Securely Straggler-Exploiting Coded Computation for Distributed Matrix Multiplication

HEE CHEOL YANG, (Member, IEEE),
SANG WOO HONG, (Student Member, IEEE),
AND JUNG WOO LEE, (Senior Member, IEEE)

1Division of Computer Convergence, Chungnam National University, Daejeon 34134, South Korea
2Communications and Machine Learning Laboratory, Department of Electrical and Computer Engineering, Seoul National University, Seoul 08826, South Korea

Corresponding author: Heecheol Yang (hcyang@cnu.ac.kr)

This work was supported in part by the National Research Foundation (NRF) of Korea under Grant 2020R1G1A1003759 (30%) and Grant 2021R1A4A1030898 (35%), and in part by the Institute of Information & Communications Technology Planning & Evaluation (IITP, 2021-0-00106 (35%)) Grant funded by the Ministry of Science and ICT (MSIT), Institute of New Media and Communications (INMAC), and BK21-plus.

ABSTRACT
In this paper, we consider coded computation for matrix multiplication tasks in distributed computing, which can mitigate the effect of slow workers, called stragglers, using a coding approach. We assume that the straggler computation results can be leveraged at the master by assigning multiple sub-tasks to the workers. In this scenario, a new coded computation scheme is proposed to preserve the data security and privacy from workers, which is called securely straggler-exploiting codes (SSEC). Moreover, the proposed SSEC can efficiently reduce the communication load in distributed computing for assigning the sub-tasks to the workers, by overlapping the encoded matrices in assigning multiple sub-tasks with appropriate polynomial functions. It is also proven that the data security and privacy constraints are satisfied in SSEC in an information-theoretic sense. In conclusion, SSEC shows good performance on the recovery threshold and communication loads and compare them with the existing secure coded computation schemes for matrix multiplication tasks.

INDEX TERMS
Distributed computing, coded computation, matrix multiplication, polynomial codes.

I. INTRODUCTION
For many machine learning operations and big data analytics, matrix multiplication is a common computation in making inferences from a dataset. As the size of a dataset increases, there has been an attempt to process numerous calculations on a large-scale dataset using distributed computing with multiple processing nodes, called workers. However, it has been reported that among multiple workers there could exist straggling workers that return computation results too slowly or even fail [2]. The effect of straggling workers, i.e., straggler effects, could be critical to the performance of distributed computing systems because it can be detrimental to the completion of a calculation task.

This study considers coded computation in distributed computing where the straggler effect can be reduced by a coding approach. For instance, it has been shown that when \((M, K)\) maximum distance separable (MDS) codes are employed for matrix-vector multiplication tasks for distributed computing with \(M\) workers, the straggler effect can be significantly mitigated because an aggregation node, called a master, only requires the fastest \(K\) computation results to obtain the final outcome [3]. This approach has been extended to various computational tasks, especially for matrix multiplication tasks [4]–[11]. In [4], the authors have employed polynomial codes [12] to the matrix multiplication tasks in distributed computing to alleviate the straggler effect. It was proved that by polynomial codes, the recovery threshold at a master, i.e., the minimum number of computation returns that the master needs to wait to obtain the final product for the worst-case scenario, does not scale with the number of workers, which implies that the straggler effect does not worsen as the number of workers increases. This work has been generalized in [5] and [6] to various matrix partitioning settings. In particular, the authors in [5] have proposed Matdot and Polydot coding schemes to reduce the memory usage of workers, and provided a fundamental trade-off between the communication load in distributed computing and the...
recovery threshold at the master. The authors in [6] have also demonstrated this trade-off by introducing a general coding scheme from polynomial codes, called entangled polynomial codes. Various coding schemes have also been proposed for matrix multiplication tasks in distributed computing to alleviate the straggler effect [7]–[11].

Recently, a different approach has been proposed to mitigate the straggler effect in distributed computing by exploiting straggling workers’ computation results at the master, rather than ignoring them [13]–[17]. This concept can be realized by assigning many partitioned small sub-tasks to the workers and letting the workers return the computation results of the sub-tasks instantly to leverage the computational capability of the straggling workers. This concept is called the straggler-exploiting coded computation in this study. The authors in [13] have introduced a new task allocation scheme to assign multiple small sub-tasks to the workers for a straggler-exploiting scenario, instead of assigning a single big sub-task to each worker. In addition, the authors in [14] have proposed a new fine-grained task allocation scheme to efficiently utilize the straggling workers’ computation ability to speed up the entire task execution. Furthermore, recent papers [15]–[17] have suggested new encoding schemes for matrix multiplication to efficiently utilize the memory of the workers and to lower the communication load in distributed computing.

In a distributed computing framework, it is important to resolve security issues when a master wants to exploit suspicious but useful workers to boost the performance of distributed computing, thus security-sensitive information, such as personal information and medical records, in input data should be protected from the workers. In addition, there could exist an eavesdropper who overhears input data from the links between a master and workers. For instance, assume that a master wants to perform training on a security-sensitive dataset with large-scale deep neural networks. It can utilize multiple workers to deal with multiplications between huge matrices in neural networks, but may want to hide information on the input dataset and parameter updates of neural networks from workers and possible eavesdroppers. This study considers two types of security issues in distributed computing for matrix multiplication tasks. The first one is to protect information about input matrices from workers in an information-theoretic sense, which has also been studied in [18]–[27].

To avoid revealing information about input matrices from the allocated sub-tasks to workers, random matrices generated at a master are added to the encoded sub-matrices that will be transmitted to the workers. The second is to maintain data privacy from the workers in an information-theoretic sense, which has also been considered in [23]–[28]. In this setting, a master wants to perform matrix multiplication between its own matrix and another in a worker library consisting of multiple matrices. To preserve data privacy, the master does not want to reveal the matrix that it wants to perform matrix multiplication from a library at workers. For this purpose, symmetry between the matrices in a library needs to be guaranteed across the encoded sub-matrices to allocate the sub-tasks.

In this paper, we utilize a coded computation scheme in straggler-exploiting distributed computing for matrix multiplication tasks, while preserving data security and privacy from workers. We propose a novel encoding scheme for assigning sub-tasks that efficiently reduces the communication load from a master to the workers for assigning sub-tasks and ensures data security and privacy at the allocated sub-matrices to the workers as well, called securely straggler-exploiting codes (SSEC). Specifically, the main contributions of this study are twofold.

- We propose a coded computation scheme for a straggler-exploiting scenario that can preserve data privacy from workers for the first time to the best of our knowledge. This implies that data privacy can be maintained from multiple sub-tasks assigned to each worker in the scheme. To preserve data privacy from multiple sub-tasks, a special polynomial function is introduced that has several sets of evaluation points that have the same value in encoding functions. With the special polynomial function, a symmetry between the matrices in a library at the encoding functions can be guaranteed, thus the index of the desired matrix can be hidden from each worker.

- In the proposed SSEC, the communication load for allocating sub-tasks to workers can be significantly reduced. The key idea of SSEC is to overlap encoded sub-matrices for assigning multiple sub-tasks with appropriate polynomial functions, as in [16], [17].

For the proposed SSEC, the encoding conditions that the polynomial function in the encoding scheme need to satisfy are derived. Furthermore, it is proven that in SSEC, data security and privacy can be guaranteed in an information-theoretic sense in the allocated sub-matrices sent to workers. Finally, the performance of the SSEC on the recovery threshold and communication loads for task allocation and computation return between a master and workers is stated. An analysis of computational complexities in the entire distributed computing process of SSEC, such as encoding the sub-matrices for allocating sub-tasks, computing the sub-tasks at the workers, and decoding the final product at the master from the returned computation results of the sub-tasks is also complementally provided.

**A. RELATED WORK**

In the literature, several coded computation schemes have been proposed for matrix multiplication tasks considering security issues in distributed computing. Existing studies have focused on preserving data security [18]–[27], and some studies have assumed a straggler-exploiting scenario, but this was limited to matrix-vector multiplication tasks [21]. It should be noted that other schemes for preserving data security from colluding workers but not considering a straggler-exploiting scenario can be easily extended to this scenario by allocating multiple sub-tasks to each worker.
which were designed to be allocated to colluding workers. In addition, several studies have covered data privacy for matrix multiplication tasks [23]–[28]. However, they did not consider a straggler-exploiting scenario or protected data privacy from multiple colluding workers, with the exception of the authors in [28], who proposed a coded computation scheme that preserves data privacy for a straggler-exploiting scenario. However, their scheme is based on a grouping-based approach, thus decoding conditions should be satisfied for all groups of workers and therefore shows poor performance in terms of the recovery threshold considering the worst-case scenario. However, the proposed SSEC is not affected by the worst case, and a master requires a fixed number of computation results from workers.

### B. ORGANIZATION

In Section II, we describe the system model considered in this study, and formulate a distributed matrix multiplication problem to be addressed. In Section III and IV, we propose SSEC for private and secure distributed matrix multiplications, and provide the corresponding main results. In Section V, we analyze the computational complexity of the SSEC. Finally, we conclude this paper in Section VI.

### C. NOTATION

For $a, b \in \mathbb{Z}, [a : b]$ denotes $\{a, a+1, \ldots, b\}$, and $[a]$ denotes $\{a\}$. For $\mathcal{N} \subset \mathbb{N}, A_{\mathcal{N}}$ denotes $\{A_n\}_{n \in \mathcal{N}}$.

### II. SYSTEM MODEL & PROBLEM FORMULATION

We consider a straggler-exploiting distributed computing scenario for the matrix multiplication task. For instance, a master wants to perform a matrix multiplication $C = AB$ on large-scale input matrices $A \in \mathbb{F}^{N \times N}$ and $B \in \mathbb{F}^{N \times N}$, i.e., $N \gg 1$, for a sufficiently large finite field $\mathbb{F}$ with $W$ workers in parallel. Each worker $W_i, i \in [1 : W]$ is assigned to compute $L$ sub-tasks $\{C_{i,j}\}_{j=1}^L$ and each sub-task for $W_i$ is computed by multiplying two coded matrices $\tilde{C}_{i,j} = \tilde{A}_{i,j} \tilde{B}_{i,j}$, where $\tilde{A}_{i,j}$ and $\tilde{B}_{i,j}$ are encoded from $A$ and $B$ by the encoding functions at the master, respectively.

