Conference on Artificial Intelligence and Statistics, pp. 1215–1225 (2019)
12. Ozfatura, E., Ulukus, S., Gündüz, D.: Coded distributed computing with partial recovery. IEEE Trans. Inf. Theor. **68**(3), 1945–1959 (2021)
13. Jiang, Y., et al.: Secure data transmission and trustworthiness judgement approaches against cyber-physical attacks in an integrated data-driven framework. IEEE Trans. Syst. Man Cybern. Syst. **52**(12), 7799–7809 (2022)
14. Byrne, E., Gnilke, O.W., Kliewer, J.: Straggler-and adversary-tolerant secure distributed matrix multiplication using polynomial codes. Entropy **25**(2), 266 (2023)
15. Aliasgari, M., Simeone, O., Kliewer, J.: Private and secure distributed matrix multiplication with flexible communication load. IEEE Trans. Inf. Forensics Secur. **15**, 2722–2734 (2020)
16. Sadhukhan, D., Ray, S., Biswas, G., Khan, M.K., Dasgupta, M.: A lightweight remote user authentication scheme for IoT communication using elliptic curve cryptography. J. Supercomput. **77**, 1114–1151 (2021)

17. Berrut, J.P.: Rational functions for guaranteed and experimentally well-conditioned global interpolation. Comput. Math. Appl. **15**(1), 1–16 (1988)
18. Dutta, S., Fahim, M., Haddadpour, F., Jeong, H., Cadambe, V., Grover, P.: On the optimal recovery threshold of coded matrix multiplication. IEEE Trans. Inf. Theor. **66**(1), 278–301 (2020)