This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2846601, IEEE Transactions on Information Forensics and Security

Secure Distributed Computing with Straggling Servers Using Polynomial Codes

Heecheol Yang, *Student Member, IEEE* and Jungwoo Lee, *Senior Member, IEEE*

***Abstract*—In this paper, we consider a secure distributed computing scenario in which a master wants to perform matrix multiplication of confidential inputs with multiple workers in parallel. In such a setting, a master does not want to reveal information about the two input matrices to the workers in an information-theoretic sense. We propose a secure distributed computing scheme that can efficiently cope with straggling effects by applying polynomial codes on sub-tasks assigned to workers. The achievable recovery threshold, i.e., the number of workers that a master needs to wait for to get the final product, of our proposed scheme is revealed to be order-optimal to the number of workers. Moreover, we derive the achievable recovery threshold of the proposed scheme is within a constant multiplicative factor from information-theoretic lower bound. As a byproduct, we extend our strategy to secure distributed computing for convolution tasks on confidential data.**

***Index Terms*—distributed computing, data security, polynomial codes**

I. Introduction

Recently, there has been much interest in machine learning and big data analytics, along with their applications to image processing, collaborative filtering, and so on. To cope with a large computation load and memory requirement to process massive datasets, distributed computing frameworks have been developed to complete whole tasks with multiple servers (workers) in parallel, which can leverage a large number of servers to speed up processing a big task. However, in a distributed computing scenario, it has been reported a major bottleneck to increase computation time is waiting time for slowest or even unresponsive workers, which is referred to as *stragglers* [1]. One simple solution to alleviate the impact of stragglers, i.e., straggling effect, is to allow redundancies when assigning tasks to the workers [2]. Furthermore, coding techniques may be exploited to mitigate the straggling effect for various distributed computing scenarios such as distributed matrix multiplication [3]-[8], distributed convolution [9], and distributed gradient descent [10]-[11].

In this work, we consider data security in distributed computing frameworks. Preserving security of input data in a distributed computing model could be a critical issue when 1) there could exist an eavesdropper with access to the link between a master and workers, 2) a master needs to exploit suspicious but useful workers, and 3) information of input data to be computed must be protected from workers, e.g., personal location information and medical data. There have been significant studies on data security in a distributed computing model. Numerous papers have considered private multi-party computation model in which data security is guaranteed for workers to privately own their local data, and not to reveal them to other workers [12]-[14]. Recently, the authors in [15], [16] have studied the computing model of securely outsourcing data in which each party has own confidential data, and the master wants to perform computations on confidential data without requesting for disclosing the data to the parties. In this work, we consider a *secure* distributed computing problem in which a master wants to perform a matrix multiplication task on confidential data with non-colluding workers in parallel, while not revealing information about confidential data to workers in an information-theoretic sense. In a secure distributed computing scenario, straggling effects must be controlled to quickly finish entire tasks as in a conventional distributed computing scenario.

In this paper, we raise fundamental questions on the secure distributed computing problem regarding the minimum number of workers the master needs to wait for to obtain the final product for the worst-case scenario (will be defined as *recovery threshold*). We also discuss how to design a secure distributed computing scheme that can achieve the optimum recovery threshold. The contributions of this work are threefold.

- We first derive the upper bound on the recovery threshold of the secure distributed computing problem, i.e., the minimum number of arbitrary workers that can guarantee decodability of the final output at the master.
- Secondly, we prove a lower bound on the recovery threshold of the secure distributed computing.
- As a consequence, we claim that optimal recovery threshold can be characterized within a factor of 2.

We reveal the upper bound on the recovery threshold by proposing an achievable secure distributed computing scheme for matrix multiplication using *polynomial codes* [17]. The polynomial code in the proposed secure distributed computing scheme is carefully designed to preserve security of input data from workers, and to address straggling effects effectively as well, by modifying existing polynomial code for (security-free) distributed computing scheme that can achieve optimal recovery threshold [8]. Achievable recovery threshold of our proposed scheme is order-optimal to the number of workers, i.e., it does not scale with the number of workers, that is believed to the first result on the secure distributed computing problem in literature.

We extend our approach to the secure distributed computing scenario for convolution tasks. Beyond extending polynomial codes for distributed matrix multiplication, we decrease the

H. Yang and J. Lee are with the Communications and Machine Learning Lab., Department of Electrical and Computer Engineering, Seoul National University, Seoul, 08826, Korea (e-mail: hee2070@snu.ac.kr, junglee@snu.ac.kr).

This work is in part supported by Basic Science Research Program (NRF-2017R1A2B2007102) through NRF funded by MSIP, Technology Innovation Program (10051928) funded by MOTIE, Bio-Mimetic Robot Research Center funded by DAPA (UD130070ID), INMAC, and BK21-plus.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2846601, IEEE Transactions on Information Forensics and Security

achievable recovery threshold of the secure distributed convolution by leveraging inherent property of convolution tasks, where the sums of the convolutions of sub-vectors are needed to get the overall convolution result. The proposed scheme can achieve order-optimality to the number of workers on the recovery threshold. With the proposed scheme we demonstrate that optimal recovery threshold for convolution tasks can be characterized within a factor of 3. To the best of our knowledge, this is the first result to use coding techniques to perform distributed convolution while preserving data security from workers.

From results on achievable recovery threshold, we analyze overall runtime distribution of the proposed scheme, by focusing on the waiting time of returning sub-products from workers at the master. We estimate the mean waiting time under the assumption that the computing time at workers follows exponential distribution [25]. In particular, compared with an uncoded scheme that allocates whole tasks to workers as much as possible without considering straggling effects, we demonstrate that our scheme can reduce the overall runtime by efficiently mitigating straggling effects.

Related Work: There are a large body of work on the secure distributed computing problem in the computer science and machine learning literature. Several works have proposed securely outsourcing computation schemes with expensive homomorphic encryption [18], [19] or secret key generation [20]. Secure distributed computing schemes that address straggling effects have been proposed using secret sharing [21], [22] or staircase codes [7], [23] ensuring information-theoretic security on input data. However, these schemes assume that only one input of two matrices for multiplication is regarded as confidential data that must be preserved from workers, while our scheme using polynomial codes regards the two inputs as confidential data. The authors in [24] considered secure distributed matrix multiplication on confidential inputs, yet did not consider straggling effects. To the best of our knowledge, our paper is the first in proposing a secure distributed computing scheme for matrix multiplication and convolution tasks preserving data security of two inputs from the workers and efficiently mitigating the straggler effects as well.

Organization: The remainder of this paper is organized as follows. In Section II, we introduce the system model and formulate a secure distributed computing problem on recovery threshold. In Section III, we state our main results and implications. In Section IV, we propose a new secure distributed computing scheme using polynomial codes. In Section V, we extend the proposed scheme to a secure distributed computing problem for convolution. In Section VI, we estimate and analyze overall runtime distribution of the proposed scheme. Finally, in Section VII, we conclude this paper.

Notation: For $a, b \in \mathbb{Z}$, $[a:b]$ denotes $\{a, a+1, \ldots, b\}$.

II. System Model & Problem Formulation

We consider a secure distributed computing scenario in which a master wants to perform a matrix multiplication task $C = A^T B$ on confidential data $A \in \mathbb{F}_q^{s\times r}$ and $B \in \mathbb{F}_q^{s\times t}$ for a sufficiently large finite field $\mathbb{F}_q$. We assume that either of the two inputs A and B is a tall matrix, i.e., $s \geq r$ or $s \geq t$, to ensure that the final product C is a full-rank matrix. The master divides this multiplication task into smaller computation tasks and assigns to N non-colluding workers $\{\mathrm{W}_i\}_{i=1}^N$, each of which can store $\frac{1}{m}$ times the size of A and $\frac{1}{n}$ times the size of B. A master encodes A and B into N sub-matrices $\{\tilde{A}_i\}_{i=1}^N$ and $\{\tilde{B}_i\}_{i=1}^N$, where $\tilde{A}_i \in \mathbb{F}_q^{s\times \frac{r}{m}}$ and $\tilde{B}_i \in \mathbb{F}_q^{s\times \frac{t}{n}}$, respectively, and sends $\tilde{A}_i$ and $\tilde{B}_i$ to W_i for all $i \in [1:N]$. Workers obtain no information about A and B from their assigned sub-matrices in an information-theoretic sense [29], i.e.,

$$I(\tilde{A}_i; A) = 0, \quad I(\tilde{B}_i; B) = 0, \quad \forall i \in [1:N]. \tag{1}$$

Each worker W_i computes a sub-product $\tilde{C}_i = \tilde{A}_i^T \tilde{B}_i$ and returns it to the master. A master waits only for sub-products from a subset of workers to cope with the straggling effect, and recovers the final product C given these sub-products. We illustrate the system model of a secure distributed computing for matrix multiplication with N workers in Fig. 1.