To allocate the sub-tasks to the workers, the master encodes the coded matrices of $A$ and $B$, and sends them to each worker. Specifically, the master encodes $\tilde{A}_{i,j}$ and $\tilde{B}_{i,j}$ for $j \in [1 : L]$ for $W_i$, and sends $\{\tilde{A}_{i,j}\}_{j=1}^L$ and $\{\tilde{B}_{i,j}\}_{j=1}^L$ to $W_i$. At each worker $W_i$, after receiving $\{\tilde{A}_{i,j}\}_{j=1}^L$ and $\{\tilde{B}_{i,j}\}_{j=1}^L$, it computes the allocated sub-tasks $\tilde{C}_{i,j} = \tilde{A}_{i,j} \tilde{B}_{i,j}$ for $j \in [1 : L]$, and sends the completed results to the master asynchroneously, which means that the workers send the computation results of the sub-tasks whenever each sub-task is completed. In addition, the computation order of the sub-tasks among $L$ sub-tasks for each worker is not restricted.

By using decoding functions, the master decodes the final product $C$ after a sufficient number of computation results are obtained. Specifically, it is assumed that the computation results $\tilde{C}_{i,j}$ for $j \in [1 : L]$ of the sub-task at $W_i$ are rearranged by the finishing time order of $\tilde{C}_{i,t}$ for $t \in [1 : L]$. If the master can decode $C$ from the received $T_i$ computation results from each $W_i$ at time $T$, the master stops receiving the computation results and decodes $C$ from $\{\tilde{C}_{i,t}\}_{i=1}^T$ for $\forall i \in [1 : W]$.

Using the above system model, the problem of straggler-exploiting distributed matrix multiplication is formulated by defining the following terminologies. We define encoding functions $\mathbf{p}_A$ and $\mathbf{p}_B$ to encode the sub-matrices $\tilde{A}_{i,j}$ and $\tilde{B}_{i,j}$ on the sub-task $\tilde{C}_{i,j}$ for $\forall i \in [1 : W]$ and $\forall j \in [1 : L]$.

In this study, we especially focus on linear encoding schemes that make linear encoding functions by using the partitions of matrices $A$ and $B$ for coefficients. In linear encoding schemes, a master divides the input matrices $A$ and $B$ into sub-matrices of equal size $A_{x,y} \in \mathbb{F}^{r \times q}$ for $q \in [1 : m]$ and $r \in [1 : p]$, and $B_{m,n} \in \mathbb{F}^{r \times n}$ for $w \in [1 : p]$ and $y \in [1 : n]$, which is given by

$$A = \begin{bmatrix} A_{1,1} & \cdots & A_{1,p} \\ \vdots & \ddots & \vdots \\ A_{m,1} & \cdots & A_{m,p} \end{bmatrix}, \quad B = \begin{bmatrix} B_{1,1} & \cdots & B_{1,n} \\ \vdots & \ddots & \vdots \\ B_{p,1} & \cdots & B_{p,n} \end{bmatrix}.$$ 

Linear encoding functions $\mathbf{p}_A$ and $\mathbf{p}_B$ are constructed using the divided sub-matrices as coefficients. Moreover, we introduce random matrices in the encoding functions to guarantee...
the security of the input matrices $A$ and $B$ from the workers. Thus, the encoding functions $p_A$ and $p_B$ can be written as

$$p_A(x) = \sum_{q,r} A_{q,r} q_r(x) + \sum_x Z_{A_i} c_i(x),$$

$$p_B(x) = \sum_{w,y} B_{w,y} b_{w,y}(x) + \sum_z Z_{B_i} d_i(z),$$

where $Z_{A_i} \in \mathbb{F}_p^{n \times \frac{m}{L}}$ and $Z_{B_i} \in \mathbb{F}_p^{n \times \frac{m}{L}}$ are the additional terms that will be defined later, $x$ represents the variable of the encoding functions $p_A$ and $p_B$, and $q_r(x), b_{w,y}(x), c_i(x), d_i(z)$ denote the polynomial functions on $x$. By using these encoding functions, $\tilde{A}_{i,j} \in \mathbb{F}_p^{\frac{m}{L} \times \frac{n}{L}}$ and $\tilde{B}_{i,j} \in \mathbb{F}_p^{\frac{m}{L} \times \frac{n}{L}}$ are encoded from $p_A(x)$ and $p_B(x)$ at evaluation points $x = x_{i,j}$, implying that $A_{i,j} = p_A(x_{i,j})$ and $B_{i,j} = p_B(x_{i,j})$. The sub-tasks assigned to $W_i$ are represented as $C_{i,j} = \tilde{A}_{i,j} \tilde{B}_{i,j} \in \mathbb{F}_p^{\frac{m}{L} \times \frac{n}{L}}$ for $j \in [1 : L]$. These results represent the values of the objective function $p_C(x)$, which is given by

$$p_C(x) = p_A(x) \times p_B(x),$$

at $x = x_{i,j}$. Given that $p_C$ is also a polynomial function as $p_A$ and $p_B$, the master aims to decode the final product $C = AB$ by interpolating the objective function $p_C(x)$ from the received results on evaluation points at $x = x_{i,j}$.

We formally define two private and secure distributed computing problems in a straggler-exploiting scenario for matrix multiplication tasks, which are also described in Fig. 1. In the considered scenarios, a master is assumed to be safe from adversaries and workers are assumed to be honest but curious, which means that workers always return correct computation results to a master but they are able to leak information to adversaries. The workers and possible adversaries are assumed to be computationally unbounded, thus the information-theoretic privacy and security constraints are considered.

i) Private and secure matrix multiplication (Scenario A): The master aims at ensuring data security among one of the input matrices, and at the same time keeping data privacy on the other input matrix from non-colluding workers. In private distributed computing, there is a library $B$ consisting of $K$ different matrices $(B_{i,k})_{k=1}^{K}$, which can be shared with the workers. The master would like to compute $AB(D)$, while hiding the identity of the desired matrix $B(D)$ from each worker. It can be said that the input matrix $B(D)$ is privately encoded from non-colluding workers if the set of $L$ elements of the allocated sub-tasks to any worker has zero information about the index $D$ among the matrices, i.e.,

$$I(\{\tilde{A}_{i,j}\}_{j=1}^{L}, \{\tilde{B}_{i,j}\}_{j=1}^{L}; C, D) = 0, \forall i \in [1 : W].$$

Additionally, the input matrix $A$ is securely encoded from non-colluding workers if the set of $L$ elements of the allocated sub-tasks to any worker has zero information about $A$, i.e.,

$$I(\{\tilde{A}_{i,j}\}_{j=1}^{L}; A, B(D)) = 0, \forall i \in [1 : W].$$

ii) Fully secure matrix multiplication (Scenario B): The master aims to ensure the security of both the input matrices from the colluding workers. We say that the input matrices $A$ and $B$ are securely encoded from $E$-colluding workers if the set of $LE$ elements of the allocated sub-tasks to any $E$ worker has zero information about $A$ and $B$, i.e.,

$$I(\{\tilde{A}_{i,j}\}_{j=1}^{L}; A, B(D)) = 0, \forall i \in [1 : W], \forall E \subset [1 : W], |E| = E.$$
• Recovery threshold $R$: The minimum number of computation results of the sub-tasks $\tilde{C}_{ij}$ that the master needs to wait to obtain the final product $C$ for the worst-case scenario, which means that the master can always obtain the final product if the number of received computation results of the sub-tasks is not smaller than recovery threshold $R$.

• Task-allocation communication load $L_t$: The number of elements of the coded matrices that the master sends to allocate the sub-tasks per worker. If there exist some replicated matrices among the coded matrices for a single worker, they are not counted again. It should be noted that this metric corresponds to the memory usage of each worker, as it can also represent the required memory size at each worker to store the coded matrices for the allocated sub-tasks from the master.

• Computation-return communication load $L_r$: The number of the elements of the computation results that the workers send to the master, as the master can decode the final product $C$ from the returned computation results.

The notations used in this paper are summarized in Table 1, and they are also defined throughout this paper.

### III. SECURELY STRAGGLER-EXPLOITING CODES FOR PRIVATE AND SECURE DISTRIBUTED MATRIX MULTIPLICATION

We dedicate this section to introduce SSEC for private and secure distributed matrix multiplication (Scenario A) in a straggler-exploiting scenario.

#### A. MOTIVATING EXAMPLE

We provide a motivating example for the SSEC, where the case of $m = 2$, $n = 2$, $K = 3$, $D = 1$ and where the master wants to compute $C = AB^{(1)}$ while hiding the computation $AB^{(1)}$ among $AB^{(1)}$, $AB^{(2)}$, and $AB^{(3)}$ from each worker is considered. The master wants to ensure the security of the input $A$ from each worker as well.

**Example 1 (Private and Secure Distributed Matrix Multiplication):** The master divides the input matrices $A$ and $B^{(k)}$ for $k \in [1 : 3]$ into half ($m = 2$, $n = 2$) to encode the sub-tasks with a $\frac{1}{2}$ size of the original task as follows.

\[
A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}, \quad B^{(k)} = \begin{bmatrix} B_1^{(k)} \\ B_2^{(k)} \end{bmatrix}, \quad \forall k \in [1 : 3].
\]

In this case, the final product $C = AB^{(1)}$ can be written as

\[
C = \begin{bmatrix} A_1B_1^{(1)} + A_2B_2^{(1)} \\ A_2B_1^{(1)} + A_1B_2^{(1)} \end{bmatrix}.
\]

Using the divided sub-matrices as coefficients, the encoding functions of SSEC are given by

\[
\text{p}_A(x) = A_1 + A_2x + R_{A_1}x^2 + R_{A_2}x^3,
\]

\[
\text{p}_B(x) = B_1^{(1)}g(x) + B_2^{(1)}g(x) + B_1^{(2)}g(x_{1,1}) + B_2^{(2)}g(x_{2,1}) + B_1^{(2)}g(x_{1,2}) + B_2^{(2)}g(x_{2,2}),
\]

where $x$ denotes the variables of the polynomials of $\text{p}_A$ and $\text{p}_B$. $R_{A_1}$ and $R_{A_2}$ are random matrices generated at the master, and $x_{1,1}^*$, $x_{1,2}^*$, $x_{2,1}^*$, and $x_{2,2}^*$ are constant values that will be explained later.

