Secure and Private Coding for Edge Computing Against Cooperative Attack with Low Communication Cost and Computational Load

Xiaotian Zou, Jin Wang(✉), Can Liu, Lingzhi Li, Fei Gu, and Guojing Li

Department of Computer Science and Technology, Soochow University, Suzhou, China
{20204227064,20214027012,20204227039}@stu.suda.edu.cn,
{wjin1985,lilingzhi,gufei}@suda.edu.cn

Abstract. Edge computing is an efficient computing paradigm, which can utilize computing devices at the edge of network to provide real-time proximity service. Since edge devices lack centralized management, they are more vulnerable to being attacked. Therefore, the issues of data security and user privacy in edge computing are particularly important. A large number of existing literature focus on the data security and user privacy with independent attackers. However, cooperative attacks, in which multiple attackers can collaborate to obtain the data content and user privacy, have not been fully investigated. In particular, we take the matrix-vector multiplication which is a basic component of most machine learning algorithms as the basic task. Therefore, in this paper, we focus on the *Secure and Privacy Matrix-vector Multiplication* (SPMM) issue for edge computing against cooperative attack and design a general coded computation scheme to achieve lowest system resource consumption, *i.e.* communication cost and computational load. Specifically, we propose two coding schemes: *Secure and Private Coding with lower communication Cost* (SPCC) and *Secure and Private Coding with lower computational Load* (SPCL). We also conduct solid theoretical analyses and extensive experiments to demonstrate that both two proposed coding schemes can achieve lower communication cost and computational load than existing work. Finally, we perform extensive analyses to the superiority of the proposed schemes.

This work is supported in part by National Natural Science Foundation of China (62072321, 61972272), Six Talent Peak Project of Jiangsu Province (XYDXX-084), China Postdoctoral Science Foundation (2020M671597), Jiangsu Postdoctoral Research Foundation (2020Z100), Suzhou Planning Project of Science and Technology (SNG2020073, SS202023, SYG202024), Tang Scholar of Soochow University, Collaborative Innovation Center of Novel Software Technology and Industrialization, and Soochow University Interdisciplinary Research Project for Young Scholars in the Humanities.

Responding author: Jin Wang, wjin1985@suda.edu.cn

H. Gao et al. (Eds.): CollaborateCom 2022, LNICST 460, pp. 213–232, 2022.
https://doi.org/10.1007/978-3-031-24383-7_12

Keywords: Edge computing · Security · Privacy · Communication cost · Computational load

1 Introduction

With the rapid arrival of the 5G era, it is unrealistic to transmit massive data to the cloud for real-time processing due to the bandwidth limitation between the cloud and the edge network [1]. Since edge devices are closer to the user, edge computing can provide efficient and real-time computing to the user [1–4]. Nowadays, with the development of artificial intelligence, edge computing has been used in extensive fields, such as *virtual reality* (VR) [2], smart city [3], autonomous vehicle [4], *etc.*, paving the way for the application of artificial intelligence [2–5].

Despite the enormous potential of edge computing, there are still many challenges [1,5–23]. The first major issue is the straggler problem [6–9]. Due to the differences in network environment and devices, edge devices return computation results at different speeds. Therefore, the user needs to wait for slower edge devices, namely stragglers, to complete the computation, which is the straggler problem. This greatly affects the calculation latency. In addition, since edge devices are at the edge of the network, they are more vulnerable to being attacked [10–23]. This seriously threatens the security of data. Therefore, the original data must be encoded before being allocated to edge devices [10–16]. Finally, due to the trustlessness of edge devices, they are more likely to leak the privacy of user when they participate in computing [17–23]. Matrix-vector multiplication, as one of the key modules of machine learning, image processing and deep learning, is the starting point to study the above problems [6–23].

To solve the above problems, researchers proposed many *coded distributed computing* (CDC) schemes in different scenarios [6–23]. In CDC, the user first divides the original data into blocks and encodes them, *i.e.* linearly combines

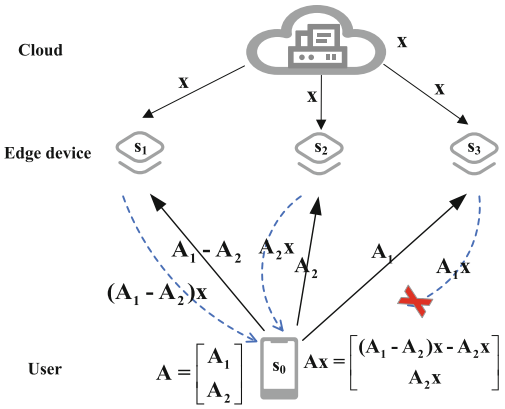
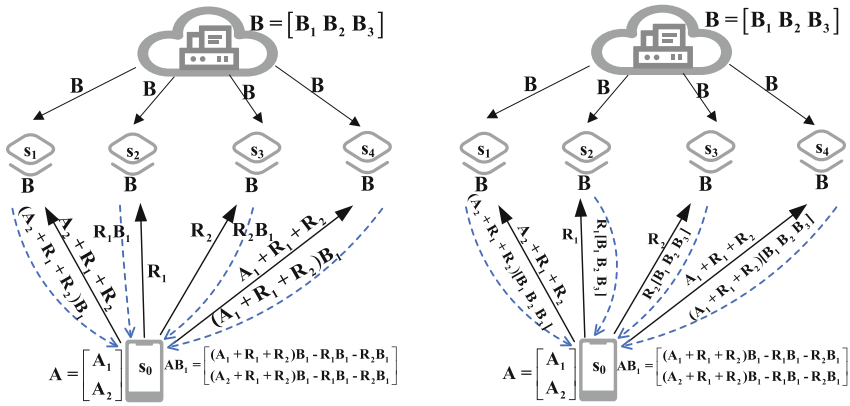


Fig. 1. Traditional distributed computing.



(a) Secure coded edge computing. (b) Secure and private coded edge computing.

Fig. 2. An example of coded edge computing.

blocks. Then, the user allocates the coded blocks to edge devices. After receiving the coded blocks, edge devices calculate and return the computation results, *i.e.* intermediate results, to the user. Finally, once the user receives sufficient intermediate results, the user can decode and obtain the final result.

Through redundant calculation of data, the recovery threshold which is the number of intermediate results required to decode the final result can be reduced [6–9]. This greatly alleviates the straggler problem. As shown in Fig. 1, when the edge devices s_1 and s_2 return the results, the user can decode the final results without waiting for slower edge device s_3. However, the above research on CDC do not consider security issues. To solve the data security problems, an increasing number of researchers begin to investigate *secure coded distributed computing* (SCDC) [10–16] to satisfy *information-theoretic security* (ITS) [24], *i.e.* edge devices cannot obtain a linear combination of the original blocks. As shown in Fig. 2(a), the user protects data security by adding random blocks to the original blocks in the coding phase. Moreover, the scheme in Fig. 2(a) can protect the data security in the case of cooperative attack by any two devices. However, edge devices are curious about whichever piece of data in public data the user operates because the user typically has a variety of computing needs. This seriously undermines the privacy of user. Consequently, a large number of researchers are dedicated to studying *private coded distributed computing* (PCDC), *i.e.* edge devices have no way of knowing which piece of public data the user wants to calculate [17–23].