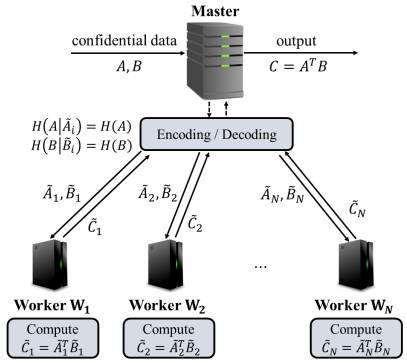

Fig. 1. System model of secure distributed computing for matrix multiplication of two confidential data A and B with N workers.

We formally define the secure distributed computing problem of matrix multiplication. First, we denote the sets of encoded sub-matrices of A and B as

$$\tilde{\mathbf{A}} \triangleq \{\tilde{A}_1, \tilde{A}_2, \ldots, \tilde{A}_N\}, \quad \tilde{\mathbf{B}} \triangleq \{\tilde{B}_1, \tilde{B}_2, \ldots, \tilde{B}_N\}. \tag{2}$$

We say matrices A and B are *securely encoded* for distributed computing using N workers if each of the elements in $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$ has zero information about A and B, respectively (1). Each of the sub-matrices $\tilde{A}_i$ and $\tilde{B}_i$ for all $i \in [1:N]$ should satisfy the storage constraint at the workers, i.e., $\tilde{A}_i \in \mathbb{F}_q^{s\times \frac{r}{m}}$ and $\tilde{B}_i \in \mathbb{F}_q^{s\times \frac{t}{n}}$ for all $i \in [1:N]$. Using this terminology, as in [8] we define the *recovery threshold* $k(\tilde{\mathbf{A}}, \tilde{\mathbf{B}})$ of securely encoded sets as the minimum number of workers k where the master can recover C given the sub-products $\tilde{C}_i = \tilde{A}_i^T \tilde{B}_i$ from any k workers. The goal of the secure distributed computing problem of matrix multiplication is to find the optimum recovery threshold K^*, that can be denoted by

$$K^* \triangleq \min_{\tilde{\mathbf{A}}, \tilde{\mathbf{B}}} k(\tilde{\mathbf{A}}, \tilde{\mathbf{B}}), \tag{3}$$

and to find securely encoded sets $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$ that can achieve optimum recovery threshold.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2846601, IEEE Transactions on Information Forensics and Security

III. Main Results

In this section, we introduce our main results on the recovery threshold of secure distributed computing problem. We also reveal notable implications of our work.

The first theorem presents a recovery threshold that can be achieved by the proposed distributed computing scheme using polynomial codes, yielding an upper bound on optimal recovery threshold.

Theorem 1. *For secure distributed computing of matrix multiplication task $C = A^T B$ using N workers, where each worker can store $\frac{1}{m}$ fraction of A and $\frac{1}{n}$ fraction of B without information about A and B, optimum recovery threshold K^* is upper bounded by*

$$K^* \leq mn + m + n. \tag{4}$$

Proof: The achievable scheme will be given in Section IV. ∎

Remark 1. Without security constraint, the optimum recovery threshold K^* for a distributed matrix multiplication task $C = A^T B$, where each worker can store $\frac{1}{m}$ fraction of A and $\frac{1}{n}$ fraction of B, is given by

$$K^* = mn. \tag{5}$$

Optimum recovery threshold can be achieved by distributed computing strategy using polynomial codes [8]. In a secure distributed computing scenario, we reveal that the recovery threshold $mn+m+n$ can be achieved in Theorem 1, implying that the master should wait for extra $m+n$ workers to preserve information about two confidential data from workers with our scheme. Additional $m+n$ workers for the master to wait for can be regarded as the *price of security* in our scheme for a secure distributed computing scenario.

We also provide an information-theoretic lower bound on optimal achievable recovery.

Theorem 2. *For secure distributed computing of matrix multiplication task $C = A^T B$ using N workers, where each worker can store $\frac{1}{m}$ fraction of A and $\frac{1}{n}$ fraction of B without information about A and B, optimum recovery threshold K^* is lower bounded by*

$$K^* \geq mn + 1. \tag{6}$$

Proof: See Appendix. ∎

The proof of Theorem 2 is that the master cannot obtain any information about the final product from each of the sub-products, i.e.,

$$I\left(C; \tilde{C}_i\right) = 0, \quad \forall i \in [1:N], \tag{7}$$

since each of the sub-matrices assigned to the workers has zero information about two inputs A and B (1). Theorem 2 reveals that the master should wait for at least one more worker for a secure distributed computing compared to a non-secure distributed computing scenario. However, this lower bound does not match with the derived upper bound, thus we state that optimal recovery threshold can be characterized within a constant factor by comparing two bounds.

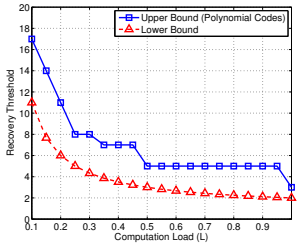

Fig. 2. Tradeoff between computation load and recovery threshold for a secure distributed computing scenario using $N = 20$ workers.

Theorem 3. *For secure distributed computing of matrix multiplication task $C = A^T B$ using N workers, where each worker can store $\frac{1}{m}$ fraction of A and $\frac{1}{n}$ fraction of B without information about A and B, optimum recovery threshold K^* is characterized within a factor of 2, i.e.,*

$$\frac{1}{2} K_{polynomial} < K^* \leq K_{polynomial}, \tag{8}$$

where $K_{polynomial} = mn + m + n$ is the upper bound on the recovery threshold K^*, that can be achieved by polynomial codes.

Proof: See Appendix. ∎

Remark 2. From results on the recovery threshold in Theorems 1 and 2, we provide a fundamental trade-off between recovery threshold and computation load at each worker, by assuming that there is no storage constraint at the workers, and A and B can be divided into $\frac{1}{m}$ fraction and $\frac{1}{n}$ fraction, respectively, i.e., $r \gg m$ and $t \gg n$. We define the computation load L at each worker as computational complexity of $\tilde{C}_i = \tilde{A}_i^T \tilde{B}_i$, normalized by computational complexity of $C = A^T B$, i.e., $L \triangleq \frac{1}{mn}$. Recovery threshold of a distributed computing with the computation load L at each worker is denoted as $K(L)$. By considering the assumption of $m, n \in \mathbb{N}$, achievable recovery threshold in Theorem 1 is converted to the upper bound on $K(L)$ as

$$K(L) \leq \left\{ \begin{array}{ll} \min & mn + m + n \\ \text{s.t.} & \frac{1}{L} \leq mn \leq N, \quad m, n \in \mathbb{N} \end{array} \right. \tag{9}$$

Meanwhile, the proof of Theorem 2 can be easily extended to the case of $\frac{1}{L} = mn \in \mathbb{R}$, thus the lower bound on $K(L)$ is given by

$$K(L) \geq \frac{1}{L} + 1. \tag{10}$$

For example, Fig. 2 shows a tradeoff between computation load and recovery threshold for a secure distributed computing using $N = 20$ workers.

To validate the novelty of our proposed scheme using polynomial codes for matrix multiplication of two confidential inputs, we introduce a comparable secure distributed computing scheme using secret sharing.