To ensure that the encoding function $\text{p}_B(x)$ has several sets of two points that have the same value on $g(x)$, we will use $g(x) = x^4$. It can also be achieved by $g(x) = x^2$, however, the decoding condition at the master was also considered, which will be explained later. By using the encoding functions $\text{p}_A(x)$ and $\text{p}_B(x)$, the encoded matrices $\tilde{A}_{ij}$ and $\tilde{B}_{ij}$ for $i \in [1 : W]$ and $j \in [1 : 2]$ can be determined by setting $x_{i,1} = -i$ and $x_{i,2} = i$ for $i \in [1 : W]$ of 2W points for each encoded matrix. We also determine $x_{1,1}^*$, $x_{1,2}^*$, $x_{2,1}^*$, and $x_{2,2}^*$ to satisfy $g(x_{1,1}^*) = g(x_{1,2}^*)$ and $g(x_{2,1}^*) = g(x_{2,2}^*)$ as follows.

\[
x_{1,1}^* = -W - 1, \quad x_{1,2}^* = W + 1,
\]

\[
x_{2,1}^* = -W - 2, \quad x_{2,2}^* = W + 2.
\]

Then, the encoded matrices $\tilde{A}_{ij}$ and $\tilde{B}_{ij}$ for $i \in [1 : W]$ and $j \in [1 : 2]$ can be determined as follows.

\[
\tilde{A}_{i,1} = \text{p}_A(-i) = A_1 - iA_2 + i^2R_{A_1} - i^3R_{A_2},
\]

\[
\tilde{A}_{i,2} = \text{p}_A(i) = A_1 + iA_2 + i^2R_{A_1} + i^3R_{A_2},
\]

\[
\tilde{B}_{i,1} = \tilde{B}_{i,2} = \text{p}_B(-i) = \text{p}_B(i)
\]

\[
= i^4B_2^{(1)} + i^8B_2^{(2)} + (W + 1)^4B_1^{(1)} + (W + 1)^8B_1^{(2)} + (W + 2)^4B_2^{(1)} + (W + 2)^8B_2^{(2)}.
\]

We notice that each worker cannot obtain information about which matrix the master wants to compute among $B$, because each worker cannot distinguish the variables assigned to each worker and the constant values $x_{i,j}^*$. This is because $B^{(1)}$, $B^{(2)}$, and $B^{(3)}$ are symmetric in the encoding function $\text{p}_B(x)$, and each worker cannot identify that $A$ is encoded with $x = i$ and $x = -i$ for $W_{ij}$ because of the random matrices $R_{A_1}$ and $R_{A_2}$.

In the other case, if the master wants to compute $C = AB^{(2)}$, the encoding function $\text{p}_B(x)$ is given by

\[
\text{p}_B(x) = B_1^{(2)}g(x) + B_2^{(2)}g(x) + B_1^{(1)}g(x_{1,1}^*) + B_2^{(1)}g(x_{2,1}^*) + B_2^{(2)}g(x_{1,2}^*) + B_1^{(2)}g(x_{2,2}^*),
\]

and encoded matrices $\tilde{B}_{ij}$ for $i \in [1 : W]$ and $j \in [1 : 2]$ can be determined as

\[
\tilde{B}_{i,1} = \tilde{B}_{i,2} = \text{p}_B(-i) = \text{p}_B(i)
\]

\[
= i^4B_2^{(1)} + i^8B_2^{(2)} + (W + 1)^4B_1^{(1)} + (W + 1)^8B_1^{(2)} + (W + 2)^4B_2^{(1)} + (W + 2)^8B_2^{(2)}.
\]

Because the symmetry between matrices in a library is guaranteed in the encoded matrices given to each worker, the index of the desired matrix can be hidden from each worker by randomly permuting the evaluation points and the constant values $(x_{1,1}^*, x_{1,2}^*, x_{2,1}^*, x_{2,2}^*)$ between the sets $(x_{i,j})_{i,j=1}^W$ for $i \in [1 : W]$ and $(x_{i,j}^*)_{i,j=1}^W$ for $i \in [1 : 2]$ for every matrix multiplication task.
Given the assigned sub-matrices, each worker $W_i$ computes the matrix multiplications as

$$\mathbf{pc}(x) = \mathbf{pA}(x) \times \mathbf{pB}(x)$$

$$= A_1c^* + A_2c^*x + R_A_1c^*x^2 + R_{A_2}c^*x^3 + A_1B_1^{(1)}x^4 + A_2B_1^{(1)}x^5 + R_A_1B_1^{(1)}x^6 + R_{A_2}B_1^{(1)}x^7 + A_1B_2^{(1)}x^8 + A_2B_2^{(1)}x^9 + R_A_1B_2^{(1)}x^{10} + R_{A_2}B_2^{(1)}x^{11}.$$  

on the evaluation points $x = i$ and $x = -i$, where $c^*$ denotes the constant value $\frac{(W+1)^2B_1^{(2)}+(W+1)^2B_2^{(2)}+(W+2)^2B_1^{(3)}+(W+2)^2B_2^{(3)}}{N}$. Because $\mathbf{pc}(x)$ is an 11th-degree polynomial function, the master should wait for any fastest 12 computation results from the workers to encode the coefficients of $\mathbf{pc}(x)$ by interpolating it from 12 distinct points. The master can obtain the sub-matrices of the final product $C$, which correspond to $A_1B_1^{(1)}, A_2B_1^{(1)}, A_1B_2^{(1)}$, and $A_2B_2^{(1)}$, from the coefficients of $\mathbf{pc}(x)$ (specifically from the coefficients of $x^4, x^5, x^8$, and $x^9$). Thus, the recovery threshold is $R = 12$.

Considering the communication loads, each worker $W_i$ needs to receive four sub-matrices $A_{ij} \in \mathbb{F}_2^{N \times N}$ and $B_{ij} \in \mathbb{F}_2^{N \times \frac{N}{2}}$ for $j \in [1 : 2]$ for a sub-task allocation. However, as has been noted before, $B_{ij} = \tilde{B}_{ij}$ for all $i \in [1 : W]$. Thus, the master can eliminate the redundant part and send a single sub-matrix for $\tilde{B}_{i,1}$ and $\tilde{B}_{i,2}$ to $W_i$. For computation returns, a total of 12 computation results among $\tilde{C}_{ij} \in \mathbb{F}_2^{N \times \frac{N}{2}}, i \in [1 : W], j \in [1 : 2]$ are sent to the master. Therefore, the communication loads in this example are as follows.

$$L_t = \left( \frac{N}{2} \times N \right) \times 2 + \left( \frac{N}{2} \times \frac{N}{2} \right) = \frac{3}{2}N^2.$$ 

$$L_r = \left( \frac{N}{2} \times \frac{N}{2} \right) \times 12 = 3N^2. \quad (10)$$

### B. GENERAL CONSTRUCTION OF SSEC FOR PRIVATE AND SECURE DISTRIBUTED MATRIX MULTIPLICATION

To encode the sub-tasks with a $\frac{1}{mn}$ size of the original task, the master divides the inputs $A$ and $B$ along the row and column sides into $m$ and $n$ pieces, respectively, as

$$A = \begin{bmatrix} A_1 \\ \vdots \\ A_m \end{bmatrix}, \quad B^{(k)} = \begin{bmatrix} B_{1}^{(k)} \ldots B_{n}^{(k)} \end{bmatrix}, \quad \forall k \in [1 : K].$$

The master computes $AB^{(D)}$, then the final product $C = AB^{(D)}$ can be represented as

$$C = \begin{bmatrix} A_1B_1^{(D)} \ldots A_1B_n^{(D)} \\ \vdots \\ \vdots \\ A_mB_1^{(D)} \ldots A_mB_n^{(D)} \end{bmatrix}.$$  

For sub-task encoding, we now introduce the polynomial function $g(x)$ that will be used in the encoding function $\mathbf{pB}(x)$. The polynomial function $g(x)$ needs to satisfy the following conditions.

\textbf{Definition 1:} The polynomial function $g(x)$ used in the encoding function $\mathbf{pB}(x)$ for the SSEC satisfies the following conditions.

i) $g(x)$ is an $(m + L)$-th order polynomial function.

ii) $g(x)$ has at least $W + K - 1$ sets of $L$ points $x_1, x_2, \ldots, x_{W+K-1}$ that satisfy $g(x_1) = \tilde{g}(x_2) = \cdots = g(x_{L})$ for $i \in [1 : W]$ and $g(x_{d+W}) = g(x_{d+W+1}) = \cdots = g(x_{d+W+L})$ for $d \in [1 : K] - [D]$ for a sufficiently large field $\mathbb{F}_2$.

By using the divided sub-matrices and polynomial function $g(x)$, the encoding functions are given by

$$\mathbf{PA}(x) = \sum_{i=1}^{m} A_1x^{i-1} + \sum_{j=1}^{L} R_{A,j}x^{m+j-1},$$

$$\mathbf{PB}(x) = \sum_{k=1}^{n} B_k x^{(D)} g^{(k)}(x) + \sum_{d=1,d \neq D}^{K} \sum_{s=1}^{n} B_{d,s} x^{(D)} g^{(s)}(x_{d+W}),$$

where $R_{A,j} \in \mathbb{F}_2^{N \times N}, j \in [1 : L]$ denote the random matrices generated at the master to maintain the security of the input matrix $A$ and hide the index of the desired matrix $B^{(D)}$ from each worker. It should be noted that $x_{d+W}$ is a constant, thus the second term in $\mathbf{PB}(x)$ is a constant term. Using these encoding functions, the encoded matrices $\tilde{A}_{ij}$ and $\tilde{B}_{ij}$ for $i \in [1 : W]$ and $j \in [1 : L]$ can be determined by setting $x_{ij}$ to satisfy $\tilde{g}(x_{ij}) = w_j$ for $\forall j \in [1 : L]$, and $x_{d+W}$ to satisfy $\tilde{g}(x_{d+W}) = w_{d+W}$ for $d \in [1 : K] - [D]$ and $\forall j \in [1 : L]$ as follows.

$$\tilde{A}_{ij} = \sum_{i=1}^{m} A_{ip} x_{ij}^{p-1} + \sum_{q=1}^{L} R_{A,q} x_{ij}^{m+q-1},$$

$$\tilde{B}_{ij} = \sum_{i=1}^{n} B_{ip} x_{ij}^{(D)} g^{(i)}(x_{ij}) + \sum_{d=1,d \neq D}^{K} \sum_{s=1}^{n} B_{d,s} x_{ij}^{(D)} g^{(s)}(x_{d+W}),$$

Therefore, $\tilde{B}_{ij}$ become constant for all $j \in [1 : L]$, thus the master does not need to separately send $\tilde{B}_{ij}$ for $j \in [1 : L]$ to $W_i$. The master can reduce the task-allocation communication load $L_t$ by allocating sub-tasks $\tilde{C}_{ij} = \tilde{A}_{ij} \tilde{B}_{ij}$ for $j \in [1 : L]$, which requires $\tilde{B}_{ij}$ for $j \in [1 : L]$ of the same value to $W_i$. In this scheme, the data privacy of $B^{(D)}$, i.e., the index of the desired input matrix $D$, can be hidden from each worker, because a symmetry between $B^{(k)}$, $k \in [1 : K]$ including $B^{(D)}$ in a library is guaranteed in the encoded matrices. To preserve data privacy for multiple rounds of matrix multiplication tasks, the evaluation points should not be fixed to be hidden from the workers. Thus, the master must perform a random permutation of the evaluation points and the constant values between the sets $\{x_{ij}\}_{i=1}^{W+K-1}$ for $i \in [1 : W+K-1]$ and $\{x_{d+W}\}_{d=1}^{W+K-1}$ for every multiplication task. We defer the proof of data privacy in an information-theoretic sense in Lemma 2. Moreover, we will also show that the data security on the input matrix $A$ can be guaranteed from each worker in Lemma 1.

