Secure Coded Computation for Efficient Distributed Learning in Mobile IoT

Yilin Yang*, Rafael G.L. D'Oliveira†, Salim El Rouayheb*, Xin Yang*, Hulya Seferoglu‡, Yingying Chen*
*Rutgers University, USA. †Massachusetts Institute of Technology, USA. ‡University of Illinois at Chicago, USA.
*{yy450, sye8, xin.yang, yingche}@scarletmail.rutgers.edu, †rafaeld@mit.edu, ‡hulya@uic.edu

***Abstract*—Distributed computation plays an essential role in cloud and edge computing. Data such as images, audio, and text can be represented as matrices to facilitate efficient computation, especially in the domains of distributed machine learning, computer vision, and signal processing. Many coded computation algorithms have been proposed for big data applications to securely partition and distribute matrices to parallel worker devices. However, these proposals have yet to be adapted for mobile platforms beyond theoretical means. Mobile IoT networks can greatly benefit from secure distributed computing, however, commercial devices such as smartphones and tablets are much more limited in resources compared to platforms in data centers, requiring special design considerations. We investigate existing distribution schemes from an operational complexity and security viewpoint and study their performance in several mobile IoT networks, identifying performance bottlenecks in regards to communication and computation costs. From our findings, we propose new, scalable algorithms optimized to handle the unique constraints of mobile IoT. Extensive evaluations of our proposals on publicly available image classification datasets show how distributed learning can be specially optimized to enhance runtime and battery performance on mobile IoT by over $10\times$.**

***Index Terms*—Distributed computing, coded computations, edge device, mobile IoT**

I. Introduction

Distributed learning plays an essential role in advanced edge computing, particularly in areas such as image processing [1]–[3], healthcare monitoring [4], [5], smart cities [6], [7], and vehicular networks [8]–[10]. Recently, there has been a growing interest in applying coding-theoretic techniques to speed-up distributed computing algorithms and increase their resiliency and security in what is now referred to as *coded computation* [11]–[14]. These studies were able to produce fast and secure algorithms for distributed computing and have already had theoretical performance analysis and practical demonstrations, mainly in controlled settings featuring large-scale clusters, data centers, and cloud computing facilities. However, adapting such schemes to mobile IoT devices, such as smartphones, wearable devices, home and personal assistants, smart cars, and security systems, has yet to be fully explored and validated beyond simulations [15]–[17].

Deploying existing distributed computation algorithms on mobile IoT is non-trivial and presents many practical challenges [18]. For instance, mobile IoT devices are designed to be small, portable, and convenient, meaning they typically lack high-end computing resources available to most stationary, continuously-powered facilities. Moreover, security poses a more pronounced challenge given the open environment of mobile networks and the vulnerability to threats such as denial-of-service (DOS), eavesdropping, and man-in-the-middle attacks [19], [20]. As shown in Figure 1, the incentive to overcome these challenges is very high, as many applications scenarios can benefit from efficient distributed computing algorithms, including GPS tracking, local traffic prediction, mobile gaming, and crowd sensing. As an answer to the challenges of limited computational resources, battery life, and data privacy, we perform the first study of secure coded computation for distributed learning in mobile IoT using real world commercial hardware.

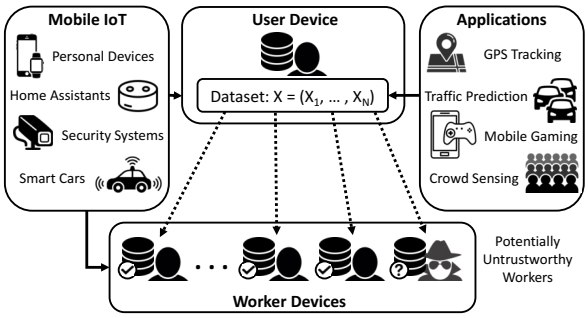


Figure 1. Use case scenarios for secure distributed learning in the mobile IoT, where user data is securely distributed to various IoT devices for computation.

To achieve effective distributed computation algorithms on mobile IoT, we aim to develop simple and lightweight secure distributed algorithms. We envision that cryptographic solutions, especially fully homomorphic encryption, can be computationally heavy on mobile IoT devices and quickly exhaust the very limited battery capacity [21]. To address this challenge, we propose efficient algorithms that ensure information-theoretic security to protect outsourced data and parameters from passive eavesdropping attacks on leveraged devices. Many distributed computation tasks are based on statistical and learning-based algorithms (e.g., logistic regression), which can be implemented through a series of matrix multiplication operations. We specifically focus on the scope of secure distributed matrix multiplication (SDMM), which is an ideal candidate for optimization due to its intensive computational complexity. Existing studies that analyze performance of SDMM algorithms are mostly performed through theoretical analysis or on cloud platforms, such as Amazon EC2 [12],