Next, we give an example in Fig. 2(a) (b) to show the coded edge computing system that we study. In this example, the goal of the user is to compute $\mathbf{AB}_1$. The system we considered consists of a user device s_0, a cloud and 4 edge devices s_1, s_2, s_3, s_4, where any two edge devices can cooperate to attack. Data $\mathbf{A}$ is stored on s_0 and $\mathbf{B} = [\mathbf{B}_1\ \mathbf{B}_2\ \mathbf{B}_3]$ as a small public data is stored on the cloud. The cloud sends data $\mathbf{B} = [\mathbf{B}_1\ \mathbf{B}_2\ \mathbf{B}_3]$ to edge devices before the calculation

starts. Then, the cloud no longer communicates with edge devices. Firstly, the user divides $\mathbf{A}$ into 2 blocks by row which are $\mathbf{A}_1$, $\mathbf{A}_2$. In order to protect the security of $\mathbf{A}$, the user generates 2 random blocks $\mathbf{R}_1$, $\mathbf{R}_2$ of the same size as $\mathbf{A}_1$. Then, in secure coded edge computing, the user encodes $\mathbf{A}_1$, $\mathbf{A}_2$ with them and allocates coded blocks to edge devices as shown in Fig. 2(a). When edge devices receive a coded block of $\mathbf{A}$, the edge devices perform the calculation and return the results (intermediate results) to the user. When the 4 edge devices all return intermediate results, the final result $\mathbf{AB}_1$ can be decoded as shown in Fig. 2(a). Since the user encodes each block of $\mathbf{A}$ with two random blocks, the original information of $\mathbf{A}$ cannot be obtained when any two edge devices cooperate. However, since the user device wants to calculate $\mathbf{AB}_1$, edge devices only operate $\mathbf{B}_1$ with the coded blocks of $\mathbf{A}$. Thus, edge devices can know $\mathbf{B}_1$ is different from $\mathbf{B}_2$, $\mathbf{B}_3$ and $\mathbf{B}_1$ is the content that the user needs to calculate. Therefore, the scheme in Fig. 2(a) leaks the privacy of the user. As shown in Fig. 2(b), similar to Fig. 2(a), this scheme can protect the security of $\mathbf{A}$ when any two devices cooperate to attack. Since operations on $\mathbf{B}_1$, $\mathbf{B}_2$, $\mathbf{B}_3$ are consistent, edge devices have no way of knowing which column of $\mathbf{B}$ we want to calculate. In addition, in the case of cooperative attack by any two devices, edge devices cannot know computing goal of the user. Therefore, the scheme protects the security of $\mathbf{A}$ and privacy of user. In this paper, we proposed two secure and private coding schemes which can achieve lower communication cost and computational load than existing work when there are stragglers in the system. The contributions of this paper are as follows:

- We give the system model, attack model, privacy condition and security condition and define SPMM problem.
- We propose SPCC for the SPMM problem. In SPCC, although the communication cost of intermediate results returned by edge devices to the user is large, the communication cost of the user sending coded blocks to edge devices is improved. In addition, SPCC has lower communication cost than the existing work.
- In order to obtain a lower computational load, we further design SPCL for the SPMM problem and give the proof of security and privacy. SPCL has lower computational load than SPCC.
- Finally, we conduct extensive simulation experiments under different parameter settings, which demonstrate that SPCC can achieve lower communication cost and SPCL can achieve lower computational load compared with the existing work.

The rest of the paper is organized as follows. The related work is investigated in Sect. 2. In Sect. 3, we introduce a detailed description of the system model and define the SPMM problem. In Sect. 4, we propose SPCC and SPCL for SPMM. Besides, we perform a theoretical analysis of the two schemes that we proposed. In Sect. 5, we conduct extensive experiments to demonstrate the superiority of SPCC and SPCL. Finally, we conclude the paper in Sect. 6.

2 Related Work

Recently, a large number of researchers begin to study CDC because CDC can provide efficient computing and mitigate the impact of stragglers [6–9]. In [6], Li *et al.* proposed distributed fog computing coding scheme which can make full use of the devices scattered around the user for calculation. Lee *et al.* [7] proposed *maximum-distance-separable* (MDS) codes which can mitigate the impact of stragglers to speed up distributed machine learning. In [9], Dutta *et al.* proposed MatDot codes which reduce the recovery threshold of the system by sacrificing communication. Moreover, in [8], Soto *et al.* proposed a coding scheme that minimize the recovery threshold for batch matrix multiplication. However, none of the above schemes considers the security of the system.

Since edge devices are at the edge of the network, they are more vulnerable to being attacked. A growing number of researchers are committed to solving the problem of data security [10–14]. In [10], Yang *et al.* proposed secure polynomial coding scheme which can be applied not only to matrix-vector multiplication but also to image convolution. Besides, Bitar *et al.* [11] designed *Staircase Codes* and analysed the latency of *Staircase Codes* is always less than the latency of secret sharing. In [12], Bitar *et al.* proposed a solution based on *Staircase Codes* to minimize latency for SCDC. In [15], Cao *et al.* proposed secure coded edge computing schemes for the heterogeneous edge computing systems which can acquire the minimal total cost. Moreover, for the collusion of edge devices, a variety of secure schemes have been proposed. In [13], Chang *et al.* proposed a secure polynomial coding scheme to achieve the maximized ratio of effective blocks to total blocks for collusion attacks. In [14], Doliveira *et al.* proposed GASP codes to minimize the recovery threshold for distributed matrix multiplication with protecting both matrix security. In [16], Zhu *et al.* proposed a secure scheme in the case of collusion and minimize the total cost of the distributed matrix multiplication. However, the above schemes does not consider the privacy of the user.

For protecting the privacy, Kim *et al.* [17] proposed a privacy coding scheme by setting up special evaluation points to reduces the stragglers effect. In addition, Kim *et al.* [18] proposed a secure privacy coding scheme based on [17] to ensure security of data and privacy of the user. In [19], Chang *et al.* studied the correlation between upload and download for secure and private distributed matrix multiplication. In [20], Qian *et al.* employ entangled polynomial coding, which lowers the total computing cost and the recovery threshold for secure and private distributed matrix multiplication. Moreover, in order to deal with privacy issues in case of devices collusion, Vaidya *et al.* [21] firstly proposed a secure and privacy scheme for distributing computing in the case of collusion of edge devices. In this scheme, the privacy of the user is protected by sending random blocks, which greatly increases communication cost. Jia *et al.* [22] proposed X-secure and T-private coding scheme against the cooperative attacks through multiple rounds of calculation. Kim *et al.* [23] proposed a coding scheme to protect the privacy of two databases for collusion issues in a distributing computing system. However, in the above schemes, only [21,22] considered the secure and

private coded scheme against colluding devices, but there is no focus on the overhead of the system. Although the above studies have proposed methods for security and privacy issues in the case of collusion of edge devices, we believe that their schemes can be better optimized to achieve lower communication cost and computational load. In this paper, we focus on security and privacy issues and propose effective solutions to further reduce the communication cost and computational load.

3 Problem Modeling

In this section, firstly, we present a full description of the system model. Next, we propose our attack model, as well as security and privacy conditions. Finally, the SPMM problem is formally defined.

3.1 System Model

In this paper, we investigate an edge computing system that includes a user device, a cloud, and a large number of edge devices. Since edge devices are at the edge of the network, they lack centralized and unified management. Therefore, they are more vulnerable to being attacked. We assume any L ($L \geq 1$) edge devices can collude. Without losing generality, we focus on the matrix-vector multiplication, which is an important module for machine learning, image processing and federation learning [7–9,13–17]. In particular, data $\mathbf{A}$ ($\mathbf{A} \in \mathbb{F}_q^{t \times p}$ and $\mathbb{F}_q$ is a finite field) is the user data. Data $\mathbf{B}$ ($\mathbf{B} \in \mathbb{F}_q^{p \times n}$) is public data which is stored in the cloud. The goal of the user is to compute the multiplication of $\mathbf{A}$ and a column vector of $\mathbf{B}$, *i.e.* $\mathbf{AB}_d$ ($1 \leq d \leq n$), in the case of ensuring data security and privacy of user. Before computing, the cloud will send data $\mathbf{B}$ to edge devices[1]. After that, the cloud will not communicate with edge devices.