\textsuperscript{2}We refer to [29] to introduce several factoring polynomials algorithms to find $g(x)$ satisfying the conditions in Definition 1 over finite fields.
Given the assigned sub-matrices $\tilde{A}_{i,j}$ and $\tilde{B}_{i,j}$ for $j \in [1:L]$, each worker $W_j$ computes the matrix multiplication as

$$
\text{pc}(x) = p_a(x) \times p_b(x) = \left( \sum_{i=1}^{m} A_{i} x^{i-1} + \sum_{j=1}^{L} R_{A_{j}} x^{m+j-1} \right) \times \left( \sum_{k=1}^{n} B_{k}^{(d)} g^k(x_d + w) \right),
$$

(12)
on the evaluation points $x = x_{i,1}, \ldots, x_{i,L}$, and sends the completed results to the master asynchronously. Because $\text{pc}(x)$ is a $(m+L)(n+1)-1$-th degree polynomial function, it can be interpolated from $(m+L)(n+1)$ distinct points of $\text{pc}(x)$ [30]. In addition, the master can obtain the coefficients of $\text{pc}(x)$ using an Reed-Solomon decoding algorithm [31]. Hence, the master should wait for any fastest $(m+L)(n+1)$ computation results from the workers to obtain the coefficients of $\text{pc}(x)$.

The terms $\sum_{k=1}^{K} \sum_{d=1}^{L} \sum_{d'\neq d} \sum_{d''} \sum_{d'''} B_{k}^{(d')} g^k(x_d + w_{d''})$ in (12) are constant, thus these terms only exist at the coefficients of $x^0, x^1, \ldots, x^{m+L-1}$ in $\text{pc}(x)$. The coefficients of $x^{m+L}, \ldots, x^{(m+L)(n+1)-1}$ are the linear combinations of the matrix multiplications of the two matrices between the one among the set $\{A_1, \ldots, A_m, R_{A_1}, \ldots, R_{A_m}\}$ and the other among the set $\{B_1^{(d)}, \ldots, B_{n}^{(d)}\}$, a total of $(m+L)n$ matrices. Thus, these matrices can be achieved at the master from the $(m+L)n$ linear combinations at the coefficients of $x^{m+L}, \ldots, x^{(m+L)(n+1)-1}$ in $\text{pc}(x)$. This implies that the master can obtain all the sub-matrices in the final product $C = AB(D)$ from the multiplications of every pair consisting of one from $\{A_1, \ldots, A_m\}$ and another from $\{B_1^{(d)}, \ldots, B_{n}^{(d)}\}$.

According to the result, the master can decode the final product $C$, thus the recovery threshold is given by

$$
R = (m+L)(n+1).
$$

In this scheme, each worker $W_j$ receives $L$ sub-matrices $\tilde{A}_{i,j} \in \mathbb{F}^{N \times N}$ for $j \in [1:L]$ and a single sub-matrices among $\tilde{B}_{i,j} \in \mathbb{F}^{N \times N}$ for $j \in [1:L]$ for a sub-task allocation, as has been observed before $\tilde{B}_{i,1} = \cdots = \tilde{B}_{i,L}$ for all $i \in [1:W]$. Therefore, the task-allocation communication load $L_t$ is given by

$$
L_t = \frac{N^2}{m} \times L + \frac{N^2}{n}.
$$

(14)

For computation returns, a total of $(m+L)(n+1)$ computation results among $\tilde{C}_{i,j} \in \mathbb{F}^{N \times N}$, $i \in [1:W], j \in [1:L]$ is sent to the master. Thus, the computation-return communication load $L_r$ is given by

$$
L_r = \frac{N^2}{m} (m+L)(n+1).
$$

(15)

It is subsequently shown that data security on $A$ can be guaranteed for each worker in the encoded sub-matrices of the SSEC.

**Lemma 1:** Each worker $W_j$ obtains no information about the input data $A$ from the sub-matrices $\{\tilde{A}_{i,j}\}_{j=1}^{L}$ and $\{\tilde{B}_{i,j}\}_{j=1}^{L}$, i.e., the data security constraint in (4) is satisfied.

**Proof:** Without loss of generality, we prove Lemma 1 for $W_1$, which can be extended to any worker. First, we define the following notations as

$$
\tilde{A}_{i,[1:L]} = \{\tilde{A}_{i,j}\}_{j=1}^{L}, \quad \tilde{B}_{i,[1:L]} = \{\tilde{B}_{i,j}\}_{j=1}^{L},
$$

$$
\tilde{x}_i,[1:L] = \{\tilde{x}_i,j\}_{j=1}^{L}, \quad \tilde{R}_{A_{i}}[1:L] = \{\tilde{R}_{A_{i,j}}\}_{j=1}^{L},
$$

(16)

to simplify the notations in the proof. From the security constraint in (4), we can derive

$$
I\left(\tilde{A}_{i,[1:L]}, \tilde{B}_{i,[1:L]} ; \tilde{A}\right)
$$

$$
= I\left(\tilde{B}_{i,[1:L]} ; \tilde{A}\right) + I\left(\tilde{A}_{i,[1:L]} ; \tilde{B}_{i,[1:L]} | \tilde{A}\right)
$$

$$
= I\left(\tilde{A}_{i,[1:L]} ; \tilde{B}_{i,[1:L]} \right)
$$

(17)

$$
= I\left(\tilde{A}_{i,[1:L]} ; \tilde{B}_{i,[1:L]} \right)
$$

(18)

$$
= I\left(\tilde{A}_{i,[1:L]} ; \tilde{B}_{i,[1:L]} \right)
$$

(19)

$$
= I\left(\tilde{A}_{i,[1:L]} ; \tilde{B}_{i,[1:L]} \right)
$$

(20)

$$
\leq I\left(\tilde{A}_{i,[1:L]} ; \tilde{B}_{i,[1:L]} \right)
$$

(21)

$$
\leq I\left(\tilde{A}_{i,[1:L]} ; \tilde{B}_{i,[1:L]} \right)
$$

(22)

$$
= I\left(\tilde{A}_{i,[1:L]} ; \tilde{B}_{i,[1:L]} \right)
$$

(23)

$$
+ I\left(\tilde{A}_{i,[1:L]} ; \tilde{B}_{i,[1:L]} ; \tilde{x}_i,[1:L] \right)
$$

(24)

$$
\leq I\left(\tilde{A}_{i,[1:L]} ; \tilde{B}_{i,[1:L]} \right)
$$

(25)

$$
+ I\left(\tilde{A}_{i,[1:L]} ; \tilde{B}_{i,[1:L]} ; \tilde{x}_i,[1:L] \right)
$$

(26)

$$
\leq I\left(\tilde{A}_{i,[1:L]} ; \tilde{B}_{i,[1:L]} \right)
$$

(27)

where (19) comes from $\tilde{B}_{i,[1:L]}$ is independent of $A$, (21) and (22) come from conditioning reduces entropy, (23) and (26) follow from the fact that $\tilde{A}_{i,[1:L]}$ is a deterministic function of $A$, $\tilde{x}_i,[1:L]$, and $R_{A_{i}}[1:L]$, and (27) comes from $R_{A_{i}}[1:L]$ is independent of $A$, $\tilde{B}_{i,[1:L]}$, $\tilde{x}_i,[1:L]$. From the above equations, it can be seen that $I\left(\tilde{A}_{i,[1:L]} ; \tilde{B}_{i,[1:L]} \right) = 0$ if

$$
H\left(\tilde{A}_{i,[1:L]} \right) \leq H\left(R_{A_{i}}[1:L]\right).
$$

Thus, we can conclude that

$$
I\left(\tilde{A}_{i,[1:L]} ; \tilde{B}_{i,[1:L]} ; \tilde{A}\right) = 0.
$$

(29)

This completes the proof.

We now provide an important Lemma to show that data privacy can be guaranteed from each worker in the encoded sub-matrices of the SSEC.
Lemma 2: Each worker \( W_i \) obtains no information about the index of the desired input matrix \( B^{(D)} \) from the sub-matrices \( \{ A_{ji} \}^{L}_{j=1} \) and \( \{ B_{ji} \}^{L}_{j=1} \), i.e., the data privacy constraint in (3) is satisfied.