We can generalize a secure distributed computing scheme using secret sharing scheme for matrix multiplication of two confidential inputs. By using $(1, m+1, \sqrt{N})$ and $(1, n+1, \sqrt{N})$ secret sharing schemes in [22], respectively, a master encodes A and B into $\sqrt{N}$ secret shares $\{S_i^A\}_{i=1}^{\sqrt{N}}$ and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2846601, IEEE Transactions on Information Forensics and Security

$\{S_j^B\}_{j=1}^{\sqrt{N}}$, each of which has $\frac{1}{m}$ times the size of A and $\frac{1}{n}$ times the size of B to meet the information-theoretic bound on the size of the secret shares. To extend the secret sharing scheme from a single confidential input to two confidential inputs for multiplication, we assume that workers are divided into $\sqrt{N}$ groups, each of which consists of $\sqrt{N}$ workers. The workers in the jth group are assigned to compute $(S_i^A)^T S_j^B$ for all $i \in [1:\sqrt{N}]$. The master can recover the sub-product $A^T S_j^B$ if $m+1$ workers in the jth group finish their tasks, and it can recover the final product $C = A^T B$ from the sub-products of $n+1$ groups. In this case, we can demonstrate that secret sharing achieves a recovery threshold of

$$K_{\text{secret-sharing}} = \sqrt{N}n + m(\sqrt{N} - n) + 1 = \Theta(\sqrt{N}). \quad (11)$$

Remark 3. According to Theorem 1, achievable recovery threshold of secure distributed matrix multiplication task using polynomial codes is order-optimal to the number of workers, i.e., it does not scale with the number of workers N.

$$K_{\text{polynomial}} = mn + m + n = \Theta(1). \quad (12)$$

Existing secure distributed computing schemes using secret sharing scheme or staircase codes [7] regard only one of the two input matrices as confidential data. Thus, simple extensions of these schemes for the scenario wherein the two inputs are confidential cannot achieve order-optimality of the recovery threshold to the number of workers, due to the worst-case scenario. To the contrary, secure distributed computing scheme using polynomial codes can achieve order-optimality by carefully encoding the two confidential inputs, which is not influenced by the worst-case scenario.

IV. Secure Distributed Computing using Polynomial Codes for Matrix Multiplication

We now prove Theorem 1 by presenting our secure distributed computing scheme using polynomial codes. In a proposed scheme, a master assigns sub-tasks of matrix multiplication to workers, but the master does not reveal information about two input matrices to workers. Let us introduce a motivating example to provide intuition, and reveal the general description of our proposed scheme.

A. Motivating Example

We first provide a motivating example of $m = n = 1$ case to reveal the main idea of our secure distributed computing scheme using polynomial codes. Consider a secure distributed matrix multiplication $C = A^T B$ on confidential data A and B using $N = 5$ workers. Each worker W_i, $\forall i \in [1:5]$ stores two securely encoded sub-matrices $\tilde{A}_i$ and $\tilde{B}_i$ which are encoded and sent by a master as

$$\begin{aligned}
\tilde{A}_1 &= R_A + 1 \cdot A, \quad \tilde{B}_1 = R_B + 1 \cdot B,\\
\tilde{A}_2 &= R_A + 2 \cdot A, \quad \tilde{B}_2 = R_B + 2 \cdot B,\\
\tilde{A}_3 &= R_A + 3 \cdot A, \quad \tilde{B}_3 = R_B + 3 \cdot B,\\
\tilde{A}_4 &= R_A + 4 \cdot A, \quad \tilde{B}_4 = R_B + 4 \cdot B,\\
\tilde{A}_5 &= R_A + 5 \cdot A, \quad \tilde{B}_5 = R_B + 5 \cdot B,
\end{aligned}$$

where $\tilde{A}_i$ and $\tilde{B}_i$ have the same size as A and B, respectively. The two random matrices R_A and R_B are generated independently of A and B by the master, and the elements of R_A and R_B are uniformly drawn from independent and identically distributed (i.i.d.) random variables over $\mathbb{F}_q$. Workers have no information about A and B since the randomly generated matrices R_A and R_B are not known to workers.

Each worker W_i computes $\tilde{C}_i = \tilde{A}_i^T \tilde{B}_i$ and returns the following result to the master.

$$\begin{aligned}
\tilde{C}_1 &= \tilde{A}_1^T \tilde{B}_1 = R_A^T R_B + 1 \cdot A^T R_B + 1 \cdot R_A^T B + 1^2 \cdot A^T B,\\
\tilde{C}_2 &= \tilde{A}_2^T \tilde{B}_2 = R_A^T R_B + 2 \cdot A^T R_B + 2 \cdot R_A^T B + 2^2 \cdot A^T B,\\
\tilde{C}_3 &= \tilde{A}_3^T \tilde{B}_3 = R_A^T R_B + 3 \cdot A^T R_B + 3 \cdot R_A^T B + 3^2 \cdot A^T B,\\
\tilde{C}_4 &= \tilde{A}_4^T \tilde{B}_4 = R_A^T R_B + 4 \cdot A^T R_B + 4 \cdot R_A^T B + 4^2 \cdot A^T B,\\
\tilde{C}_5 &= \tilde{A}_5^T \tilde{B}_5 = R_A^T R_B + 5 \cdot A^T R_B + 5 \cdot R_A^T B + 5^2 \cdot A^T B.
\end{aligned}$$

These sub-products can be represented as

$$\begin{bmatrix} \tilde{C}_1 \\ \tilde{C}_2 \\ \tilde{C}_3 \\ \tilde{C}_4 \\ \tilde{C}_5 \end{bmatrix} = \begin{bmatrix} 1^0 & 1^1 & 1^2 \\ 2^0 & 2^1 & 2^2 \\ 3^0 & 3^1 & 3^2 \\ 4^0 & 4^1 & 4^2 \\ 5^0 & 5^1 & 5^2 \end{bmatrix} \begin{bmatrix} R_A^T R_B \\ A^T R_B + R_A^T B \\ A^T B \end{bmatrix}. \quad (13)$$

Since workers store the sub-matrices that have the same size as A and B, i.e., $m = 1$ and $n = 1$, achievable recovery threshold K of our scheme is $K = 1 \cdot 1 + 1 + 1 = 3$. This means that the master can recover the final product C with any possible set of 3 sub-products $\tilde{C}_i$ from the fastest three workers. The coefficient matrix of the sub-products from workers (13) is a Vandermonde matrix. A square Vandermonde matrix is invertible if all parameters are distinct, thus a sub-matrix of the coefficient matrix of the sub-products with any three rows is always invertible for a sufficiently large finite field $\mathbb{F}_q$. Consequently, the master can recover $R_A^T R_B$, $A^T R_B + R_A^T B$, and $A^T B$ from any three sub-products from the workers by matrix inversion. Let us assume that the master receives sub-products from the fastest three workers W_1, W_2, and W_3. The master has

$$\begin{bmatrix} \tilde{C}_1 \\ \tilde{C}_2 \\ \tilde{C}_3 \end{bmatrix} = \begin{bmatrix} 1^0 & 1^1 & 1^2 \\ 2^0 & 2^1 & 2^2 \\ 3^0 & 3^1 & 3^2 \end{bmatrix} \begin{bmatrix} R_A^T R_B \\ A^T R_B + R_A^T B \\ A^T B \end{bmatrix} \quad (14)$$

$$\Rightarrow \begin{bmatrix} R_A^T R_B \\ A^T R_B + R_A^T B \\ A^T B \end{bmatrix} = \begin{bmatrix} 1^0 & 1^1 & 1^2 \\ 2^0 & 2^1 & 2^2 \\ 3^0 & 3^1 & 3^2 \end{bmatrix}^{-1} \begin{bmatrix} \tilde{C}_1 \\ \tilde{C}_2 \\ \tilde{C}_3 \end{bmatrix}$$

Hence, the master can recover $C = A^T B$ by inverting the equation (14).