Firstly, the user divides $\mathbf{A}$ into m blocks according to rows[2] ($m \leq t$) to obtain $\mathbf{A} = \left[\mathbf{A}_1^\top, \mathbf{A}_2^\top, \cdots, \mathbf{A}_m^\top\right]^\top$ where $\mathbf{A}_i \in \mathbb{F}_q^{\frac{t}{m} \times p}$ ($1 \leq i \leq m$). Then, to ensure the security of $\mathbf{A}$ and optimize the system sufficiently, the user generates L random blocks $\mathbf{R} = [\mathbf{R}_1, \mathbf{R}_2, \cdots, \mathbf{R}_L]$, where $\mathbf{R}_i \in \mathbb{F}_q^{\frac{t}{m} \times p}$ ($1 \leq i \leq L$). Next, the user encodes $\mathbf{A}$ with $\mathbf{R}$ by function $f(x)$, where $f(x)$ can be shown as follows:

$$f(x) = \sum_{i=1}^{m} \mathbf{A}_i x^{\alpha_i} + \sum_{j=1}^{L} \mathbf{R}_j x^{\alpha_{m+j}}. \tag{1}$$

The value of α_i ($\forall i \in \{1, \cdots, m+L\}$) will be shown in Sect. 4. The user selects T ($T \geq 1$) non-zero values $\{x_1, \cdots, x_T\}$ to obtain the coded matrix $\widetilde{\mathbf{A}} = \left[\widetilde{\mathbf{A}}_1^\top, \widetilde{\mathbf{A}}_2^\top, \cdots, \widetilde{\mathbf{A}}_T^\top\right]^\top$, where $\widetilde{\mathbf{A}}_i = f(x_i)$, $\forall i \in \{1, \cdots, T\}$.

At this point, we define the recovery threshold:

[1] Since the data $\mathbf{B}$ can be stored on the edge device, $t >> n > 1$.

[2] When $\frac{s}{m}$ is not an integer, we add $\mathbf{0}$ vector to the row of matrix $\mathbf{A}$ to make $\frac{s}{m}$ an integer.

Definition 1 *(**Recovery threshold**). The recovery threshold N is the minimal number of intermediate results returned by the edge devices can be used to decode the final result $\boldsymbol{AB}_d$.*

To ensure that user can get $\mathbf{AB}_d$, we give the decodability condition as follows:

Definition 2 *(**Decodability Condition**). The user can obtain $\boldsymbol{AB}_d$ iff $T \geq N$ and the exponent of $\boldsymbol{A}_i\boldsymbol{B}_d$ in each intermediate result is unique $\forall i \in \{1, \cdots, m\}$.*

3.2 Attack Model, Secure Condition and Privacy Condition

In this paper, we assume that edge devices are passive attackers or controlled by passive attackers [10–24]. The security model that we consider is based on ITS [10–16,24]. Specially, we need to make sure that any L edge devices collusion can not obtain the linear combination of $\mathbf{A}_1, \cdots, \mathbf{A}_m$. Moreover, similarly, the privacy model that we consider any edge device can not obtain the relevant information of d [17–23].

Let $I(\varrho; \sigma)$ denote the mutual information between ϱ and σ. In addition, $\widetilde{\mathbf{A}}_{s_L}$ represents the set of coded blocks of $\mathbf{A}$ on any L edge devices. At this point, we define security [10–16,24] and privacy conditions [17–23].

Definition 3 *(**Security Condition**). The solution of SPMM satisfies the security condition iff*

$$I(\boldsymbol{A}; \widetilde{\boldsymbol{A}}_{s_L}) = 0. \tag{2}$$

Definition 4 *(**Privacy Condition**). The solution of SPMM satisfies the privacy condition iff any L colluding edge devices cannot obtain relevant information about d.*

3.3 Communication Cost and Computational Load

To measure communication and computation metrics, we define the communication cost and computational load [21] as follows:

Definition 5 *(**Communication Cost**). The communication cost is the number of elements that sent by the user to edge devices and returned by edge devices to the user.*

Definition 6 *(**Computational Load**). The computational load is defined as the number of multiplication operation on each edge device participating in the calculation.*

4 Secure and Private Coded Computation Schemes

In this section, we propose SPCC for SPMM. Next, we proof the security and privacy of SPCC and make theoretical analysis. Then, to get lower computational load, we propose SPCL and analyze its performance. Finally, we give an example for SPCC and SPCL, respectively.

4.1 Secure and Private Coding Scheme with Low Communication Cost (SPCC)

In this section, we first propose SPCC for SPMM. Next, we proof the security, privacy and decodability conditions of SPCC. Finally, we analyze communication cost and computational load of SPCC.

For the given m, L, the α_i is shown as follows:

$$\alpha_i = i - 1, \forall 1 \leq i \leq m + L. \tag{3}$$

In this scheme, the computation process can be shown as follows:

- Firstly, the user encodes data $\mathbf{A}$ according to Eq. (1) and Eq. (3). Then, the user allocates $\widetilde{\mathbf{A}}_i$ $(1 \leq i \leq T)$ to the i-th edge device.
- When the i-th edge device receives the coded block $\widetilde{\mathbf{A}}_i$, the edge device performs the calculation $\widetilde{\mathbf{A}}_i\mathbf{B}_1, \cdots, \widetilde{\mathbf{A}}_i\mathbf{B}_n$ and returns all results to the user.
- Once the user receives intermediate results from N_{SPCC} edge devices (N_{SPCC} is the recovery threshold of SPCC and $N_{SPCC} \leq T$), the user can decode the final results $\mathbf{AB}_d$ by using polynomial interpolation [10].

Theorem 1. *The SPCC is secure in the case of collusion of any L edge devices, i.e. the security condition Eq. (2) is satisfied.*

Proof. According to [10] and Eq. (3), it is clear that all α_i $(1 \leq i \leq m + L)$ are different. Therefore, the security condition Eq. (2) is satisfied.

Theorem 2. *The SPCC is private in the case of collusion of any L edge devices, i.e. the privacy condition is satisfied.*

Proof. Since the i-th edge device receives the coded block $\widetilde{\mathbf{A}}_i$, the edge device performs the calculation $\widetilde{\mathbf{A}}_i\mathbf{B}_1, \cdots, \widetilde{\mathbf{A}}_i\mathbf{B}_n$, this is equivalent to i-th edge device receiving the coding sequence Θ $(\Theta \in \mathbb{F}_q^{n\times 1})$ and the elements in Θ are all 1. When any L devices collude, the coding sequence that edge devices can obtain is still Θ. At this time, according to [17–23] the privacy condition is equivalent to

$$I(d; \Theta, \widetilde{\mathbf{A}}_{s_L}, \mathbf{B}) = 0. \tag{4}$$

Therefore, we need to prove Eq. (4). According to the chain rule, the privacy condition is given as follows:

$$\begin{aligned} I(d; \Theta, \widetilde{\mathbf{A}}_{s_L}, \mathbf{B}) &= I(d; \Theta) + I(d; \widetilde{\mathbf{A}}_{s_L}, \mathbf{B}|\Theta) \\ &= I(d; \Theta) + I(d; \widetilde{\mathbf{A}}_{s_L}|\Theta) + I(d; \mathbf{B}|\widetilde{\mathbf{A}}_{s_L}, \Theta). \end{aligned} \tag{5}$$

Firstly, since all elements of Θ are the same, the uncertainty of d cannot be reduced when any L edge devices conspire to obtain Θ. Hence, $I(d; \Theta) = 0$. Secondly, since each element in $\widetilde{\mathbf{A}}_{s_L}$ is generated according to Eq. (1) and Eq. (3). In SPCC, Eq. (1) is a deterministic function and d is independent of Θ, we can get $I(d; \widetilde{\mathbf{A}}_{s_L}|\Theta) = 0$. Thirdly, since the user does not know the content of the data $\mathbf{B}$ and d is independent of $\mathbf{B}$, we can get $I(d; \mathbf{B}|\widetilde{\mathbf{A}}_{s_L}, \Theta) = 0$. To sum up the above, we can get $I(d; \Theta, \widetilde{\mathbf{A}}_{s_L}, \mathbf{B}) = 0$, *i.e.* the privacy condition is satisfied.