Proof: Without loss of generality, we prove Lemma 2 for \( W_i \), which can be extended to any worker. The simplified notation defined in (16) is used. By a chain rule,

\[
I \left( \tilde{A}_{i,[1:L]} ; x_{i,[1:L]} \right) = I (B;D) + I \left( \tilde{A}_{i,[1:L]} ; D \mid B \right)
\]

First, the matrices in the library \( B \) and the index of the desired matrix \( D \) are independent, thus \( I (B;D) = 0 \). Next, worker \( W_i \) cannot identify the assigned variables \( x_{i,[1:L]} \) from \( \tilde{A}_{i,[1:L]} \) because \( \tilde{A}_{i,[1:L]} \) are encoded with the random matrices \( R_{A_{i,[1:L]}} \) on which \( W_i \) does not have information. This can be derived by the following equations as

\[
I \left( \tilde{A}_{i,[1:L]} ; x_{i,[1:L]} \right) = H \left( \tilde{A}_{i,[1:L]} \right) - H \left( \tilde{A}_{i,[1:L]} \mid x_{i,[1:L]} \right) \\
\leq H \left( \tilde{A}_{i,[1:L]} \right) - H \left( \tilde{A}_{i,[1:L]} \mid x_{i,[1:L]}, A \right) \\
= H \left( \tilde{A}_{i,[1:L]} \right) - H \left( \tilde{A}_{i,[1:L]} \mid x_{i,[1:L]}, A \right) + H \left( x_{i,[1:L]} \mid \tilde{A}_{i,[1:L]}, A, R_{A_{i,[1:L]}} \right) \\
= H \left( \tilde{A}_{i,[1:L]} \right) - I \left( R_{A_{i,[1:L]}} ; \tilde{A}_{i,[1:L]} \mid x_{i,[1:L]}, A \right) \\
= H \left( \tilde{A}_{i,[1:L]} \right) - H \left( R_{A_{i,[1:L]}} \mid x_{i,[1:L]}, A \right) + H \left( R_{A_{i,[1:L]}} \mid x_{i,[1:L]}, A, \tilde{A}_{i,[1:L]} \right) \\
= H \left( \tilde{A}_{i,[1:L]} \right) - H \left( R_{A_{i,[1:L]}} \mid x_{i,[1:L]}, A \right) \\
= H \left( \tilde{A}_{i,[1:L]} \right) - H \left( R_{A_{i,[1:L]}} \right) \\
= 0,
\]

where (32) comes from conditioning reduces entropy, (33) and (36) follow from the fact that \( \tilde{A}_{i,[1:L]} \) is a deterministic function of \( A, x_{i,[1:L]} \), and \( R_{A_{i,[1:L]}} \), and (37) comes from \( R_{A_{i,[1:L]}} \) is independent of \( x_{i,[1:L]} \) and \( A \), and finally (38) comes from (28). Thus, \( \tilde{A}_{i,[1:L]} \) contains no information on \( D \), which implies that \( I \left( \tilde{A}_{i,[1:L]} ; D \mid B \right) = 0 \). Finally, we show that \( \tilde{B}_{i,[1:L]} \) contains no information on \( D \) given \( B \) and \( \tilde{A}_{i,[1:L]} \). From Definition 1, we can derive that all \( g(x_{i,j}) \) for \( j \in [1:L] \) have the same value, and all \( g(x_{d,W}) \) for \( d \in [1:K] \) have the same value. Thus, we can reduce \( g(x_{i,j}) \) for \( j \in [1:L] \) as \( g(x_{i,j}) \) and \( g(x_{d,W}) \) for \( d \in [1:K] \) as \( g(x_{d,W}) \), the encoded sub-matrices \( \tilde{B}_{i,[1:L]} \) given to \( W_i \) is represented as

\[
\tilde{B}_{i,j} = \sum_{k=1}^{n} B^{(D)}_{k} g^{k}(x_{i,j}) + \sum_{d=1, d \neq k}^{K} B^{(d)}_{k} g^{k}(x_{d,W}),
\]

for all \( j \in [1:L] \). Note that all of the partitioned sub-matrices \( B^{(d)}_{k} \) for \( d \in [1:K] \) and \( k \in [1:n] \) are divided into equal sizes and in an equal manner. In addition, the set of points \( x_{i,j} \) for \( i \in [1:W+K] - [W+D] \) and \( j \in [1:L] \) are distinct and randomly chosen from the \( W+K-1 \) sets of \( L \) points given in Definition 1. Moreover, the master performs random permutation between the sets \( \{ x_{i,j} \}^{L}_{j=1} \) for \( i \in [1:W+K] - [W+D] \) for every multiplication tasks, thus a symmetry between \( B^{(d)}_{k} g^{k}(x_{i,j}) \) for \( d \in [1:K] \) and \( i \in [1:W+K] - [W+D] \) is guaranteed in the encoded sub-matrices. This implies that the probability of each realization of \( B_{i,[1:L]} \) does not depend on the index of the desired matrix \( D \). Furthermore, we show that \( \tilde{A}_{i,[1:L]} \) contains no information on \( x_{i,[1:L]} \), which implies that the symmetry between \( B^{(d)}_{k} g^{k}(x_{i,j}) \) in \( B_{i,[1:L]} \) is not destroyed by \( \tilde{A}_{i,[1:L]} \). Thus, \( \tilde{B}_{i,[1:L]} \) contains no information on \( D \) given \( B \) and \( \tilde{A}_{i,[1:L]} \), which means that \( I \left( \tilde{B}_{i,[1:L]} ; D \mid B, \tilde{A}_{i,[1:L]} \right) = 0 \). This completes the proof.

C. MAIN RESULTS FOR PRIVATE AND SECURE DISTRIBUTED MATRIX MULTIPLICATION

From Section III-B, we provide the main results on the recovery threshold, task-allocation communication load, and computation-return communication load of SSEC for private and secure distributed matrix multiplications.

Theorem 1: For a private and secure matrix multiplication task in a straggler-exploiting distributed computing (Scenario A) specified in Section II, SSEC achieves the recovery threshold and communication loads as

\[
R = \left( m + L(n+1) \right),
\]

\[
L_t = \frac{N^2}{m} \times L + \frac{N^2}{n},
\]

\[
L_r = \frac{N^2}{mn} \left( m + L(n+1) \right).
\]

Proof: See Section III-B.

Remark 1 (Scalability): It is worth mentioning that the recovery threshold of SSEC does not scale with the number of workers \( W \). This means that SSEC can ensure a fixed number of required computation returns at a master with the determined \( m, n, \) and \( L \) regardless of the number of workers \( W \) that the master exploits for distributed computation. Thus, a master can efficiently deal with straggling effects by utilizing numerous workers in distributed computing systems. On the other hand, as the number of workers increases, computational complexity for encoding and decoding at a master and communication load will also increase, thus we need to carefully determine the size of a distributed computing system by considering this trade-off between straggler effects, computational complexity, and communication load.

Remark 2 (Reduction in Task-Allocation Communication Load): According to the linear encoding functions in (1), the task-allocation communication load is expected to be \( O \left( \frac{\mu}{n} N^2 L \right) \), where \( \mu \) is determined by the matrix partitioning parameters \( m, n, \) and \( p \), which scales with the number of sub-tasks at each worker \( L \) and the size of both the input matrices \( A \) and \( B \). However, it can be seen
that the task-allocation communication load of the SSEC is 
\( O \left( \frac{1}{4}(N^2L + N^2) \right) \). This is due to the reduction in the task-allocation communication load of the SSEC, which will be described in Section III. From this result, we can claim that this gain in the task-allocation communication load of SSEC will increase as more sub-tasks are assigned to each worker in order to fully exploit the stragglers’ computation results by dividing the sub-tasks smaller.

Remark 3 (Assumptions on Private Matrix Multiplication): In this paper, we assume that the master sends the encoded matrices \( \bar{A}_{ij} \) and \( \bar{B}_{ij} \) for \( j \in [1 : L] \) to each worker \( W_i \). However, because in Scenario A, the workers can share the library \( B \) consisting of the matrices \( \{B^{(k)}\}_{k=1}^L \), the master can only send the queries for \( \bar{B}_{ij} \) that contain a polynomial function and evaluation points for encoding the sub-tasks, as in the previous works for private matrix multiplication [23]–[28]. With this assumption, each worker \( W_i \) can encode the sub-matrices \( \bar{B}_{ij} \) for \( j \in [1 : L] \) from the queries, while not being aware of the identity of the desired matrix. This approach imposes computational burden on workers in encoding the sub-tasks in addition to assigned matrix multiplication tasks. In addition, because the master does not need to send the encoded matrices for \( B \) to the workers, the advantage of SSEC in reducing the task-allocation communication load cannot be guaranteed.

IV. SECURELY STRAGGLER-EXPLOITING CODES FOR FULLY SECURE DISTRIBUTED MATRIX MULTIPLICATION

We dedicate this section to propose the SSEC for fully secure distributed matrix multiplication in a straggler-exploiting scenario.

A. MOTIVATING EXAMPLE

We provide a motivating example for the proposed scheme. We consider the case of \( m = 2, n = 2, L = 2, E = 1 \). Given that \( mn \) equals 4, the original task \( C = AB \) will be divided into sub-tasks with a \( \frac{1}{4} \) size of the original task.

Example 2 (Fully Secure Distributed Matrix Multiplication): A master divides two input matrices \( A \) and \( B \) into half to encode the sub-tasks with a \( \frac{1}{4} \) size of the original task as follows.

\[
A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}, \quad B = \begin{bmatrix} B_1 & B_2 \end{bmatrix}.
\]

In this case, the final product \( C = AB \) can be written as

\[
C = \begin{bmatrix} A_1B_1 & A_1B_2 \\ A_2B_1 & A_2B_2 \end{bmatrix}.
\]

Using the divided sub-matrices as coefficients, the encoding functions of SSEC are given by

\[
p_A(x) = A_1 + A_2x + R_{A_1}x^2 + R_{A_2}x^3, \\
p_B(x) = B_1 + B_2x + R_{B_1}x^2(x),
\]

where \( x \) denotes the variable of the polynomials of \( p_A \) and \( p_B \), and \( R_{A_1} \), \( R_{A_2} \), and \( R_{B_1} \) are random matrices generated at the master to hide the information about input matrices \( A \) and \( B \) from each worker.

To make the encoding function \( p_B(x) \) has the \( W \) sets of the two points that have the same value on \( g(x) \), we will use \( g(x) = x^4 \). Using the encoding functions \( p_A(x) \) and \( p_B(x) \), the encoded matrices \( \bar{A}_{i,j} \) and \( \bar{B}_{i,j} \) for \( i \in [1 : W \} \) and \( j \in [1 : 2] \) can be determined by setting \( x_1 = -i \) and \( x_2 = i \) for \( i \in [1 : W] \). The encoded matrices are

\[
\begin{align*}
\bar{A}_{i,1} &= p_A(-i) = A_1 - iA_2 + i^2R_{A_1} - i^3R_{A_2}, \\
\bar{A}_{i,2} &= p_A(i) = A_1 + iA_2 + i^2R_{A_1} + i^3R_{A_2}, \\
\bar{B}_{i,1} &= \bar{B}_{i,2} = p_B(-i) = p_B(i) = B_1 + i^2B_2 + i^3R_B.
\end{align*}
\]

It should be noted that each worker cannot obtain information about input matrices \( A \) and \( B \) from the allocated sub-matrices \( \bar{A}_{i,1}, \bar{A}_{i,2}, \bar{B}_{i,1}, \) and \( \bar{B}_{i,2} \). This is because the random matrices \( R_{A_1}, R_{A_2}, \) and \( R_B \) are added to the allocated sub-matrices.