978-1-6654-4108-7/21/$31.00

[22]. However, the heterogeneous mobile IoT networks are significantly different from cloud facilities in terms of hardware, computational capability, and power constraints. Recent works implemented coded computation on edge devices [14] but are evaluated as simulations that do not fully account for real-world challenges, such as communication overhead and battery consumption. To demonstrate this issue, we conduct preliminary studies on existing SDMM algorithms [12], [13] in real-world mobile IoT environments. We observe that these algorithms can be highly taxing on resources such as CPU, memory, and battery that must be shared with other everyday mobile applications. Notably, we find that insertion of more resources (i.e., additional devices) into the mobile IoT network does not necessarily improve computational performance. To this end, we propose lightweight secure coded distributed learning algorithms specifically optimized for high energy efficiency and low computational complexity.

We focus on matrix multiplication as the bottleneck of many of the now well-celebrated deep learning algorithms [23], [24]. We consider the practical condition that an IoT device must find the product of matrices as part of a bigger learning algorithm, but may be unable or unwilling to perform the computation on the local device due to limited resources. The local device may, however, have access to N number of associate devices within the mobile IoT network, which can share their own resources to assist with the learning task. We propose a novel algorithmic approach to coordinate the partitioning, encoding, distribution, and decoding of the matrices between the local device (i.e. user) and associate devices (i.e. workers) such that information-theoretic security is preserved without sacrificing battery or low latency. Specifically, we consider information-theoretic security based on Shamir's secret sharing theory [25] by dividing the encoded data into multiple parts before sharing. To reconstruct the original data will require knowledge of all encoded parts. We develop a prototype implementation that can perform secure distributed learning, using logistic regression as an example, on heterogeneous mobile devices and conduct extensive evaluations under real-world conditions. Moreover, our approach is scalable in order to support dynamically changing IoT networks, where devices may be added or removed freely, as well as adaptable to support more complex learning problems beyond linear regression. Mobile applications leveraging computer vision (e.g. geotagging, gaming, etc.) can greatly benefit from our prototype, as images are easily convertible into 2D pixel-based matrix representations. As such, we evaluate the performance of the proposed algorithms using publicly available EMNIST [26] and CIFAR [27] image datasets, providing a proof-of-concept study on the feasibility of distributed computing algorithms on commercially accessible hardware in real-world settings. Our prototype is lightweight (i.e. designed with minimal memory footprint) and has been optimized to accelerate the computation speed of classification tasks. While these optimizations were designed to address mobile IoT constraints, our algorithms can also be deployed in traditional data centers to provide similar benefits. We view this work as the first step to bringing secure coded computation algorithms to mobile IoT devices and application scenarios. Our contributions are summarized as follows:

- We propose a secure coded distributed computation framework for mobile IoT with guaranteed information-theoretic security, which can facilitate reliable and secure distributed computing on mobile IoT devices.
- Compared to existing works that consider distributed computations for big data centers or evaluate distributed computing algorithms for edge devices via simulations, we focus on SDMM and secure coded computations in real-world, heterogeneous mobile IoT environments.
- We identify performance bottlenecks in communication and computation operations when performing large scale classification tasks on mobile IoT and propose *lightweight codes* for SDMM algorithms optimized towards high energy efficiency and low computational complexity.
- We develop a secure coded distributed learning prototype for Android platforms deployable on commercial mobile IoT devices to solve practical classification problems through secure distributed logistic regression.
- Extensive evaluations of our algorithms on binary image classification tasks using multiple public datasets and mobile devices show average runtime speed up by $13\times$ and average battery consumption reduced by $10\times$.

II. Related Works

Distributed learning is connected to several closely related areas, including federated learning, collaborative learning, and secure-multiparty computation, all with vested interests in preserving data privacy and minimizing communication overhead [28]. Many algorithms draw inspiration from strategies such as Shamir's secret sharing [25]. Secure partitioning, distribution, and reconstruction of matrix information has been applied to many areas such as visual cryptography [29], [30], audio sharing [31], and electronic voting [32]. Major contributions often include optimizations in data compression, faster performance, or stronger data protection. Ensuring privacy of distributed information is a primary concern for many systems, leading to many investigations on how secrecy can be improved, such as by addressing straggler mitigation [33], defending colluding workers [34], and information-theoretic security [35].

Information-theoretic security protects data such that adversaries cannot brute force the solution, even with hypothetically unlimited computing power. Some examples of SDMM algorithms incorporating information-theoretic security include [12], [13], [31], [36]. Previous evaluations of SDMM have considered properties such as the uplink/downlink trade-off [22], download rates [37], and communication and computation times [38]. Operation complexity has also been considered [39], [40], but not for information-theoretic secure systems.