Theorem 3. *The SPCC satisfies the decodability condition iff* $T \geq m + L$.

Proof. Since the polynomial of Eq. (1) in SPCC has degree $m + L - 1$, we can decode the final results when $m + L$ edge devices return intermediate results. Therefore, the recovery threshold of SPCC is $N_{SPCC} = m + L$. On one hand, if $T \geq m + L$, we can get $T \geq N_{SPCC}$. According to Eq. (1) and Eq. (3), we can know that the exponent of $\mathbf{A}_i\mathbf{B}_d$ in each intermediate result is unique $\forall i \in \{1, \cdots, m\}$. Therefore, the decodability condition is satisfied. On the other hand, if SPCC satisfies the decodability condition, the number of blocks generated after encoding $\mathbf{A}$ is greater than N_{SPCC}, *i.e.* $T \geq N_{SPCC} = m + L$. To sum up, the SPCC satisfies the decodability condition iff $T \geq m + L$.

Next, we give the analysis of communication cost and computational load of SPCC.

For communication cost of SPCC, firstly, the user sends T coded blocks of $\mathbf{A}$ to the edge devices in total and the dimension of each block is $\frac{t}{m} \times p$. The communication cost of sending is $\frac{tpT}{m}$. Secondly, in order to protect the privacy of data $\mathbf{B}$, each edge device returns n calculation results, *i.e.* $\widetilde{\mathbf{A}}_i\mathbf{B}_1, \cdots, \widetilde{\mathbf{A}}_i\mathbf{B}_n$, and the dimension of them is $\frac{t}{m} \times 1$. In addition, when edge devices return $m + L$ intermediate results, we can decode the final results. The communication cost of intermediate results returned by edge devices is $\frac{n(m+L)t}{m}$. Therefore, the communication cost of SPCC is $(Tp + n(m + L))\frac{t}{m}$.

For computational load of SPCC, in order to protect the privacy of data $\mathbf{B}$, each edge device needs to perform n matrix-vector multiplication. The size of the matrix is $\frac{t}{m} \times p$ and the size of the vector is $p \times 1$. Therefore, the computational load is $\frac{npt}{m}$.

4.2 Secure and Private Coding Scheme with Low Computational Load (SPCL)

However, in SPCC, since edge devices need to perform n matrix-vector multiplication, SPCC obtains a high computational load. Therefore, in this section, we propose the SPCL. Similarly, we prove the security, privacy and decodability conditions of SPCL. Finally, we analyze the communication cost and computational load of SPCL.

In SPCL, to satisfy the privacy condition, the user needs to send the coding sequences $\mathbf{Q}$ to edge devices [17–23]. Firstly, the user lets θ be the d-th column of $n \times n$ identity matrix and generates L random vectors $\mathbf{Z} = [\mathbf{Z}_1, \cdots, \mathbf{Z}_L]$, where $\mathbf{Z}_i \in \mathbb{F}_q^{n \times 1}$, $i \in \{1, \cdots, L\}$. Then, the user selects T non-zero value $\{x_1, \cdots, x_T\}$ to get $\mathbf{Q} = [\mathbf{Q}_1, \cdots, \mathbf{Q}_T]$, where $\mathbf{Q}_i = \mathbf{Q}_{(x_i)}$ and the general expression $\mathbf{Q}_{(x)}$ of $\mathbf{Q}_{(x_i)}$ is shown in Eq. (6).

For the given m, L, we redesign the $\mathbf{Q}_{(x)}$ based on [23] and give α_i are shown as follows:

$$\mathbf{Q}_{(x)} = \frac{\theta}{x} + \mathbf{Z}_1 + x\mathbf{Z}_2 + \cdots + x^{L-1}\mathbf{Z}_L, \tag{6}$$

Table 1. The exponents of x in $f(x)g(x)$ when using SPCL.

	$\theta : -1$	$\mathbf{Z}_1 : 0$	$\cdots$	$\mathbf{Z}_L : L-1$
$\mathbf{A}_1 : 0$	-1	0	$\ldots$	$L-1$
$\mathbf{A}_2 : L+1$	L	$L+1$	$\cdots$	$2L$
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
$\mathbf{A}_m : (m-1)(L+1)$	$(m-1)(L+1)-1$	$(m-1)(L+1)$	$\cdots$	$(m-1)(L+1)+L-1$
$\mathbf{R}_1 : mL-L+m$	$(m-1)(L+1)$	$(m-1)(L+1)+1$	$\cdots$	$(m-1)(L+1)+L$
$\mathbf{R}_2 : mL-L+m+1$	$(m-1)(L+1)+1$	$(m-1)(L+1)+2$	$\cdots$	$(m-1)(L+1)+L+1$
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
$\mathbf{R}_L : mL+m-1$	$(m-1)(L+1)+L-1$	$(m-1)(L+1)+L$	$\cdots$	$(m-1)(L+1)+2L-1$

$$\alpha_i = \begin{cases} (i-1)(L+1), & \forall 1 \le i \le m; \\ mL-L+i-m, & \forall m+1 \le i \le m+L. \end{cases} \tag{7}$$

In this scheme, in order to prevent edge devices from getting relevant information of d in the case of collusion of any L edge devices, we add L random vectors to each coding sequence of $\mathbf{Q}$ in the process of coding. Therefore, one coding sequence is composed of $L+1$ vectors and only θ is valid. As shown in Table 1, in order to guarantee the decodability condition of the scheme, the values from rows $\mathbf{A}_1$ to $\mathbf{A}_m$ of the θ column are unique. The exponential interval of the first m term in $f(x)$ is $m+L$. In addition, for the rows from $\mathbf{R}_1$ to $\mathbf{R}_L$, we need to overlap these items as much as possible. Therefore, the value of α_i $(m+1 \le i \le m+L)$ is different from α_j $(1 \le j \le m)$. By setting Eq. (6) and Eq. (7), we make the interference terms overlap as much as possible and decrease the degree of the highest term of the polynomial.

The computation process are as follows:

- Firstly, the user encodes data $\mathbf{A}$ according to Eq. (1) and Eq. (7) to obtain $\widetilde{\mathbf{A}} = \left[\widetilde{\mathbf{A}}_1^\top, \widetilde{\mathbf{A}}_2^\top, \cdots, \widetilde{\mathbf{A}}_T^\top\right]^\top$. Besides, the user also generates $\mathbf{Q} = [\mathbf{Q}_1, \cdots, \mathbf{Q}_T]$, where $\mathbf{Q}_i = \mathbf{Q}_{(x_i)}$ according to Eq. (6). Then, $\widetilde{\mathbf{A}}_i^\top$ and $\mathbf{Q}_i$ are sent to the i-th edge device.
- Then, when the edge device receives the coded block $\widetilde{\mathbf{A}}_i$ and coding sequence $\mathbf{Q}_i$, the i-th edge device firstly performs the calculation $g(x_i)$ where the general expression $g(x)$ of $g(x_i)$ is shown as below:

$$g(x) = \mathbf{B}\mathbf{Q}_{(x)}. \tag{8}$$

  After i-th edge devices obtaining $g(x_i)$, the edge device performs the computation $f(x_i)g(x_i)$ and returns the intermediate result.
- Finally, once the user receives N_{SPCL} intermediate results (N_{SPCL} is the recovery threshold of SPCL and $N_{SPCL} \le T$), the user can obtain the final result $\mathbf{A}\mathbf{B}_d$ by using polynomial interpolation [10]. The general expression of the intermediate result $f(x_i)g(x_i)$ can be shown in Eq. (9).

$$
\begin{aligned}
f(x)g(x) &= (\sum_{i=1}^{m} \mathbf{A}_i x^{\alpha_i} + \sum_{j=1}^{L} \mathbf{R}_j x^{\alpha_{m+j}}) \mathbf{B}\mathbf{Q}_{(x)} \\
&= \sum_{i=1}^{m} \mathbf{A}_i x^{\alpha_i} \mathbf{B}\mathbf{Q}_{(x)} + \sum_{j=1}^{L} \mathbf{R}_j x^{\alpha_{m+j}} \mathbf{B}\mathbf{Q}_{(x)}.
\end{aligned} \tag{9}
$$

Similar to the proof of Theorem 1, we can get the SPCL meets the security condition.