Given the assigned sub-matrices, each worker \( W_i \) computes the matrix multiplication as

\[
p_C(x) = p_A(x) \times p_B(x) = A_1B_1 + A_2B_1x^2 + R_{A_1}B_1x^3 + R_{A_2}B_1x^3 + A_1B_2x^4 + A_2B_2x^4 + R_{A_1}B_2x^5 + R_{A_2}B_2x^5 + A_1B_1x^6 + A_2B_1x^6 + R_{A_1}B_1x^7 + R_{A_2}B_1x^7 + A_1B_2x^7 + A_2B_2x^7 + R_{A_1}B_2x^8 + R_{A_2}B_2x^8 + R_{A_1}B_1x^9 + R_{A_2}B_1x^9 + R_{A_1}B_2x^9 + R_{A_2}B_2x^9,
\]

on the evaluation points \( x = i \) and \( x = -i \). The decoding process is similar to that of Example 1. Because \( p_C(x) \) is an 11th-degree polynomial function, the master should wait for any fastest 12 computation results from the workers to decode the coefficients of \( p_C(x) \) by interpolating it from 12 distinct points. The master can obtain the sub-matrices of the final product \( C \), which correspond to \( A_1B_1^{(1)}, A_2B_1^{(1)}, A_1B_2^{(1)}, \) and \( A_2B_2^{(1)} \) from the coefficients of \( x^4, x^5, x^8, \) and \( x^9 \). Thus, the recovery threshold is \( R = 12 \).

Considering the communication loads, each worker \( W_i \) should receive two sub-matrices \( \bar{A}_{ij} \in \mathbb{F}_2^{N \times N} \) for \( j \in [1 : 2] \) and a single sub-matrix of \( \bar{B}_{ij} \in \mathbb{F}_2^{N \times N} \) for \( j \in [1 : 2] \), because \( \bar{B}_{i,1} = \bar{B}_{i,2} \) for all \( i \in [1 : W] \). For computation returns, a total of 12 computation results among \( \bar{C}_{ij} \in \mathbb{F}_2^{N \times N}, i \in [1 : W], j \in [1 : 2] \) should be sent to the master to decode the final product. Therefore, the communication loads in this example are as

\[
L_c = \left( \frac{N}{2} \times N \right) \times 2 + \left( \frac{N}{2} \times \frac{N}{2} \right) = \frac{3}{2}N^2,
\]

\[
L_r = \left( \frac{N}{2} \times \frac{N}{2} \right) \times 12 = 3N^2.
\]

B. GENERAL CONSTRUCTION OF SSEC FOR FULLY SECURE DISTRIBUTED MATRIX MULTIPLICATION

To encode the sub-tasks with a \( \frac{1}{mn} \) size of the original task, the master divides the inputs \( A \) row-wise and \( B \) column-wise.
into $m$ and $n$ pieces, respectively, as
\[
A = \begin{bmatrix} A_1 \\ \vdots \\ A_m \end{bmatrix}, \quad B = \begin{bmatrix} B_1 & \cdots & B_n \end{bmatrix}.
\]
The final product $C = AB$ can be represented by
\[
C = \begin{bmatrix} A_1B_1 & \cdots & A_1B_n \\ \vdots & \ddots & \vdots \\ A_mB_1 & \cdots & A_mB_n \end{bmatrix}.
\] (44)

For sub-task encoding, we introduce the polynomial function $g(x)$ that will be used in the encoding function $p_B(x)$. The polynomial function $g(x)$ should satisfy the following conditions.

**Definition 2:** The polynomial function $g(x)$ used in the encoding function $p_B(x)$ for the SSEC satisfies the following conditions.

i) $g(x)$ is an $(m + LE)$-th order polynomial function.

ii) $g(x)$ has at least $W$ sets of $L$ points $x_{1,1}, x_{1,2}, \ldots, x_{W,L}$ that satisfies $g(x_{i,1}) = g(x_{i,2}) = \cdots = g(x_{i,L})$ for $i \in [1 : W]$ for a sufficiently large field $\mathbb{F}$.

Using the divided sub-matrices and polynomial function $g(x)$, the encoding functions are given by
\[
p_A(x) = \sum_{i=1}^{m} A_i x^{i-1} + \sum_{j=1}^{LE} R_{Aj} x^{m+j-1},
\]
\[
p_B(x) = \sum_{k=1}^{n} B_k g^{k-1}(x) + \sum_{l=1}^{R} R_{Bl} g^{n+l-1}(x).
\]
Using these encoding functions, the encoded matrices $\tilde{A}_{i,j}$ and $\tilde{B}_{i,j}$ for $i \in [1 : W]$ and $j \in [1 : L]$ can be determined by setting $x_{i,j}$ to satisfy $g(x_{i,j}) = x_{i,j}$ for $j \in [1 : L]$ as
\[
\tilde{A}_{i,j} = \sum_{p=1}^{m} A_{pj} x_{i,j}^{p-1} + \sum_{q=1}^{LE} R_{Aq} x_{i,j}^{m+q-1},
\]
\[
\tilde{B}_{i,j} = \sum_{k=1}^{n} B_k g^{k-1}(x_{i,j}) + \sum_{l=1}^{R} R_{Bl} g^{n+l-1}(x_{i,j}).
\]
The master can reduce the task-allocation communication load $L_t$ by allocating sub-tasks $\tilde{C}_{i,j} = A_{i,j} B_{i,j}$ for $j \in [1 : L]$, which requires $\tilde{B}_{i,j}$ for $j \in [1 : L]$, which becomes constant to $W_l$. It should be noted that the data security of the input matrices $A$ and $B$ can be guaranteed from any $E$ colluding workers because of the added random matrices $R_{Aj}$ for $j \in [1 : LE]$ and $R_{Bl}$ for $l \in [1 : E]$. The detailed proof is provided in Lemma 3.

Given the assigned sub-matrices $\tilde{A}_{i,j}$ and $\tilde{B}_{i,j}$ for $j \in [1 : L]$, each worker $W_l$ computes the matrix multiplication as
\[
\text{pc}(x) = p_A(x) \times p_B(x)
= \left( \sum_{i=1}^{m} A_i x^{i-1} + \sum_{j=1}^{LE} R_{Aj} x^{m+j-1} \right) \times \left( \sum_{k=1}^{n} B_k g^{k-1}(x) + \sum_{l=1}^{R} R_{Bl} g^{n+l-1}(x) \right).
\] (45)

on the evaluation points $x = x_{1,1}, \ldots, x_{1,L}$. Because $\text{pc}(x)$ is a $(m + LE)(n + E) - 1$-th degree polynomial function, it can be interpolated from $(m + LE)(n + E)$ distinct points of $\text{pc}(x)$. Hence, the master must wait for any fastest $(m + LE)(n + E)$ computation results from the workers to obtain the evaluation $C = AB$.

**Lemma 3:** The $E$-colluding workers $W_{E}, \forall E \subset [1 : W], |E| = E$ obtain no information about the input data $A$ and $B$ from the sub-matrices $\{A_{i,j}\}_{j=1}^{L}$ or $\{B_{i,j}\}_{j=1}^{L}$, i.e., the data security constraint in (5) is satisfied.

**Proof:** Without loss of generality, we prove Lemma 3 for $W_{E}, E \subset [1 : W], |E| = E$, which can be generalized to any $E$ workers. We use the simplified notations defined in (16) and the additional notation as
\[
R_{Bl} = \{R_{Bl}\}_{l=1}^{E}.
\] (49)
First, we use the important property of SSEC in that the sub-matrices $\tilde{B}_{i,[1:L]}$ have the same value for
each $i \in [1: W]$. Thus,

$$H \left( \{ \tilde{B}_{i,1} \}_{i \in \mathcal{E}} \right) = H \left( \{ \tilde{B}_{i,1} \}_{i \in \mathcal{E}} \right) + H \left( \{ \tilde{B}_{i,1} \}_{i \in \mathcal{E}} \right)$$

because $\tilde{B}_{i,1} = \cdots = \tilde{B}_{i,L}$ for all $i \in \mathcal{E}$.

Considering the sub-matrices $\{ \tilde{B}_{i,1} \}_{i \in \mathcal{E}}$, we have

$$I \left( \{ \hat{B}_{i,1} \}_{i \in \mathcal{E}} ; \hat{B}_{i,1} \right)$$

$$= H \left( \{ \hat{B}_{i,1} \}_{i \in \mathcal{E}} \right) - H \left( \{ \hat{B}_{i,1} \}_{i \in \mathcal{E}} \right)$$

$$\leq H \left( \{ \hat{B}_{i,1} \}_{i \in \mathcal{E}} \right) - H \left( \{ \hat{B}_{i,1} \}_{i \in \mathcal{E}} \right)$$

$$= H \left( \{ \hat{B}_{i,1} \}_{i \in \mathcal{E}} \right) - H \left( \{ \hat{B}_{i,1} \}_{i \in \mathcal{E}} \right)$$

$$+ H \left( \{ \hat{B}_{i,1} \}_{i \in \mathcal{E}} \right) - R_{\mathcal{B}_{i,1}}$$

$$= H \left( \{ \hat{B}_{i,1} \}_{i \in \mathcal{E}} \right) - R_{\mathcal{B}_{i,1}}$$

$$= H \left( \{ \hat{B}_{i,1} \}_{i \in \mathcal{E}} \right) - R_{\mathcal{B}_{i,1}}$$

where (53) comes from (50), (54) comes from conditioning reduces entropy, (55) and (58) follow from the fact that $\{ \tilde{B}_{i,1} \}_{i \in \mathcal{E}}$ is a deterministic function of $B_{i,1}$, $(g(x_{i,1}))_{i \in \mathcal{E}}$, and $R_{\mathcal{B}_{i,1}}$, and (59) comes from $R_{\mathcal{B}_{i,1}}$ is independent of $B_{i,1}$ and $(g(x_{i,1}))_{i \in \mathcal{E}}$. From (59), it is clear that $I \left( \{ \tilde{B}_{i,1} \}_{i \in \mathcal{E}} ; B \right) = 0$ if $H \left( \{ \tilde{B}_{i,1} \}_{i \in \mathcal{E}} \right) = H \left( \{ \tilde{B}_{i,1} \}_{i \in \mathcal{E}} \right)$. Because $\{ \tilde{B}_{i,1} \}_{i \in \mathcal{E}}$ and $R_{\mathcal{B}_{i,1}}$ have the same size, and $R_{\mathcal{B}_{i,1}}$ are randomly generated matrices that can have the maximum entropy of the given size, we can guarantee that

$$H \left( \{ \tilde{B}_{i,1} \}_{i \in \mathcal{E}} \right) \leq H \left( \{ \tilde{B}_{i,1} \}_{i \in \mathcal{E}} \right).$$