On the other hand, the master can also recover the final product by solving a polynomial interpolation problem, requiring more efficient decoding process. Sub-products from workers are the value of the 2nd-degree polynomial $h(x)$ at point $x = i$, where $h(x)$ is given by

$$h(x) = R_A^T R_B + x(A^T R_B + R_A^T B) + x^2 A^T B. \quad (15)$$

Thus, the master can recover coefficients of $h(x)$, $R_A^T R_B$, $A^T R_B + R_A^T B$, and $A^T B$, using any possible set of three sub-products by interpolating $h(x)$ given its values at 3 points. Consequently, the master recovers the final product $C = A^T B$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2846601, IEEE Transactions on Information Forensics and Security

B. General Description (Proof of Theorem 1)

We generalize the secure distributed computing scheme using polynomial codes that achieves the upper bound on optimum recovery threshold K^* in Theorem 1. We divide the whole procedure of the secure distributed computing into three steps: *encoding* at the master, *computing* at workers, and last, *decoding* at the master.

1) Encoding: If we divide two input matrices A and B into m and n fractions, respectively, A and B can be represented as

$$A = [A_1 A_2 \cdots A_m], \quad B = [B_1 B_2 \cdots B_n].$$

The master securely encodes two inputs with the fractions of them and randomly generated matrix $R_A \in \mathbb{F}_q^{s \times \frac{r}{m}}$ and $R_B \in \mathbb{F}_q^{s \times \frac{t}{n}}$ consisting of the elements drawn from i.i.d. random variables, each of which is generated independently of A and B with the same size of the fractions of A and B, respectively. Specifically, the master assigns the variables x_i , $\forall i \in [1 : N]$ to each worker such that all x_i's are distinct in $\mathbb{F}_q$ (in the motivating example, $x_i = i$ is assigned to worker W_i), and generates the securely encoded sets $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$ using polynomial codes on $\{R_A, A_1, \ldots, A_m\}$ and $\{R_B, B_1, \ldots, B_n\}$, respectively. The securely encoded sub-matrices $\tilde{A}_i$ and $\tilde{B}_i$ assigned to worker W_i are represented as

$$\begin{aligned} \tilde{A}_i &= R_A x_i^0 + \sum_{j=1}^{m} A_j x_i^j, \\ \tilde{B}_i &= R_B x_i^0 + \sum_{k=1}^{n} B_k x_i^{k(m+1)-1}, \quad \forall i \in [1 : N]. \end{aligned} \tag{16}$$

The degrees of the polynomial code for two securely encoded sets are carefully chosen to ensure that all the terms with $A_j^T B_k, \forall j \in [1 : m], \forall k \in [1 : n]$ in the sub-product $\tilde{C}_i = \tilde{A}_i^T \tilde{B}_i$ have different exponents of x_i. We can claim that data security can be preserved from the workers when the workers receive the encoded sub-matrices in (16).

Lemma 1. *Workers obtain no information about input data from the sub-matrices in* (16)*, i.e., the data security constraint in* (1) *is satisfied.*

Proof: See Appendix. ∎

2) Computing: After workers store securely encoded inputs sent by the master, each worker W_i computes the sub-product

$$\begin{aligned} \tilde{C}_i &= \tilde{A}_i^T \tilde{B}_i \\ &= R_A^T R_B x_i^0 + \sum_{j=1}^{m} A_j^T R_B x_i^j + \sum_{k=1}^{n} R_A^T B_k x_i^{k(m+1)-1} \\ &\quad + \sum_{j=1}^{m} \sum_{k=1}^{n} A_j^T B_k x_i^{j+k(m+1)-1}, \quad \forall i \in [1 : N], \end{aligned} \tag{17}$$

and returns it to the master. It should be noted that these sub-products are the value of the $(mn + m + n - 1)$th-degree polynomial $h(x)$ at point $x = x_i$, where $h(x)$ is given by

$$\begin{aligned} h(x) &= R_A^T R_B x^0 + \sum_{j=1}^{m-1} A_j^T R_B x^j + (A_m^T R_B + R_A^T B_1) x^m \\ &\quad + \sum_{k=2}^{n} R_A^T B_k x^{k(m+1)-1} + \sum_{j=1}^{m} \sum_{k=1}^{n} A_j^T B_k x^{j+k(m+1)-1}. \end{aligned} \tag{18}$$

3) Decoding: The master decodes the final product after receiving the sub-products from the fastest $mn + m + n$ workers. It can recover all coefficients of $h(x)$ in (18) using any possible set of $mn+m+n$ sub-products from the fastest $mn+m+n$ workers finishing their tasks by interpolating $h(x)$ given its values at $mn+m+n$ points. It is worth mentioning that the problem of polynomial interpolation for decoding polynomial codes at the master can be solved efficiently by using existing decoding algorithms for Reed-Solomon codes [26], [27]. From the coefficients of $h(x)$, the master gets the final output $C = A^T B$ as

$$C = A^T B = \begin{bmatrix} A_1^T B_1 & A_1^T B_2 & \cdots & A_1^T B_n \\ A_2^T B_1 & A_2^T B_2 & \cdots & A_2^T B_n \\ \vdots & \vdots & \ddots & \vdots \\ A_m^T B_1 & A_m^T B_2 & \cdots & A_m^T B_n \end{bmatrix}. \tag{19}$$

Note that the degrees of the sub-matrices assigned to workers are carefully chosen to ensure that all terms with $A_j^T B_k, \forall j \in [1 : m], \forall k \in [1 : n]$ in the sub-product have different exponents of x_i, thus all the elements of the final product matrix in (19) can be recovered at the master. The overall procedure of the secure distributed computation using polynomial codes for matrix multiplication is described in Fig. 3.

V. Extension to Secure Distributed Computing for Convolution

We dedicate this section to propose an extension of secure distributed computation using polynomial codes for a convolution task. We consider a secure distributed convolution scenario in which a master wants to perform a convolution task $\mathbf{c} = \mathbf{a} * \mathbf{b}$ using N workers on two confidential data $\mathbf{a} \in \mathbb{F}_q^{tm}$ and $\mathbf{b} \in \mathbb{F}_q^{tn}$. If we divide two input vectors into m and n fractions, respectively, they can be represented as

$$\mathbf{a} = [\mathbf{a}_1 \mathbf{a}_2 \cdots \mathbf{a}_m], \quad \mathbf{b} = [\mathbf{b}_1 \mathbf{b}_2 \cdots \mathbf{b}_n],$$

where $\mathbf{a}_j \in \mathbb{F}_q^t$, $\mathbf{b}_k \in \mathbb{F}_q^t$ for all $i \in [1 : m]$ and $k \in [1 : n]$. To exploit inherent property of the convolution task in our scheme, sub-vectors of $\mathbf{a}$ and $\mathbf{b}$ assigned to workers should have the same size. We provide an achievable recovery threshold on the secure distributed convolution scenario as follows.

Theorem 4. *For secure distributed computing of convolution task $\mathbf{c} = \mathbf{a} * \mathbf{b}$ using N workers, where each worker can store $\frac{1}{m}$ fraction of $\mathbf{a}$ and $\frac{1}{n}$ fraction of $\mathbf{b}$ without information about $\mathbf{a}$ and $\mathbf{b}$, the optimal recovery threshold K_{conv}^* is upper bounded by*

$$K_{conv}^* \leq max\{m, n\} + m + n. \tag{20}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2846601, IEEE Transactions on Information Forensics and Security

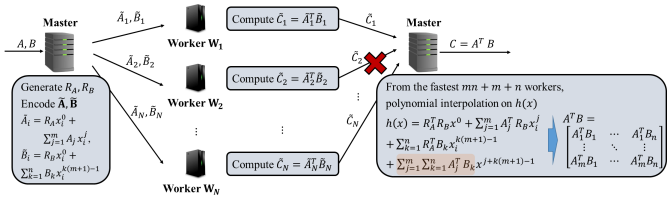

Fig. 3. The overall procedure of secure distributed computation using polynomial codes for matrix multiplication $C = A^T B$: 1) Encoding at the master, 2) Computing at workers, and 3) Decoding at the master. Worker W_2 represents one of the straggler nodes which is not included in the fastest $mn + m + n$ workers computing the assigned sub-tasks.

To achieve recovery threshold of $\max\{m, n\} + m + n$ for a secure distributed convolution task, we modify distributed computing strategy using polynomial codes for matrix multiplication in Section IV.