Theorem 4. *The SPCL is private in the case of collusion of any L edge devices, i.e. the privacy condition is satisfied.*

Proof. We let $\mathbf{Q}_{s_L}$ be the set of coding sequences on any L edge devices. When we use SPCL, according to [17–23] the privacy condition is equivalent to

$$
I(d; \mathbf{Q}_{s_L}, \widetilde{\mathbf{A}}_{s_L}, \mathbf{B}) = 0. \tag{10}
$$

Therefore, we need to prove Eq. (10). According to the chain rule, the privacy condition is given as follows:

$$
\begin{aligned}
I(d; \mathbf{Q}_{s_L}, \widetilde{\mathbf{A}}_{s_L}, \mathbf{B}) &= I(d; \mathbf{Q}_{s_L}) + I(d; \widetilde{\mathbf{A}}_{s_L}, \mathbf{B} | \mathbf{Q}_{s_L}) \\
&= I(d; \mathbf{Q}_{s_L}) + I(d; \widetilde{\mathbf{A}}_{s_L} | \mathbf{Q}_{s_L}) + I(d; \mathbf{B} | \widetilde{\mathbf{A}}_{s_L}, \mathbf{Q}_{s_L}).
\end{aligned} \tag{11}
$$

Firstly, since each element in $\mathbf{Q}_{s_L}$ is by encoding θ with L random vector according to Eq. (6) by the user, any L colluding edge devices can not obtain relevant information of θ, *i.e.* $I(\theta; \mathbf{Q}_{s_L}) = 0$. In addition, since θ contains the information of d, we can know that any L edge devices cannot acquire information of d, *i.e.* $I(d; \mathbf{Q}_{s_L}) = 0$. Secondly, since each element in $\widetilde{\mathbf{A}}_{s_L}$ is generated according to Eq. (1) and Eq. (7) and both are certain function, the d is independent of $\widetilde{\mathbf{A}}_{s_L}$. Hence, $I(d; \widetilde{\mathbf{A}}_{s_L} | \mathbf{Q}_{s_L}) = 0$. Thirdly, similar to the proof of Theorem 2, we can get $I(d; \mathbf{B} | \widetilde{\mathbf{A}}_{s_L}, \mathbf{Q}_{s_L}) = 0$. To sum up the above, we can get $I(d; \mathbf{Q}_{s_L}, \widetilde{\mathbf{A}}_{s_L}, \mathbf{B}) = 0$, *i.e.* the privacy condition is satisfied.

Theorem 5. *The SPCL satisfies the decodability condition iff* $T \geq (m-1)(L+1) + 2L + 1$.

Proof. According to Eq. (6) and Eq. (7), we can obtain the exponents of x in $f(x)g(x)$ in Table 1. As shown in Table 1, we can get the recovery threshold of SPCL is $N_{SPCL} = (m-1)(L+1) + 2L + 1$. On one hand, if $T \geq (m-1)(L+1) + 2L + 1$, we can get $T \geq N_{SPCL}$. In addition, by designing $\mathbf{Q}_{(x)}$ and α_i, we can make that the values from rows $\mathbf{A}_1$ to $\mathbf{A}_m$ of the θ column in Table 1 are unique. Further, we can get the exponents of $\mathbf{A}_i \mathbf{B}_d$ in each intermediate result is unique $\forall i \in \{1, \cdots, m\}$. Therefore, the decodability condition is satisfied. On the other hand, if SPCC satisfies the decodability condition, the number of blocks generated after encoding $\mathbf{A}$ is greater than N_{SPCL}, *i.e.* $T \geq N_{SPCL} = (m-1)(L+1) + 2L + 1$. To sum up, the SPCL satisfies the decodability condition iff $T \geq (m-1)(L+1) + 2L + 1$.

Next, we give the analysis of communication cost and computational load of SPCL.

For communication cost of SPCL, firstly, the user sends T coded blocks of $\mathbf{A}$ and coding sequence $\mathbf{Q}_x$ to edge devices in total. The dimension of each coded blocks of $\mathbf{A}$ is $\frac{t}{m} \times p$ and the dimension of each coding sequence $\mathbf{Q}_x$ is $n \times 1$. Therefore, the communication of data sent by the user to the edge devices is $nT + \frac{tpT}{m}$. Moreover, the dimension of an intermediate result is $\frac{t}{m} \times 1$, and the recovery threshold of SPCL is $(m-1)(L+1) + 2L + 1$. Therefore, the communication cost of intermediate results sent by edge devices to the user is $((m-1)(L+1) + 2L + 1)\frac{t}{m}$. Therefore, the communication cost of the system SPCL is $T(\frac{tp}{m} + n) + ((m-1)(L+1) + 2L + 1)\frac{t}{m}$.

For computational load of SPCL, firstly, in order to protect the privacy of data $\mathbf{B}$, the user needs to send one coding sequence to each edge device. Then, the i-th edge device performs one matrix-vector multiplication. The size of the matrix is $p \times n$ and the size of the vector is $n \times 1$. After that, each device can obtain a vector and the dimension of it is $p \times 1$. The computational load of this process is np. Then, the i-th edge device perform $f(x_i)g(x_i)$. The computational load of the process is $\frac{tp}{m}$. Therefore, the computational load of SPCL is $\frac{tp}{m} + np$.

4.3 Example of SPCC and SPCL

In this section, we give two examples of the two schemes proposed in this paper. In the following example, we assume $t = 6$, $m = 3$, $p = 2$, $n = 3$ $L = 2$. The goal of the user is computing $\mathbf{AB}_1$ satisfying security, privacy and decodability conditions. For clarity, we assume $T = N$. The detailed example of SPCC can be shown as follows:

Example of SPCC

- **Coding** Firstly, the user divides $\mathbf{A}$ by rows to obtain $\mathbf{A} = \left[\mathbf{A}_1^\top, \mathbf{A}_2^\top, \mathbf{A}_3^\top\right]^\top$. Then, the cloud sends $\mathbf{B} = [\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3]$ to edge devices before computing. Next, the cloud will not communicate with edge devices. The user randomly generates 2 blocks $\mathbf{R}_1, \mathbf{R}_2$ where $\mathbf{R}_1, \mathbf{R}_2 \in \mathbb{F}_q^{2\times p}$. Moreover, the user selects 5 values $x_1, \cdots, x_5$ for encoding $\mathbf{A}$ according to Eq. (12) to obtain 5 coded blocks $\widetilde{\mathbf{A}}_1, \cdots, \widetilde{\mathbf{A}}_5$ where $\widetilde{\mathbf{A}}_i = f(x_i)$, $\forall i \in \{1, \cdots, 5\}$. Finally, the user sends $\widetilde{\mathbf{A}}_1, \cdots, \widetilde{\mathbf{A}}_5$ to 5 edge devices respectively.

$$f(x) = \mathbf{A}_1 x^0 + \mathbf{A}_2 x^1 + \mathbf{A}_3 x^2 + \mathbf{R}_1 x^3 + \mathbf{R}_2 x^4. \tag{12}$$

- **Computing** After the i-th edge device receives $\widetilde{\mathbf{A}}_i$, edge devices perform calculation $\widetilde{\mathbf{A}}_i\mathbf{B}_1, \cdots, \widetilde{\mathbf{A}}_i\mathbf{B}_n$. Finally, the i-th edge device return $\widetilde{\mathbf{A}}_i\mathbf{B}_1, \cdots, \widetilde{\mathbf{A}}_i\mathbf{B}_n$ to the user.
- **Decoding** The user receives $\widetilde{\mathbf{A}}_i\mathbf{B}_1, \cdots, \widetilde{\mathbf{A}}_i\mathbf{B}_n, \forall i \in \{1, \cdots, 5\}$. The user only uses $\widetilde{\mathbf{A}}_i\mathbf{B}_1$, $\forall i \in \{1, \cdots, 5\}$, where $\widetilde{\mathbf{A}}_i\mathbf{B}_1$ can be shown as follows:

$$\begin{aligned}\widetilde{\mathbf{A}}_i\mathbf{B}_1 = \mathbf{A}_1\mathbf{B}_1 + \mathbf{A}_2\mathbf{B}_1 x_i + \mathbf{A}_3\mathbf{B}_1 x_i^2 \\ + \mathbf{R}_1\mathbf{B}_1 x_i^3 + \mathbf{R}_2\mathbf{B}_1 x_i^4.\end{aligned}$$

The recovery threshold, communication cost, and computational load are 5, 25, and 12 respectively.