Thus, we can conclude that

$$I \left( \{ \tilde{B}_{i,1} \}_{i \in \mathcal{E}} ; B \right) = 0.$$
Therefore, it is evident that the security constraint in (5) can be satisfied if \( H \left( \tilde{A}_{i;[1:L]} \right) \leq H \left( R_{A_{i;[1:L]}} \right) \). Because \( \tilde{A}_{i;[1:L]} \) and \( R_{A_{i;[1:L]}} \) have the same size and are randomly generated matrices with the maximum entropy of a given size, we can guarantee that

\[
H \left( \tilde{A}_{i;[1:L]} \right) \leq H \left( R_{A_{i;[1:L]}} \right).
\]  

(74)

Thus, we can conclude that the security constraint in (5) is satisfied. This completes the proof.

\[ \square \]

C. MAIN RESULTS FOR FULLY SECURE DISTRIBUTED MATRIX MULTIPLICATION

From Section IV-B, we derive the main results on the recovery threshold, task-allocation communication load, and computation-return communication load of SSEC for fully secure distributed matrix multiplications.

**Theorem 2:** For a fully secure matrix multiplication task in a straggler-exploiting distributed computing (Scenario B) specified in Section II, SSEC achieves the recovery threshold and communication loads as follows.

\[
\begin{align*}
R &= (m + LE)(n + E), \\
L_t &= \frac{N^2}{m} \times L + \frac{N^2}{n}, \\
L_r &= \frac{N^2}{mn} (m + LE)(n + E).
\end{align*}
\]  

(75)

**Proof:** See Section IV-B.

**Remark 4:** Note that Remark 1 and Remark 2 can also be applied to fully secure distributed matrix multiplication (Scenario B) for SSEC.

**Remark 5 (Comparison):** We compare the recovery threshold and the communication loads of SSEC with those of the existing schemes, namely gap additive secure polynomial codes (GASP) [22] and secure generalized polydot (SGPD) [26], for fully secure distributed matrix multiplication. Although GASP and SGPD were not proposed for a straggler-exploiting scenario, we simply modify them for a straggler-exploiting scenario by allocating LE sub-tasks, which were designed to preserve data security from LE colluding workers, to E colluding workers with L sub-tasks each.

Fig. 2 presents the performances on the recovery threshold and the communication loads of the proposed SSEC, GASP, and SGPD for the minimum recovery threshold of the three schemes. It can be seen that SSEC have the lower recovery threshold than SGPD for the case of \( p = 1 \) where the input matrices are divided in the same way as in SSEC. This comes from the fact that in SSEC, the number of allocated sub-matrices for the sub-tasks at each worker is reduced so that the additional overhead on recovery threshold to preserve data security from workers can be lowered. However, SSEC shows the greater recovery threshold than GASP. In GASP, the recovery threshold could be reduced by the special approach on a polynomial function for minimizing the number of non-zero coefficients, whereas SSEC are designed to reduce the communication load while sacrificing the loss of the recovery threshold.

Fig. 2 (b) shows the task-allocation communication load of the three schemes. Compared to the other schemes, SSEC has lower task-allocation communication load. Moreover, this gap gets larger as the number of the sub-tasks at each worker \( L \) increases, as shown in Remark 2. This implies that by using SSEC, we can alleviate the communication overhead in a distributed computing system on allocating the sub-tasks to the workers.

Fig. 2 (c) shows the computation-return communication load of three schemes. When we set \( m = n = 1 \) in SGPD to have the minimum recovery threshold, the communication load to return each computation result becomes \( N^2 \), while the other schemes (including SGPD with \( p = 1 \)) have \( \frac{N^2}{mn} \) of communication load for returning each computation result. Thus, SGPD with \( m = n = 1 \) has greater computation-return communication load than other schemes although it has lower recovery threshold. The other schemes including SSEC, GASP and SGPD with \( p = 1 \) have the same communication load to return each computation result, thus the computation-return communication loads differ in the difference of recovery threshold which corresponds to the number of required computation results at the master.

In summary, SSEC outperforms the other coding schemes in reducing the task-allocation communication load. Let us simply express the total execution time for distributed matrix multiplication as

\[
T_{\text{sum}} = T_A + T_P + T_{R},
\]  

(76)

where \( T_A \), \( T_P \), and \( T_R \) denote the required times for task allocation from a master to workers, computation at workers, and returning computation results from workers to a master, respectively, for some constants \( c_A > 0, c_P > 0, \) and \( c_R > 0 \). If \( c_A \gg c_P \), where the communication for task allocation is heavy in distributed computing, SSEC can show better performance in terms of the total execution time than GASP and SGPD. On the other hand, if \( c_P \gg c_A \), where the workers require considerable computing time, GASP and SGPD for \( m = n = 1 \) may show better performance than SSEC. In addition, if \( c_R \gg c_A, c_P \), where the communication for returning computation results is heavy, SGPD for \( m = n = 1 \) may show worse performance than others although it achieves the minimum recovery threshold.

V. COMPLEXITY ANALYSIS

We dedicate this section to evaluate the computational complexities for the encoding and decoding procedures of the proposed SSEC, and the computational costs for the workers.
A. SSEC FOR PRIVATE AND SECURE DISTRIBUTED MATRIX MULTIPLICATION

1) ENCODING COMPLEXITY

To encode the sub-matrices $A_{i,j}$ for $j \in [1 : L]$ for each worker $W_j$, the master adds $m + L$ matrices of sizes $\mathbb{F}_n^N \times N$. In addition, to encode a single sub-matrix among $B_{i,1}, \ldots, B_{i,L}$, the master adds $n + (K - 1)n = Kn$ matrices of size $\mathbb{F}_n^N \times N$. Thus, the computational complexity of encoding the sub-matrices for each worker is $O((m + L)N^2 L + KnN^2)$. The overall encoding complexity for all $W$ workers is $O \left( N \left( L + L^2 + K \right) N^2 \right)$. We assume that the sizes of input matrices $A$ and $B$ are very large, thus $N$ is very much greater than $L$, $W$, and $K$, i.e., $N \gg L, W, K$. Thus, the encoding complexity for SSEC is much less than the computational complexity for the whole task $C = AB$ which corresponds to $O(N^3)$. 

2) COMPUTATIONAL COST AT WORKERS

Each worker multiplies two matrices $A_{i,j}$ and $B_{i,j}$ which have dimensions of $\mathbb{F}_n^N \times N$ and $\mathbb{F}_n^N \times 2$. This requires $\frac{N^3}{mn} \cdot N \cdot \frac{N}{n} = \frac{N^3}{mn}$ multiplication operations. Thus, the computational cost for a sub-task of the workers is $O \left( \frac{N^3}{mn} \right)$. Let us calculate the computation time at the workers when the computation time is $\mu$ where

$$\mu = \frac{N}{mn} \cdot N \cdot \frac{N}{n} = \frac{N^3}{mn}$$

multiplication operations. Thus, the computational cost for a sub-task of the workers is $O \left( \frac{N^3}{mn} \right)$. Let us calculate the computation time at the workers when the computation time is $\mu$ where

$$\mu = \frac{N}{mn} \cdot N \cdot \frac{N}{n} = \frac{N^3}{mn}$$

Thus, the computation time distribution for a sub-task with SSEC is given by

$$Pr[T_s \leq t] = 1 - e^{-\mu mn(t - \frac{\alpha}{mn})}, \text{ for } t \geq \frac{\alpha}{mn}. \quad (79)$$

From the computation time distribution, we can derive the expected computation time $E[T_s]$ for a sub-task at each worker as

$$E[T_s] = \frac{\alpha}{mn} + \frac{1}{\mu mn} = \frac{1}{mn} \left( \frac{\alpha + 1}{\mu} \right). \quad (80)$$

Finally, the expected computation time $E[T_{k-1}]$ for $k$ sub-tasks at each worker is given by

$$E[T_{k-1}] = k \cdot E[T_s] = \frac{k}{mn} \left( \frac{\alpha + 1}{\mu} \right). \quad (81)$$

3) DECODING COMPLEXITY

To decode the final product $C = AB^{(D)}$, the master solves a polynomial interpolation problem on a $((m + L)(n + 1) - 1)$-th degree polynomial for $\frac{N^2}{mn}$ elements. Because the polynomial interpolation algorithm has a complexity of $O(k \log^2 k)$ for a $k$-th degree polynomial [32], the decoding complexity for polynomial interpolation is given by

$$\mathcal{O} \left( \frac{N^2}{mn} ((m + L)(n + 1) - 1) \log^2 ((m + L)(n + 1) - 1) \right). \quad (82)$$

In addition, the master needs to solve the linear equations on $(m + L)n$ elements for $\frac{N^2}{mn}$ times for each matrix element. Solving a linear equation on $(m + L)n$ using the Gaussian elimination method has a computational complexity of $\mathcal{O}(mL^3n^3)$. In conclusion, the decoding complexity of the master is given by

$$\mathcal{O} \left( \frac{N^2}{mn} ((m + L)(n + 1) - 1) \log^2 ((m + L)(n + 1) - 1) + (m + L)^3n^3 \right).$$
B. SSEC FOR FULLY SECURE DISTRIBUTED MATRIX MULTIPLICATION

1) ENCODING COMPLEXITY
To encode the sub-matrices $\tilde{A}_{i,j}$ for $j \in [1:L]$ for each worker $W_i$, the master adds $m + LE$ matrices of sizes $F_N^N \times N$. In addition, to encode a single sub-matrix among $\tilde{B}_{i,1}, \ldots, \tilde{B}_{i,L}$, the master adds $n + E$ matrices of size $F_N^{N \times 2}$. Thus, the computational complexity to encode the sub-matrices for all $W$ workers is given by

$$O \left( W \left( \frac{N^2}{m} L + \frac{n + E}{n} \right) \right).$$

With SSEC, the encoding complexity and task-allocation communication load can be reduced by overlapping $L$ sub-matrices for $\tilde{B}_{i,j}, j \in [1:L]$ into a single matrix of the same size. If the proposed SSEC is not used in the encoding functions, the computational complexity to encode the sub-matrices is given by

$$O \left( W \left( L + \frac{L^2 E}{m} + 1 + \frac{E}{n} \right) N^2 \right).$$

(83)

Compared to that, SSEC requires half of the encoding complexity under the assumptions of $m = n$ and $L \gg 1$.