1) Encoding: The master randomly generates two vectors $\mathbf{r}_a \in \mathbb{F}_q^t$ and $\mathbf{r}_b \in \mathbb{F}_q^t$, that have the same size of the fractions of $\mathbf{a}$ and $\mathbf{b}$. With two random vectors $\mathbf{r}_a$ and $\mathbf{r}_b$, and fractions of $\mathbf{a}$ and $\mathbf{b}$, the master generates securely encoded sets of $\mathbf{a}$ and $\mathbf{b}$, and assigns them to workers. Securely encoded sub-vectors $\tilde{\mathbf{a}}_i$ and $\tilde{\mathbf{b}}_i$ assigned to worker W_i are represented as

$$\tilde{\mathbf{a}}_i = \sum_{j=1}^{m} \mathbf{a}_j x_i^{j-1} + \mathbf{r}_a x_i^{m+n-1}, \tag{21}$$
$$\tilde{\mathbf{b}}_i = \sum_{k=1}^{n} \mathbf{b}_k x_i^{k-1} + \mathbf{r}_b x_i^{m+n-1}, \quad \forall i \in [1:N].$$

Contrary to the secure distributed computing strategy for matrix multiplication, degrees of the polynomial code for fractions of $\mathbf{a}$ and $\mathbf{b}$ are carefully chosen for the convolution terms $\mathbf{a}_{j-k} * \mathbf{b}_k$ to have the same exponent of x_i (x_i^{j-2}), to improve the recovery threshold by leveraging inherent property of a convolution task. In addition, degrees of the polynomial code $m + n - 1$ for the randomly generated vectors $\mathbf{r}_a$ and $\mathbf{r}_b$ are chosen to ensure the decodability of $\mathbf{c} = \mathbf{a} * \mathbf{b}$ at the master, provided that all the terms for the convolution of the fractions of $\mathbf{a}$ and $\mathbf{b}$ in $\tilde{\mathbf{c}}_i = \tilde{\mathbf{a}}_i * \tilde{\mathbf{b}}_i$ do not have the same exponent of x_i with the convolution terms including $\mathbf{r}_a$ or $\mathbf{r}_b$.

2) Computing: Each worker W_i computes the sub-task $\tilde{\mathbf{c}}_i = \tilde{\mathbf{a}}_i * \tilde{\mathbf{b}}_i$ from the assigned vectors $\tilde{\mathbf{a}}_i$ and $\tilde{\mathbf{b}}_i$ in (21) as

$$\begin{aligned}\tilde{\mathbf{c}}_i &= \tilde{\mathbf{a}}_i * \tilde{\mathbf{b}}_i \\ &= \sum_{j=1}^{m}\sum_{k=1}^{n} \mathbf{a}_j * \mathbf{b}_k x_i^{j+k-2} + \sum_{j=1}^{m} \mathbf{a}_j * \mathbf{r}_b x_i^{j-1+m+n-1} \\ &\quad + \sum_{k=1}^{n} \mathbf{r}_a * \mathbf{b}_k x_i^{m+n-1+k-1} + \mathbf{r}_a * \mathbf{r}_b x_i^{2(m+n-1)}.\end{aligned} \tag{22}$$

If the worker finishes its convolution task, it returns $\tilde{\mathbf{c}}_i$ to the master. Note that all the terms of x_i in $\sum_{j=1}^{m}\sum_{k=1}^{n} \mathbf{a}_j * \mathbf{b}_k x_i^{j+k-2}$ have different exponents of x_i from the other terms including $\mathbf{r}_a$ or $\mathbf{r}_b$. These sub-vectors from workers are the value of the $(2m + 2n - 2)$th-degree polynomial $h(x)$ at point $x = x_i$,

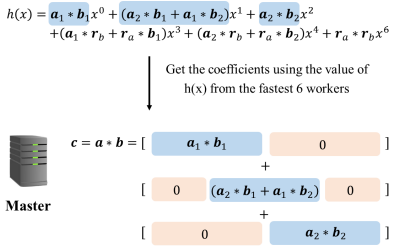

Fig. 4. An illustration of the decoding process at the master in the distributed convolution task for a simple case of $m = 2$ and $n = 2$.

where $h(x)$ is given by

$$\begin{aligned}h(x) &= \sum_{j=1}^{m}\sum_{k=1}^{n} \mathbf{a}_j * \mathbf{b}_k x^{j+k-2} + \sum_{j=1}^{m} \mathbf{a}_j * \mathbf{r}_b x^{j+m+n-2} \\ &\quad + \sum_{k=1}^{n} \mathbf{r}_a * \mathbf{b}_k x^{k+m+n-2} + \mathbf{r}_a * \mathbf{r}_b x^{2(m+n-1)}.\end{aligned} \tag{23}$$

We note that the coefficients of $x^{\max\{m,n\}+m+n-1}, \ldots, x^{2m+2n-3}$ are zero, thus all the coefficients of $h(x)$ can be recovered from the $\max\{m, n\} + m + n$ values of $h(x)$.

3) Decoding: The master can recover all the coefficients of $h(x)$ using any possible set of $\max\{m, n\}+m+n$ sub-products from the fastest $\max\{m, n\} + m + n$ workers finishing their tasks by interpolating $h(x)$ given its values at $\max\{m, n\} + m + n$ points. The convolution terms of the fractions of $\mathbf{a}$ and $\mathbf{b}$ can be represented as

$$\sum_{j=1}^{m}\sum_{k=1}^{n} \mathbf{a}_j * \mathbf{b}_k x^{j+k-2} = \sum_{j=2}^{m+n} \sum_{k=\max\{1,j-m\}}^{\min\{j-1,n\}} \mathbf{a}_{j-k} * \mathbf{b}_k x^{j-2}. \tag{24}$$

From coefficients of these terms, the master can recover the final output $\mathbf{c} = \mathbf{a} * \mathbf{b}$ by adding them with proper shift by zero-padding.

For example, let us consider a simple case of $m = 2$ and $n = 2$, wherein workers are assigned $\frac{1}{4}$ fractions of the whole convolution task. In this case, workers return the value of $h(x)$,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2846601, IEEE Transactions on Information Forensics and Security

where $h(x)$ is given by

$$\begin{aligned} h(x) &= \sum_{j=1}^{2}\sum_{k=1}^{2} \mathbf{a}_j * \mathbf{b}_k x^{j+k-2} + \sum_{j=1}^{2} \mathbf{a}_j * \mathbf{r}_b x^{j+2} \\ &\quad + \sum_{k=1}^{2} \mathbf{r}_a * \mathbf{b}_k x^{k+2} + \mathbf{r}_a * \mathbf{r}_b x^6 \\ &= \mathbf{a}_1 * \mathbf{b}_1 x^0 + (\mathbf{a}_2 * \mathbf{b}_1 + \mathbf{a}_1 * \mathbf{b}_2) x^1 + \mathbf{a}_2 * \mathbf{b}_2 x^2 \\ &\quad + (\mathbf{a}_1 * \mathbf{r}_b + \mathbf{r}_a * \mathbf{b}_1) x^3 + (\mathbf{a}_2 * \mathbf{r}_b + \mathbf{r}_a * \mathbf{b}_2) x^4 \\ &\quad + \mathbf{r}_a * \mathbf{r}_b x^6. \end{aligned}$$

By the six sub-products from the fastest six workers, the master can recover the coefficients of $h(x)$. Hence, the master gets $\mathbf{a}_1 * \mathbf{b}_1$, $\mathbf{a}_2 * \mathbf{b}_1 + \mathbf{a}_1 * \mathbf{b}_2$, and $\mathbf{a}_2 * \mathbf{b}_2$. From these sub-results, the master can recover the final product $\mathbf{c} = \mathbf{a} * \mathbf{b}$ with proper shift by zero-padding as illustrated in Fig. 4.

In addition, we reveal the order-optimality of the achievable recovery threshold to the number of workers, and characterize optimal recovery threshold K^*_{conv} within a constant factor for a secure distributed convolution task, as in a secure distributed matrix multiplication task.