Example of SPCL

- **Coding** In the coding stage, the user select 11 values $\{x_1, \cdots, x_{11}\}$ to encode $\mathbf{A}$ and generate coding sequence $\mathbf{Q}$ according to Eq. (13) where $\theta = [1\ 0\ 0]^\top$.

$$\begin{aligned} f(x) &= \mathbf{A}_1 x^0 + \mathbf{A}_2 x^3 + \mathbf{A}_3 x^6 + \mathbf{R}_1 x^9 + \mathbf{R}_2 x^{12}, \\ \mathbf{Q}_{(x)} &= \frac{\theta}{x} + \mathbf{Z}_1 + x\mathbf{Z}_2. \end{aligned} \tag{13}$$

- **Computing** After coding phase, the user sends coded block $\widetilde{\mathbf{A}}_i$ and coding sequence $\mathbf{Q}_{(x_i)}$ to the i-th edge devices for computation. When the i-th edge device receives the coding sequence $\mathbf{Q}_{(x_i)}$, they encode $\mathbf{B}$ according to $g(x)$ in Eq. (14). Finally, the i-th edge device computes $f(x_i)g(x_i)$ and returns the result.

$$g(x) = [\mathbf{B}_1\ \mathbf{B}_2\ \mathbf{B}_3]\mathbf{Q}_{(x)}. \tag{14}$$

- **Decoding** Once the user receives 11 intermediate results, the user can decode the result $\mathbf{A}\mathbf{B}_d$. The intermediate result $f(x)g(x)$ can be shown in Eq. (15). In Eq. (15), J_i is the interference term, $\forall i \in \{1, \cdots, 8\}$.

$$\begin{aligned} f(x)g(x) = {} & \mathbf{A}_1\mathbf{B}_1 x^{-1} + \mathbf{A}_2\mathbf{B}_1 x^2 + \mathbf{A}_3\mathbf{B}_1 x^5 \\ & + J_1 x^0 + J_2 x^1 + J_3 x^3 + J_4 x^4 + J_5 x^6 + J_6 x^7 + J_7 x^8 + J_8 x^9. \end{aligned} \tag{15}$$

The recovery threshold, communication cost, and computational load are 11, 66, and 10 respectively.

5 Experiments

In this section, we compare the proposed schemes SPCC and SPCL with the existing schemes SPC [21] and XSTP [22] in the case that one round calculation can obtain the final result. We conduct extensive experiments to demonstrate that both the two proposed coding schemes can achieve lower communication cost and computational load than existing work.

5.1 Parameter Settings

To ensure that each scheme can be decoded in the simulation experiment, we do not limit the number of edge devices but we must ensure $T \geq N$ for each scheme. Specifically, we consider the changes of the following six parameters to compare the performances of the two baseline schemes: (1) t, the number of rows in $\mathbf{A}$, (2) m, the number of blocks divided by $\mathbf{A}$, (3) p, the number of columns in $\mathbf{A}$, (4) n, the number of columns in $\mathbf{B}$, (5) L, the number of collusive edge devices (6) $\beta = \frac{N}{T}$, the percentage of edge devices that return the intermediate results on time. Besides, we set $t = 2000$, $m = 200$, $p = 100$, $n = 20$, $L = 15$ and $\beta = 0.8$ as the default parameters. For the convenience of observation, we will use the natural logarithm of the y-axis in the experiment.

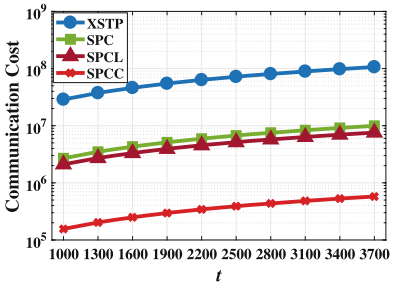
5.2 Experimental Results of Communication Cost

As shown in Fig. 3(a), the communication cost of SPCC, SPCL, SPC, and XSTP rise as t increases. This is because the row numbers in $\mathbf{A}$ increase but the block numbers divided by $\mathbf{A}$ remain the same. This means an increase in the row numbers in each coded block of $\mathbf{A}$. In addition, since the threshold does not change with the change of t, the user's communication cost to the edge device and the edge device's communication cost to the user both rise. Therefore, the communication cost of all schemes rise. In all cases, the proposed SPCC can achieve the least amount of communication cost.

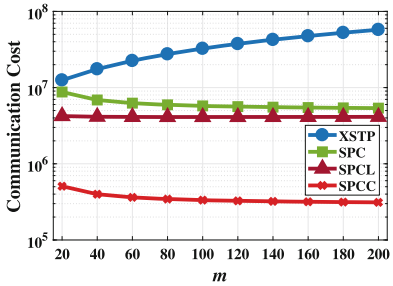
In Fig. 3(b), as m increases, the communication cost of SPCL remains essentially unchanged. However, the communication cost of SPCC and SPC decrease, and the communication cost of XSTP increases. This is because the number of blocks divided by $\mathbf{A}$ increases, and the row numbers of $\mathbf{A}$ does not change. This means the row numbers of coded blocks in $\mathbf{A}$ decreases. Besides, the recovery thresholds of SPCC, SPCL, SPC, and XSTP all increase. However, in SPCL, the communication cost gained from the decrease in the row numbers per coded block is generated by increasing m. This compensates for the communication cost created by the increased recovery threshold. For SPCC and SPC, when m is set a small value, the row numbers of the coded blocks are large, which means the row numbers of intermediate results are large. With the increase of m, the communication cost decreases, which means that the influence of threshold on communication cost is less than that of block size. For XSTP, although the row numbers of coded blocks of $\mathbf{A}$ decreases, the user needs to send m coded blocks of $\mathbf{A}$ to each device, and the threshold increases, the communication cost of XSTP increases. In all cases, the proposed SPCC can achieve the least amount of communication cost.

In Fig. 3(c), with the increase of p, the communication cost of SPCC, SPCL, SPC, and XSTP all increase. Besides, the communication cost between SPCC and SPC is significantly different. This is due to the fact that the column numbers of per coded block in $\mathbf{A}$ increase as p increases. This means that the number of elements in each block of $\mathbf{A}$ increases. In addition, since the threshold does not change with the change of p, the user's communication cost to the edge device and the edge device's communication cost to the user increase. Therefore, the communication cost of all schemes rise. In all cases, the proposed SPCC can achieve the least amount of communication cost.