2) COMPUTATIONAL COST AT WORKERS
Each worker multiplies two matrices $\tilde{A}_{i,j}$ and $\tilde{B}_{i,j}$ that have dimensions of $F_N^N \times N$ and $F_N^{N \times 2}$, which requires $N^3$ operations. Thus, the computational cost for a sub-task at the workers is $O \left( \frac{N^3}{mn} \right)$. The computation time for the sub-tasks at each worker can be derived as the same value as in Section V-A.

3) DECODING COMPLEXITY
To decode the final product $C = AB$, the master solves a polynomial interpolation problem on a $((m + LE)(n + E) - 1)$th-degree polynomial for $N^2$ elements. Thus, the decoding complexity for polynomial interpolation is given by

$$O \left( \frac{N^2}{mn} ((m + LE)(n + E) - 1) \log^2 ((m + LE)(n + E) - 1) \right).$$

(85)

In addition, the master should solve linear equations on $(m + LE)(n + E)$ elements for $N^2$ times for each matrix element, which has a computational complexity of $O \left( \frac{N^2}{mn} (m + LE)^3 (n + E)^3 \right)$. In conclusion, the decoding complexity of the master is given by

$$O \left( \frac{N^2}{mn} \left( ((m + LE)(n + E) - 1) \log^2 ((m + LE)(n + E) - 1) \right. \right.

$$

$$\left. + (m + LE)^3 (n + E)^3 \right).$$

It can be seen that the decoding complexity at the master is highly related to the recovery threshold, thus it may have a similar tendency between SSEC and the other existing schemes as in the comparison of the recovery threshold in Remark 5.3

VI. CONCLUSION
In this paper, we explored a secure coded computation scheme for matrix multiplication tasks in straggler-exploiting distributed computing. We proposed a new coded computation scheme, namely SSEC, which can preserve data security and privacy from workers and efficiently reduce the task-allocation communication load as well. It was shown that by SSEC, data security and privacy can be ensured at the encoded sub-matrices sent to the workers to allocate the sub-tasks. Finally, we discussed the performance of the SSEC in terms of the recovery threshold and communication loads between a master and workers. A future work could be to modify the SSEC to assign computation tasks with a common encoded matrix to exploit the advantage of broadcasting the encoded matrices to multiple workers and to preserve data privacy and data security from multiple colluding workers.

ACKNOWLEDGMENT
An earlier version of this paper was presented in part at the International Symposium on Information Theory (ISIT) 2021 [1] [DOI: 10.1109/ACCESS.2021.3135581].

REFERENCES
[1] H. Yang, S. Hong, and J. Lee, “Private and secure coded computation in straggler-exploiting distributed matrix multiplication,” in Proc. IEEE Int. Symp. Inf. Theory (ISIT), Melbourne, VIC, Australia, Jul. 2021.
[2] J. Dean and L. A. Barroso, “The tail at scale,” Commun. ACM, vol. 56, no. 2, pp. 74–80, Feb. 2013.
[3] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, “Speeding up distributed machine learning using codes,” IEEE Trans. Inf. Theory, vol. 64, no. 3, pp. 1514–1529, Mar. 2018.
[4] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, “Polynomial codes: An optimal design for high-dimensional coded matrix multiplication,” in Proc. Adv. Neural Inf. Process. Syst. (NIPS), Dec. 2017, pp. 4403–4413.
[5] S. Dutta, M. Fahim, F. Haddadpour, H. Jeong, V. Cadambe, and P. Grover, “On the optimal recovery threshold of coded matrix multiplication,” IEEE Trans. Inf. Theory, vol. 66, no. 1, pp. 278–301, Jan. 2020.
[6] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, “Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding,” IEEE Trans. Inf. Theory, vol. 66, no. 3, pp. 1920–1933, Mar. 2020.
[7] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, “A unified coding framework for distributed computing with straggling servers,” in Proc. IEEE Globecom Workshops (GC Wkshps), Washington, DC, USA, Dec. 2016.
[8] S. Dutta, V. Cadambe, and P. Grover, “Short-dot: Computing large linear transforms distributedly using coded short dot products,” in Proc. 34th Annu. Conf. Neural Inf. Process. Syst. (NIPS), Barcelona, Spain, Dec. 2016.
[9] K. Lee, C. Suh, and K. Ramchandran, “High-dimensional coded matrix multiplication,” in Proc. IEEE Int. Symp. Inf. Theory (ISIT), Aachen, Germany, Jun. 2017.

3 It should also be noted that SGPD with $m = n = 1$ would require higher decoding complexity than expected in the comparison of recovery threshold, since it has the larger size of the computation results from the workers than the other schemes including SSEC.
[10] H. Park, K. Lee, J.-Y. Sohn, C. Suh, and J. Moon, “Hierarchical coding for distributed computing,” in Proc. IEEE Int. Symp. Inf. Theory (ISIT), Vail, CO, USA, Jun. 2018.

[11] A. Reisizadeh, S. Prakash, R. Pedarsani, and A. S. Avestimehr, “Coded computation over heterogeneous clusters,” IEEE Trans. Inf. Theory, vol. 65, no. 7, pp. 4227–4242, Jul. 2019.

[12] I. S. Reed and G. Solomon, “Polynomial codes over certain finite fields,” J. Soc. Ind. Appl. Math., vol. 8, no. 2, pp. 300–304, Jun. 1960.

[13] S. Kiani, N. Ferdinand, and S. C. Draper, “Exploitation of stragglers in coded computation,” in Proc. IEEE Int. Symp. Inf. Theory (ISIT), Vail, CO, USA, Jun. 2018.

[14] A. B. Das, L. Tang, and A. Ramamoorthy, “CILE: Codes for coded computation that leverage stragglers,” in Proc. IEEE INFOCOM Workshop (ITW), Guangzhou, China, Nov. 2018.

[15] B. Hasircioğlu, J. Gomez-Vilardebo, and D. Gunduz, “Bivariate polynomial coding for efficient distributed matrix multiplication,” Jan. 2020, arXiv:2001.07277.

[16] S. Hong, H. Yang, and J. Lee, “Squeezed polynomial codes: Communication-efficient coded computation in straggler-exploiting distributed matrix multiplication,” IEEE Access, vol. 8, pp. 190516–190528, 2020.

[17] S. H. Hong, H. Yang, and J. Lee, “Chebyshev polynomial codes: Task entanglement-based coding for distributed matrix multiplication,” in Proc. Int. Conf. Mach. Learn. (ICML), Jun. 2021, pp. 4319–4327.

[18] H. Yang and J. Lee, “Secure distributed computing with straggling servers using polynomial codes,” IEEE Trans. Inf. Forensics Security, vol. 14, no. 1, pp. 141–150, Jan. 2019.

[19] W.-T. Chang and R. Tandon, “On the capacity of secure distributed matrix multiplication,” Jun. 2018, arXiv:1806.00469.

[20] Q. Yu, S. Li, N. Raviv, S. M. M. Kalan, M. Soltanolkotabi, and A. S. Avestimehr, “Lagrange coded computing: Optimal design for resiliency, security, and privacy,” in Proc. 22nd Int. Conf. Artif. Intell. Statist. (AISTATS), Okinawa, Japan, Apr. 2019.

[21] R. Bitar, Y. Xing, Y. Keshtkarjahromi, V. Dasari, S. El Rouayheb, and H. Seferoglu, “Private and rateless adaptive coded matrix-vector multiplication,” Sep. 2019, arXiv:1909.12611.

[22] R. G. L. D’Oliveira, S. El Rouayheb, and D. Karpuk, “GASP codes for secure distributed matrix multiplication,” IEEE Trans. Inf. Theory, vol. 66, no. 7, pp. 4038–4050, Jul. 2020.

[23] W.-T. Chang and R. Tandon, “On the upload versus download cost for secure and private matrix multiplication,” Jun. 2019, arXiv:1906.10684.

[24] Q. Yu and A. S. Avestimehr, “Entangled polynomial codes for secure, private, and batch distributed matrix multiplication: Breaking the ‘Cubic’ barrier,” Jun. 2020, arXiv:2001.05101.

[25] Q. Yu and A. S. Avestimehr, “Coded computing for resilient, secure, and privacy-preserving distributed matrix multiplication,” IEEE Trans. Commun., vol. 69, no. 1, pp. 59–72, Jan. 2021.

[26] M. Aliasgari, O. Simeone, and J. Kliewer, “Private and secure distributed matrix multiplication with flexible communication load,” IEEE Trans. Inf. Forensics Security, vol. 15, pp. 2722–2734, 2020.

[27] M. Kim and J. Lee, “Private secure coded computation,” IEEE Commun. Lett., vol. 23, no. 11, pp. 1918–1921, Nov. 2019.

[28] M. Kim, H. Yang, and J. Lee, “Private coded matrix multiplication,” IEEE Trans. Inf. Forensics Security, vol. 15, pp. 1434–1443, 2020.

[29] J. von zur Gathen and D. Panario, “Factoring polynomials over finite fields: A survey,” J. Symbolic Comput., vol. 31, nos. 1–2, pp. 3–17, Jan. 2001.

[30] K. Kedlaya and C. Umans, “Fast polynomial factorization and modular composition,” SIAM J. Comput., vol. 40, no. 6, pp. 1767–1802, 2011.

[31] F. Didier, “Efficient erasure decoding of Reed–Solomon codes,” Jan. 2009, arXiv:0901.1886.

[32] H. T. Kung, “Fast evaluation and interpolation,” Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep., Jan. 1973.