Remark 4. According to Theorem 4, the achievable recovery threshold of secure distributed convolution using polynomial codes is also order-optimal to the number of workers, as in the achievable recovery threshold of secure distributed matrix multiplication in Remark 3.

$$K_{\text{conv, polynomial}} = \max\{m, n\} + m + n = \Theta(1). \tag{25}$$

Theorem 5. *For secure distributed computing of convolution task $\mathbf{c} = \mathbf{a} * \mathbf{b}$ using N workers, where each worker can store $\frac{1}{m}$ fraction of $\mathbf{a}$ and $\frac{1}{n}$ fraction of $\mathbf{b}$ without any information about $\mathbf{a}$ and $\mathbf{b}$, the optimum recovery threshold K^*_{conv} is characterized within a factor of 3, i.e.,*

$$\frac{1}{3} K_{conv,\ polynomial} < K^*_{conv} \leq K_{conv,\ polynomial}, \tag{26}$$

where $K_{conv,\ polynomial} = \max\{m, n\} + m + n$ is the upper bound on the recovery threshold K^*_{conv}, which can be achieved by polynomial codes.

Proof: See Appendix. ∎

VI. Discussion: Overall Runtime Distribution

We dedicate this section to estimate overall runtime distribution to recover the final product at the master. We assume that runtime for encoding and decoding process at the master are deterministic, and we highlight the waiting time of returning sub-products from workers at the master.

A. Encoding and Decoding Complexities

To estimate the encoding and decoding times at the master, we compute the computational complexities for encoding and decoding of our proposed scheme.

1) Encoding Complexity: To encode the sub-matrices $\tilde{A}_i$ for each worker, the master adds $m + 1$ matrices of size $s \times \frac{r}{m}$. In terms of the sub-matrices $\tilde{B}_i$, the master adds $n + 1$ matrices of size $s \times \frac{t}{n}$. Thus, the computational complexity to encode the sub-matrices for each worker is $O\left((m+1)\frac{sr}{m} + (n+1)\frac{st}{n}\right) = O\left(s(r+t)\right)$. The overall encoding complexity for N workers is $O\left(Ns(r+t)\right)$.

2) Decoding Complexity: To decode the final product C, the master solves a polynomial interpolation problem on a $(2m + 2n - 2)$th-degree polynomial for $\frac{rt}{mn}$ elements. Since the polynomial interpolation algorithm has a complexity of $O\left(k \log^2 k\right)$ for a kth-degree polynomial [28], the decoding complexity at the master is $O\left(\frac{rt}{mn}(m+n)\log^2(m+n)\right)$.

B. Waiting Time at Master

We compare waiting time distributions for three cases as follows.

- *Polynomial Codes*: The master divides the sub-tasks to workers by using polynomial codes proposed in Section IV. Workers are assigned a size of $\frac{1}{mn}$ fraction of the whole task to compute $C = A^T B$. Thus, the master needs to wait for the fastest $mn + m + n$ workers.
- *Lower Bound*: We consider the ideal case wherein workers are assigned a size of $\frac{1}{mn}$ fraction of the whole task, and the master needs to wait for the fastest $mn + 1$ workers only, which is a lower bound.
- *Minimum Load*: We consider an ideal secure scheme in terms of computation load at the workers, which is not designed to mitigate the straggling effect. Assigned tasks are encoded to preserve the security of the two inputs from workers. In this case, the master should wait for all the workers to recover the final product. According to the information-theoretic lower bound given in Theorem 2, the minimum computation load at each worker is $\frac{1}{N-1}$ fraction of the whole task. Thus, each worker computes $\frac{1}{N-1}$ fractions of the whole task in this scheme.

We denote the time spent to compute $C = A^T B$ as a random variable T_C. We assume that the computing time distribution $\Pr(T_C \leq t)$ follows an exponential distribution [25], i.e.,

$$\Pr(T_C \leq t) = 1 - e^{-\mu t}, \quad \forall t \geq 0, \tag{27}$$

where exponential rate μ is referred to as straggling parameter. If workers are assigned with L fraction of the whole task, we denote the time spent at each worker as a random variable T_W. Computing time distribution at each worker is

$$\Pr(T_W \leq t) = 1 - e^{-\frac{\mu}{L} t}, \quad \forall t \geq 0, \tag{28}$$

We now calculate expected values of waiting time for three cases. Note that expected value of the kth statistics of n independent random variables with an exponential distribution with rate $\frac{\mu}{L}$ is $\frac{L(H_n - H_{n-k})}{\mu}$, where $H_n = \sum_{i=1}^{n} \frac{1}{i} \simeq \log n$ is the nth harmonic sum [4]. In our scheme using polynomial codes, the master needs to wait for the fastest $mn + m + n$ workers among N workers, each of which is assigned to compute $\frac{1}{mn}$ fraction of the whole computation task $C = A^T B$. The expected waiting time $\mathbb{E}[T_{\text{poly}}]$ of our scheme for the fastest $mn + m + n$ workers among N workers to compute the final

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2846601, IEEE Transactions on Information Forensics and Security

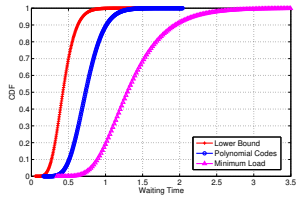

Fig. 5. CDF of the waiting times when $m = 3$ and $n = 3$ with $N = 30$ workers.

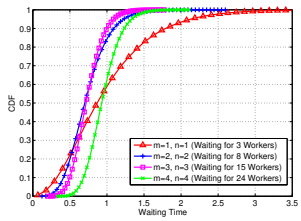

Fig. 6. CDF of the waiting times with $N = 30$ workers.

product at the master is given by

$$\mathbb{E}[T_{\text{poly}}] = \frac{H_N - H_{N-(mn+m+n)}}{\mu mn} \simeq \frac{1}{\mu mn} \log\left(\frac{N}{N-(mn+m+n)}\right). \tag{29}$$

Using the same approach, we estimate expected waiting time $\mathbb{E}[T_{\text{bound}}]$ for the ideal scheme that achieves the lower bound as

$$\mathbb{E}[T_{\text{bound}}] = \frac{H_N - H_{N-(mn+1)}}{\mu mn} \simeq \frac{1}{\mu mn} \log\left(\frac{N}{N-(mn+1)}\right). \tag{30}$$

In addition, we can easily derive expected waiting time $\mathbb{E}[T_{\text{minload}}]$ for the secure scheme with minimum computation load as

$$\mathbb{E}[T_{\text{minload}}] = \frac{H_N}{\mu(N-1)} \simeq \frac{1}{\mu(N-1)} \log N. \tag{31}$$

From the expected waiting times for the scheme using polynomial codes and the secure scheme with minimum computation load, it is not easy to reveal the effect of polynomial codes to reduce waiting time by mitigating the straggling effect (i.e., the coding gain is statistical). Thus, we compare cumulative distribution functions (CDF) for specific problem parameters. We consider that the master divides its task to $N = 30$ workers. In the proposed scheme using polynomial codes and the ideal scheme that achieves lower bound, we assume that each worker is assigned with sub-matrices which have $\frac{1}{3}$ the size of A and B, respectively ($m = 3$, $n = 3$). In the secure scheme with minimum computation load, each worker is assigned with the sub-task which is a $\frac{1}{30-1} = \frac{1}{29}$ fraction of the whole task. Fig. 5 shows the CDFs of waiting times for three schemes. It is observed that the waiting time at the master is reduced by using polynomial codes, that can efficiently mitigate straggler effects than the secure scheme with minimum computation load.

We compare waiting times of the proposed scheme with different sub-task assignments. We consider the four cases of the sizes of sub-tasks, the CDFs of which are given in Fig. 6. If smaller sub-tasks are assigned to each worker, workers can finish their sub-tasks faster, but the master should wait for more workers to finish to recover the final product. Conversely, if the larger sub-tasks are assigned, the master waits for fewer workers to finish, but workers may finish their sub-tasks at a slower pace. Waiting time at the master depends on the straggling parameter μ and the number of workers N according to the size of the sub-tasks. Thus, we can optimize waiting time at the master by considering these system parameters.