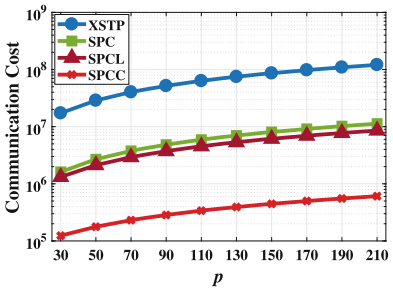
In Fig. 3(d), with the increase of n, the communication cost of SPCC and SPC increase. However, SPCL and XSTP are basically unchanged. Besides, when $n < 22$, the communication cost of SPCL is greater than SPC. With the increase of n, the communication cost of SPCL is lower than SPC. This is because each edge device needs to return n results in SPCC. For SPC, the user needs to send n coded blocks to each edge device. As n increases, the communication cost will increase significantly. For SPCL and XSTP, since the thresholds of the two schemes do not change, and the intermediate results and coding sequences account for only a small part of the communication cost of the two schemes, so the communication cost will not change significantly. When n is set to a small



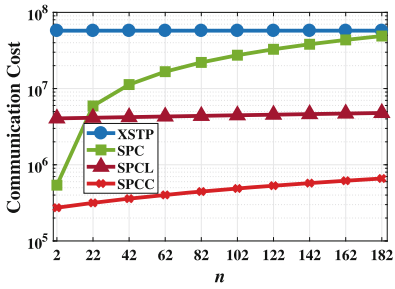
(a) the number of rows in **A**



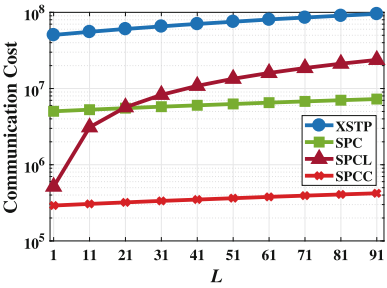
(b) the number of blocks divided by **A**



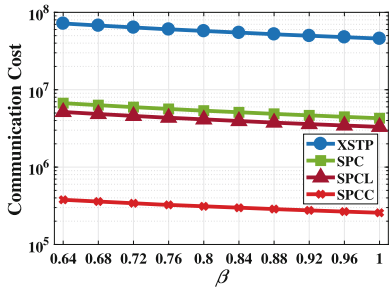
(c) the number of columns in **A**



(d) the number of columns in **B**



(e) the number of collusive edge devices



(f) the percentage of edge devices that return the intermediate results on time

Fig. 3. The communication cost when changing different parameters: t, m, p, n, L and β.

value, in SPC, the block numbers sent by the user to each edge device are small. However, at this time, due to the large recovery threshold of SPCL scheme, the communication cost brought by coding sequences and coded blocks is large. When n is set to a large value, since each edge device needs to send n coded blocks to each edge device in SPC, the communication cost of SPC is larger than SPCL. In all cases, the proposed SPCC can achieve the least amount of communication cost.

In Fig. 3(e), with the increase of L, the communication cost of SPCL, SPCC, SPC, and XSTP increase. This is due to the fact that the recovery thresholds

of all schemes increase. Consequently, the number of blocks that the user sends to edge devices and the number of intermediate results increase. This means the communication cost of all schemes increase. Furthermore, the communication cost of SPCL changes the greatest. because the increase of L leads to the increase of the threshold of SPCL, which is much larger than that of other schemes. When L is set a small value, the communication cost of SPCL is lower than SPC. However, the communication cost of SPCL is larger than SPC when $L > 21$ because the SPCL recovery threshold is small when L is small and the user sends only one coded blocks to each edge device. Therefore, the communication cost of SPCL is lower than SPC. However, when L is set a large value, the SPC recovery threshold does not change considerably. In SPCL, the recovery threshold becomes large. Therefore, the communication cost of SPCL is larger than SPC when L is large. In all cases, the proposed SPCC can achieve the least amount of communication cost.

In Fig. 3(f), with the increase of β, the communication cost of SPCC, SPCL, SPC, and XSTP all decrease. This is because the proportion of stragglers in edge devices decreases with the increase of β. Furthermore, the percentage of edge devices allocated to tasks and returning results on time rises, while the cost of incorrect communication falls. Therefore, the communication cost of SPCC, SPCL, SPC, and XSTP decrease. In all cases, the proposed SPCC can achieve the least amount of communication cost.

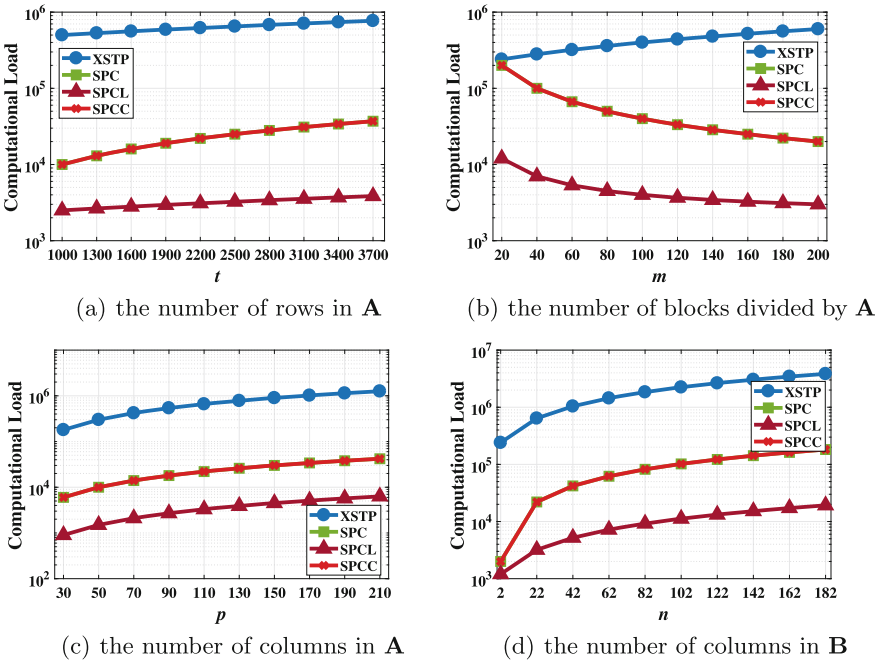
5.3 Experimental Results of Computational Load

Since the computational load only change with t, m, p, and n, we consider the changes of the four parameters to compare the performances of the two baseline schemes.

In all the following experiments, in SPCC, since each edge device needs to perform n matrix-vector multiplication, the size of the matrix is $\frac{t}{m} \times p$ and the size of the vector is $p \times 1$. Similarly, in SPC, each edge device needs to perform n matrix-vector multiplication of size $\frac{t}{m} \times p$ and $p \times 1$. Therefore, the computational load of them is the same.

As shown in Fig. 4(a), with the increase of t, the computational load of SPCC, SPCL, SPC, and XSTP all increase. This is because the row numbers in $\mathbf{A}$ increase. It means an increase in the row numbers in each coded block of $\mathbf{A}$ and each edge device needs to more row operations. However, for SPCL and XSTP, the change in computational load is not obvious. For SPCL, this is because although the rows of each coded block become large, the computational load of $g(x)$ is larger than $f(x)g(x)$. For XSTP, this is because the computational load of coding sequence accounts for a large part of the computational load. In all cases, the proposed SPCL can achieve the least amount of computational load.

As shown in Fig. 4(b), with the increase of m, the computational load of SPCC, SPCL, and SPC all decrease. However, the computational load of XSTP increase. Besides, when m is set a enough large value, the computational load of SPCL is close to SPC and SPCC. This is because the block numbers divided by $\mathbf{A}$ increase and the row numbers of $\mathbf{A}$ do not change. This means the row numbers



(a) the number of rows in **A** (b) the number of blocks divided by **A**

(c) the number of columns in **A** (d) the number of columns in **B**

Fig. 4. The computational load when changing different parameters: t, m, p and n.

in coded blocks of **A** decrease. In SPCC and SPC, the computational load of them decrease considerably because of performing n matrix-vector multiplication of size $\frac{t}{m} \times p$ and $p \times 1$. For SPCL, only the computational load of $f(x)g(x)$ decreases, and the computational load of $g(x)$ remains unchanged. For XSTP, each device needs to calculate the sum of m coded blocks multiplied by the coding sequence. In all cases, the proposed SPCL can achieve the least amount of computational load.