VII. Conclusion and Future Directions

In this paper, we investigated a secure distributed computing problem wherein a master wants to hide information about input data from workers. We proposed a secure distributed computing scheme using polynomial codes for matrix multiplication and convolution tasks of two confidential inputs. We demonstrated that our proposed scheme efficiently adresses stragglers by using polynomial codes on the design of sub-tasks assigned to workers. It was further derived that achievable recovery threshold of our proposed scheme is order-optimal to the number of workers. We derived lower bound on the optimal recovery threshold, and by using results we claimed that the optimal recovery threshold can be characterized within a constant multiplicative factor.

There are a few interesting future directions of this work.

1) Sharpening bounds: The upper and lower bounds on the optimal recovery threshold are within a constant factor. Thus, there is a room for further development of these bounds. For example, in our scheme a master can also recover unnecessary products of random matrices as well as products of input matrices, thus better schemes with no redundancy could exist.

2) Weak data security: In the proposed scheme, data security is preserved from workers to exploit suspicious but useful workers for distributed computing. We referred to it as *strong data security*. However, we can imagine a secure distributed computing scheme from the network security perspective, that preserves *weak data security* from an eavesdropper with access to the link between the master and workers, under the assumption that workers are trustworthy.

3) Practical issues on overall runtime: In section VI, we estimate overall runtime distribution with emphasis on waiting time of returning sub-products from workers at the master. We need to further consider several issues to reduce overall runtime to perform secure distributed computing. First, encoding and decoding times at the master should be considered. By considering encoding and decoding complexities of polynomial codes, we can reduce overall runtime with modifications on the proposed scheme. Second, the communication load between the master and workers is a key metric in distributed computing to determine overall runtime. In the proposed scheme, the master allocates the sub-tasks to all possible workers since the recovery threshold does not scale

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2846601, IEEE Transactions on Information Forensics and Security

with the number of workers. However, if the communication load to allocate sub-tasks to workers is considered, there could be another solution to reduce overall runtime.

4) Heterogeneous networks: We can also imagine several heterogeneous networks. For example, workers could have different performance to compute sub-tasks or different memory sizes to store assigned sub-matrices. In addition, the master could require different level of data security to workers: data security should be guarded against some workers, but it is not required for the others.

APPENDIX

Proof of Theorem 2: We assume that the elements of the two input matrices A and B are drawn from i.i.d. random variables. In addition, we would like to recall that either of the two inputs A and B is a tall matrix, i.e., $s \geq r$ or $s \geq t$, to ensure that $C = A^T B$ is a full-rank matrix. These two assumptions are essentially needed for information-theoretic derivations on the considered problem. Otherwise, the considered problem could be degenerate.

Without loss of generality, let us consider that the master can recover the final product C from sub-products of the K^* workers $\{\mathrm{W}_i\}_{i=1}^{K^*}$. It can be represented as

$$H\left(C\middle|\tilde{C}_1,\ldots,\tilde{C}_{K^*}\right)=0. \tag{32}$$

In addition, the mutual information between the final product and each of the sub-products is given by

$$\begin{aligned}
I\left(C;\tilde{C}_i\right) &= I\left(A^TB;\tilde{A}_i^T\tilde{B}_i\right) && (33)\\
&= H\left(A^TB\right)-H\left(A^TB\middle|\tilde{A}_i^T\tilde{B}_i\right) && (34)\\
&\leq H\left(A^TB\right)-H\left(A^TB\middle|\tilde{A}_i^T\tilde{B}_i,\tilde{A}_i,\tilde{B}_i\right) && (35)\\
&= H\left(A^TB\right)-H\left(A^TB\middle|\tilde{A}_i,\tilde{B}_i\right) && (36)\\
&= H\left(A^TB\right)-I\left(A^TB;A,B\middle|\tilde{A}_i,\tilde{B}_i\right) && (37)\\
&\quad -H\left(A^TB\middle|\tilde{A}_i,\tilde{B}_i,A,B\right)\\
&= H\left(A^TB\right)-I\left(A^TB;A,B\middle|\tilde{A}_i,\tilde{B}_i\right) && (38)\\
&= H\left(A^TB\right)-H\left(A,B\middle|\tilde{A}_i,\tilde{B}_i\right) && (39)\\
&\quad +H\left(A,B\middle|\tilde{A}_i,\tilde{B}_i,A^TB\right)\\
&= H\left(A^TB\right)-H\left(A,B\right)+H\left(A,B|A^TB\right) && (40)\\
&= H\left(A^TB\right)-I\left(A^TB;A,B\right) && (41)\\
&= H\left(A^TB|A,B\right) && (42)\\
&= 0 && (43)
\end{aligned}$$

where (35) is due to the fact that conditioning reduces entropy, (36) is due to the fact that $\tilde{A}_i^T\tilde{B}_i$ is a deterministic function of $\tilde{A}_i$ and $\tilde{B}_i$, (38) and (43) are due to the fact that A^TB is a deterministic function of A and B, and (40) follows from (1). According to the definition of the mutual information [29], $I\left(C;\tilde{C}_i\right), \forall i \in [1:K^*]$ is not a negative value. Hence, from (39),

$$I\left(C;\tilde{C}_i\right)=0. \tag{44}$$

Finally, we have

$$\begin{aligned}
H\left(C\right) &= I\left(C;\tilde{C}_1,\ldots,\tilde{C}_{K^*}\right)+H\left(C\middle|\tilde{C}_1,\ldots,\tilde{C}_{K^*}\right) && (45)\\
&= I\left(C;\tilde{C}_1,\ldots,\tilde{C}_{K^*}\right) && (46)\\
&= H\left(\tilde{C}_1,\ldots,\tilde{C}_{K^*}\right)-H\left(\tilde{C}_1,\ldots,\tilde{C}_{K^*}\middle|C\right) && (47)\\
&\leq H\left(\tilde{C}_1,\ldots,\tilde{C}_{K^*}\right)-H\left(\tilde{C}_i\middle|C\right) && (48)\\
&= H\left(\tilde{C}_1,\ldots,\tilde{C}_{K^*}\right)-H\left(\tilde{C}_i\right) && (49)\\
&\leq \left(1-\frac{1}{K^*}\right)H\left(\tilde{C}_1,\ldots,\tilde{C}_{K^*}\right) && (50)\\
&\leq \left(1-\frac{1}{K^*}\right)\sum_{i=1}^{K^*}H\left(\tilde{C}_i\right) && (51)\\
&\leq \frac{1}{mn}(K^*-1)H\left(C\right), && (52)
\end{aligned}$$

for any $i \in [1:K^*]$ where (46) follows from (32), (49) follows from (44), (50) follows from Han's inequality [29], (51) is due to the fact that dropping conditioning does not reduce entropy, and (52) follows from the fact that the size of $\tilde{C}_i, \forall i \in [1:K^*]$ is $\frac{1}{mn}$ times smaller than the size of the final product C. From (52), we have

$$K^* \geq mn+1. \tag{53}$$

This completes the proof. ■

Proof of Theorem 3: We prove Theorem 3 by using the upper and the lower bounds in Theorems 1 and 2.