In Fig. 4(c), with the increase of p, the computational load of SPCC, SPCL, SPC, and XSTP all increase. This is due to the fact that the column numbers in per coded block of **A** increases as p increases. This means that the number of elements in each block of **A** increases. In SPCC and SPC, the computational load of them decrease considerably because of performing n matrix-vector multiplication of size $\frac{t}{m} \times p$ and $p \times 1$. For SPCL and XSTP, except that the coded block of **A** leads to an increase in the amount of computational load, the coding sequence also leads to an increase in the amount of computational load. In all cases, the proposed SPCL can achieve the least amount of computational load.

In Fig. 4(d), with the increase of n, the computational load of SPCC, SPCL, SPC, and XSTP all increase. This is because each edge device performs n matrix-vector multiplication in SPCC and SPC. In addition, for SPCL and XSTP, the size of coding sequences becomes large. Therefore, the computational load of all

schemes increase. In all cases, the proposed SPCL can achieve the least amount of computational load.

5.4 Intuitive Understanding

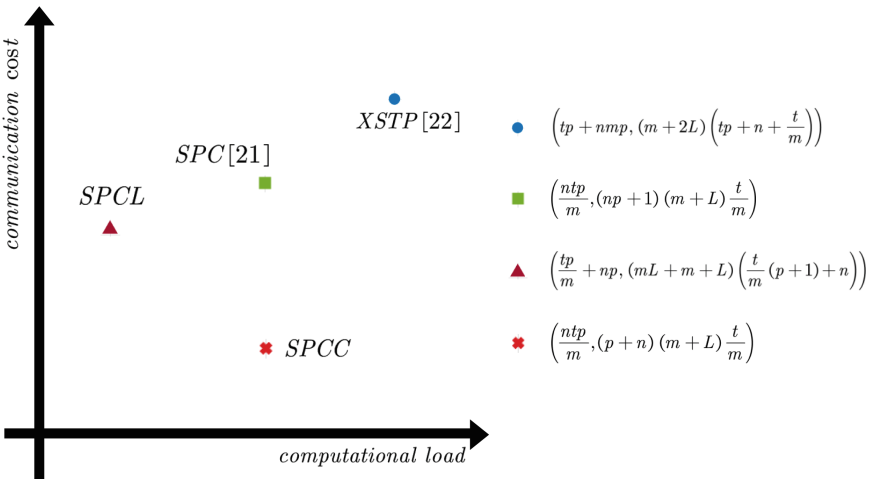


Fig. 5. Relationship between computational load volume and communication cost in all schemes

Finally, in order to give an intuitive understanding, Fig. 5 is given which is related with computational load and communication cost. For the convenience of comparison, we do not consider the straggler problem, *i.e.* $T = N$ in all schemes. The advantages of the two schemes we proposed are shown in the Fig. 5.

6 Conclusion

In this paper, we study the SPMM issue for edge computing against cooperative attack and design a general coded computation scheme to achieve lowest system resource consumptions, *i.e.* communication cost and computational load. Specially, we propose two coding schemes, *i.e.* SPCC and SPCL, to protect the security of data and the privacy of the user. In addition, we give the communication cost and computational load under the two schemes and discover that it is lower than existing work. Specifically, SPCC has more advantages in communication cost, while SPCL has more advantages in computational load. Therefore, they can be flexibly chosen for different scenarios. However, in this paper, we do not consider multi-round calculation and more complex attack models such as active attack. In our future work, we will consider designing the coding scheme to adapt to more complex attack models.

References

1. Asim, M., Wang, Y., Wang, K., Huang, P.Q.: A review on computational intelligence techniques in cloud and edge computing. IEEE Trans. Emerg. Top. Comput. Intell. **4**(6), 742–763 (2020)
2. Wang, L., Jiao, L., He, T., Li, J., Mühlhäuser, M.: Service entity placement for social virtual reality applications in edge computing. In: Proceedings of INFOCOM Conference on Computer Communications, pp. 468–476 (2018)
3. Lv, Z., Chen, D., Lou, R., Wang, Q.: Intelligent edge computing based on machine learning for smart city. Futur. Gener. Comput. Syst. **115**, 90–99 (2021)
4. Liu, S., Liu, L., Tang, J., Yu, B., Wang, Y., Shi, W.: Edge computing for autonomous driving: opportunities and challenges. Proc. IEEE **107**(8), 1697–1716 (2019)
5. Zhou, Z., Chen, X., Li, E., Zeng, L., Luo, K., Zhang, J.: Edge intelligence: paving the last mile of artificial intelligence with edge computing. Proc. IEEE **107**, 1738–1762 (2019)
6. Li, S., Maddah-Ali, A.M., Avestimehr, S.A.: Coding for distributed fog computing. IEEE Commun. Mag. **55**(4), 34–40 (2017)
7. Lee, K., Lam, M., Pedarsani, R., Papailiopoulos, D., Ramchandran, K.: Speeding up distributed machine learning using codes. IEEE Trans. Inf. Theory **64**(3), 1514–1529 (2017)
8. Soto, P., Li, J.: Straggler-free coding for concurrent matrix multiplications. In: Proceedings of International Symposium on Information Theory (ISIT), pp. 233–238 (2020)
9. Dutta, S., Fahim, M., Haddadpour, F., Jeong, H., Cadambe, R.V., Grover, P.: On the optimal recovery threshold of coded matrix multiplication. IEEE Trans. Inf. Theory **66**, 278–301 (2020)
10. Yang, H., Lee, J.: Secure distributed computing with straggling servers using polynomial codes. IEEE Trans. Inf. Forensics Secur. **14**, 141–150 (2019)
11. Bitar, R., Parag, P., Rouayheb, S.E.: Minimizing latency for secure distributed computing. In: Proceedings of International Symposium on Information Theory (ISIT), pp. 2900–2904 (2017)
12. Bitar, R., Parag, P., Rouayheb, S.E.: Minimizing latency for secure coded computing using secret sharing via staircase codes. IEEE Trans. Commun. **68**(8), 4609–4619 (2020)
13. Chang, W., Tandon, R.: On the capacity of secure distributed matrix multiplication. In: Proceedings of Global Communications Conference (GLOBECOM), pp. 1–6 (2019)
14. D'Oliveira, G.L.R., Rouayheb, E.S., Karpuk, D.: GASP codes for secure distributed matrix multiplication. IEEE Trans. Inf. Theory **66**(7), 4038–4050 (2020)
15. Wang, J., Cao, C., Wang, J., Lu, K., Jukan, A., Zhao, W.: Optimal task allocation and coding design for secure edge computing with heterogeneous edge devices. IEEE Trans. Cloud Comput. (2021)
16. Zhu, L., Wang, J., Shi, L., Zhou, J., Lu, K., Wang, J.: Secure coded matrix multiplication against cooperative attack in edge computing. In: Proceedings of International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 547–556 (2020)
17. Kim, M., Yang, H., Lee, J.: Private coded computation for machine learning. arXiv:1807.01170 (2018)

18. Kim, M., Lee, J.: Private secure coded computation. IEEE Commun. Lett. 1918–1921 (2019)
19. Chang, W.T., Tandon, R.: On the upload versus download cost for secure and private matrix multiplication. In: IEEE Information Theory Workshop (ITW), pp. 469–473 (2019)
20. Yu, Q., Avestimehr, S.A.: Coded computing for resilient, secure, and privacy-preserving distributed matrix multiplication. IEEE Trans. Commun. **69**, 59–72 (2021)
21. Vaidya, K., Rajan, S.B.: Distributed computation-privacy, straggler mitigation, and security against colluding workers. In: Proceedings of Global Communications Conference (GLOBECOM), pp. 1–6 (2020)
22. Jia, Z., Jafar, S.A.: X-secure T-private information retrieval from MDS coded storage with byzantine and unresponsive servers. IEEE Trans. Inf. Theory **66**, 7427–7438 (2020)
23. Kim, M., Yang, H., Lee, J.: Fully private coded matrix multiplication from colluding workers. IEEE Commun. Lett. **25**, 730–733 (2021)
24. Cai, N., Chan, T.: Theory of secure network coding. Proc. IEEE **99**(3), 421–437 (2011)