$$\begin{aligned}
K^* &\geq mn+1 && (54)\\
&\geq \frac{mn+1+m+n}{2} && (55)\\
&> \frac{mn+m+n}{2} && (56)\\
&= \frac{1}{2}K_{\text{polynomial}}. && (57)
\end{aligned}$$

where (54) follows from Theorem 2, (55) is due to the fact that $mn+1 \geq m+n$ when $m \geq 1$ and $n \geq 1$, and (57) follows from Theorem 1. From (57) and Theorem 1, we have

$$\frac{1}{2}K_{\text{polynomial}} < K^* \leq K_{\text{polynomial}}. \tag{58}$$

This completes the proof. ■

Proof of Lemma 1: Without loss of generality, we prove Lemma 1 for $\mathrm{W}_i, i \in [1:N]$ and for the input A. It can be simply generalized to any worker and the input B. From (16),

$$\begin{aligned}
I\left(\tilde{A}_i;A\right) &= H\left(\tilde{A}_i\right)-H\left(\tilde{A}_i\middle|A\right) && (59)\\
&= H\left(\tilde{A}_i\right)-H\left(\tilde{A}_i\middle|A\right)+H\left(\tilde{A}_i\middle|A,R_A\right) && (60)\\
&= H\left(\tilde{A}_i\right)-I\left(R_A;\tilde{A}_i\middle|A\right) && (61)\\
&= H\left(\tilde{A}_i\right)-H\left(R_A\middle|A\right)+H\left(R_A\middle|\tilde{A}_i,A\right) && (62)\\
&= H\left(\tilde{A}_i\right)-H\left(R_A\middle|A\right) && (63)\\
&= H\left(\tilde{A}_i\right)-H\left(R_A\right), && (64)
\end{aligned}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2846601, IEEE Transactions on Information Forensics and Security

where (60) and (63) follow from the fact that $\tilde{A}_i$ is a deterministic fuction of A and R_A. From (64), we can see that $I\left(\tilde{A}_i; A\right) = 0$ if $H\left(\tilde{A}_i\right) = H\left(R_A\right)$. This implies that if R_A has the same size of $\tilde{A}_i$, i.e., the size of R_A is the same as the fraction of A ($A_j, j \in [1:m]$), and R_A is generated in a finite field $\mathbb{F}_q$, then $I\left(\tilde{A}_i; A\right) = 0$. This completes the proof. ∎

Proof of Theorem 5: For a distributed convolution task where each worker can store $\frac{1}{m}$ fraction of $\mathbf{a}$ and $\frac{1}{n}$ fraction of $\mathbf{b}$ without security constraints, lower bound on optimal recovery threshold is $\max\{m, n\}$ [8]. This bound can be also applied to a secure distributed convolution task since the security constraint to workers does not lower the bound. Hence, we have

$$\begin{aligned} K^*_{\text{conv}} &\geq \max\{m, n\} && (65)\\ &= \frac{1}{3}\max\{m, n\} + \frac{2}{3}\max\{m, n\} && (66)\\ &\geq \frac{1}{3}\max\{m, n\} + \frac{2}{3}\frac{m+n}{2} && (67)\\ &= \frac{\max\{m, n\} + m + n}{3} && (68)\\ &= \frac{1}{3}K_{\text{conv, polynomial}}. && (69) \end{aligned}$$

where (67) is due to the fact that the maximum of the two values is not less than the mean of them, and (69) follows from Theorem 4. From (69) and Theorem 4, we have

$$\frac{1}{3}K_{\text{conv, polynomial}} < K^*_{\text{conv}} \leq K_{\text{conv, polynomial}}. \quad (70)$$

This completes the proof. ∎

REFERENCES

[1] J. Dean and L. A. Barroso, "The tail at scale," *Communications of the ACM*, vol. 56, no. 2, pp. 74-80, Feb. 2013.

[2] D. Wang, G. Joshi, and G. Wornell, "Using straggler replication to reduce latency in large-scale parallel computing," *ACM SIGMETRICS Performance Evaluation Review*, vol. 43, no. 3, pp. 7-11, Dec. 2015.

[3] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "A unified coding framework for distributed computing with straggling servers," *in Proc. Global Communications Conference (GLOBECOM) Workshops*, Washington D. C., Dec. 2016.

[4] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ranchandran, "Speeding up distributed machine learning using codes," *in Proc. International Symposium on Information Theory (ISIT)*, Barcelona, Spain, Jul. 2016.

[5] S. Dutta, V. Cadambe, and P. Grover, ""Short-dot": computing large linear transforms distributedly using coded short dot product," *in Proc. 31th Annual Conference on Neural Information Processing Systems (NIPS)*, Barcelona, Spain, Dec. 2016.

[6] K. Lee, C. Suh, and K. Ramchandran, "High-dimensional coded matrix multiplication," *in Proc. International Symposium on Information Theory (ISIT)*, Aachen, Germany, Jun. 2017.

[7] R. Bitar, P. Parag, and S. E. Rouayheb, "Minimizing latency for secure distributed computing," *in Proc. International Symposium on Information Theory (ISIT)*, Aachen, Germany, Jun. 2017.

[8] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Polynomial codes: an optimal design for high-dimensional coded matrix multiplication," *arXiv preprint arXiv: 1705.10464v1*, May 2017.

[9] S. Dutta, V. Cadambe, and P. Grover, "Coded convolution can provide arbitrarily large gains in successfully computing before a deadline," *arXiv preprint arXiv: 1705.03875*, May 2017.

[10] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding," *in Proc. 31th Annual Conference on Neural Information Processing Systems (NIPS) Workshop on Machine Learning Systems*, Barcelona, Spain, Dec. 2016.

[11] W. Halbawi, N. Azizan-Ruhi, F. Salehi, and B. Hassibi, "Improving distributed gradient descent using Reed-Solomon codes," *arXiv preprint arXiv: 1706.05436v1*, Jun. 2017.

[12] W. Du and M. J. Atallah, "Secure multi-party computation problems and their applications: a review and open problem," *in Proc. the 2001 Workshop on New Security Paradigms*, Cloudcroft, NM, Sep. 2001.

[13] R. Cramer, I. Damgard, and U. Maurer, "General secure multi-party computation from any linear secret-sharing scheme," *Advances in Cryptology-EUROCRYPT 2000*, pp. 316-334, May 2000.

[14] H. Takabi, E. Hesamifard, and M. Ghasemi, "Privacy preserving multi-party machine learning with homomorphic encryption," *in Proc. 31th Annual Conference on Neural Information Processing Systems (NIPS) Workshop on Private Multi-Party Machine Learning*, Barcelona, Spain, Dec. 2016.

[15] H. B. Mcmahan, E. Moore, D. Ramage, S. Hampson, and B. A. Arcas, "Communication-efficient learning of deep networks from decentralized data," *in Proc. 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, Fort Lauderdale, FL, Apr. 2017.

[16] V. Smith, C. -K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," *in Proc. 32th Annual Conference on Neural Information Processing Systems (NIPS)*, Long Beach, CA, Dec. 2017.

[17] I. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of Society for Industrial & Applied Mathematics*, vol. 8, no. 2, pp. 300-304, Jun. 1960.

[18] P. Mohassel, "Efficient and secure delegation of linear algebra," *IACR Cryptology ePrint Archive: http://eprint.iacr.org/2011/605.pdf*, 2011.

[19] Y. Zhang and M. Blanton, "Efficient secure and verifiable outsourcing of matrix multiplications," *in Proc. International Conference on Information Security*, pp. 158-178, 2011.

[20] A. Castiglione, P. D. Arco, A. D. Santis, and R. Russo, "Secure group communication schemes for dynamic heterogeneous distributed computing," *Future Generation Computer Systems*, vol. 74, pp. 313-324, Sep. 2017.

[21] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, Nov. 1979.

[22] H. Yamamoto, "Secret sharing system using (k, L, n) threshold scheme," *Electronics and Communications in Japan*, vol. 69, no. 9, Sep. 1986.

[23] R. Bitar, P. Parag, and S. E. Rouayheb, "Minimizing latency for secure distributed computing using secret sharing via staircase codes," *arXiv preprint arXiv: 1802.02640*, Feb. 2018.

[24] M. J. Atallah and K .B. Frikken, "Securely outsourcing linear algebra computations," *in Proc. 5th ACM Symposium on Information, Computer and Communications Security*, New York, NY, 2010.

[25] G. Liang and U. C. Kozat, "TOFEC: achieving optimal throughput-delay trade-off of cloud storage using erasure codes," *in Proc. IEEE Conference on Computer Communications (INFOCOM)*, Toronto, Canada, Apr. 2014.

[26] V. Guruswami and M. Sudan, "Improved decoding of Reed-Solomon and algebraic geometric codes," *in Proc. 39th Annual Symposium on Foundations of Computer Science*, Palo Alto, CA, Nov. 1998.

[27] F. Didier, "Efficient erasure decoding of Reed-Solomon codes," *arXiv preprint arXiv: 0901.1886*, Jan. 2009.

[28] H. T. Kung, *Fast evaluation and interpolation*, Technical Report, Carnegie Mellon University, 1973.

[29] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd edtion, NJ: Wiley, 2006.