While thorough in many regards, existing literature omits consideration for hardware constraints and implementation challenges. Secure distribution schemes for mobile platforms have been examined from a theoretic standpoint only [15]. Studies with real-world deployments typically feature data

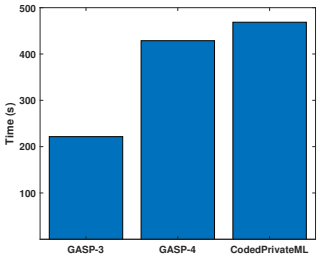


Figure 2. Runtime of existing distribution schemes using EMNIST.

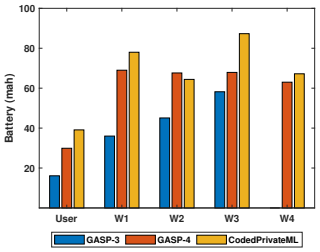


Figure 3. Battery expended for existing distribution schemes using EMNIST.

clusters or commercial services such as Amazon EC2 [12], [22], which merely offloads hardware burdens to third parties. Our work is the first to study performance of SDMM algorithms on commercial hardware with real-world data.

III. Preliminary Study

We first conduct preliminary studies to investigate the feasibility of deploying existing secure distributed learning algorithms on mobile IoT devices. Specifically, we identify implementation challenges and verify existing algorithm performance using mobile platforms. We implement two existing SDMM algorithms that are initially designed for cloud facilities on smart IoT devices: GASP [13] and CodedPrivateML [12]. We consider an example mobile IoT network consisting of five smart devices, with one device serving as the user and the rest as workers. Details on hardware are provided in Section VII. GASP is supports collaboration with three or four workers whereas CodedPrivateML requires a minimum of four workers. Therefore, we evaluate both the 3-worker and 4-worker schemes for GASP to compare with the 4-worker scheme of CodedPrivateML. On top of the existing SDMM schemes, we developed a logistic regression-based learning model for Android platforms to examine the performance of matrix computation towards practical distributed learning use cases. We employ the publicly availble EMNIST [26] dataset for secure distributed image classification, using 1200 training images and 2000 testing images at a $28 \times 28px$ resolution.

The computational bottleneck lies in the matrix multiplication, particularly the product of the matrix with its own transpose. We denote the matrix as $W \in \mathbb{F}_q^{r \times s}$ in the finite field $\mathbb{F}_q$ with a size of $r \times s$. The multiplication of $W^\intercal W$ exhibits complexity of $\mathcal{O}(rs^2)$. We measure average elapsed runtime and battery consumption for 10 trials of logistic regression-based binary image classification to quantify existing SDMM algorithm performance, comparing GASP and CodedPrivateML in Figures 2 and 3 as an example. For a typical binary image classification task, the average runtime necessary to reach at least 95% classification accuracy using GASP-based logistic regression takes more than $200s$ with the help of 3 workers (i.e., $W1$, $W2$, $W3$). Interestingly, we find that adding a fourth worker $W4$ (i.e., adding resources to the IoT network) more than doubles runtime to over $400s$. Our studies using CodedPrivateML and battery consumption produce similar trends. These results suggest that coordinating

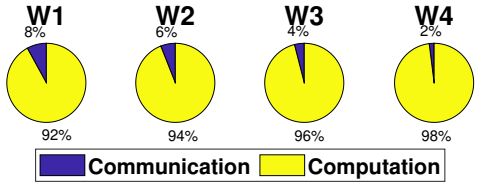


Figure 4. Average runtime breakdown by communication time and computation time for four worker devices using existing algorithms.

multiple actors in the IoT network is especially expensive, undermining computational optimizations by prior literature.

To further understand which components of distributed computing stand to benefit the most from optimization, we examine runtime more closely, differentiating between time spent on communication (i.e. transmitting or receiving data) and computation (i.e. calculating numerical operations). Figure 4 shows our initial findings, revealing that the overwhelming majority of runtime, over 90% for all workers and all algorithms, is spent on computation, illustrating a need for more effective distribution and lower complexity operations

Our preliminary study identifies key limitations for deploying existing SDMM algorithms on IoT devices: 1) The CPUs of smart IoT devices are optimized for energy-efficiency and lightweight computing tasks, due to the restricted battery capacity. Thus traditional SDMM algorithms, which can be computationally expensive and power-intensive, are difficult to deploy on mobile devices. 2) Heterogeneity within the mobile IoT means different participating mobile devices have varying computational capabilities. Therefore, a straggler device that has the lowest performance could drastically slow down the overall efficiency of the secure distributed learning algorithm. To address these challenges, we propose optimized secure distributed learning algorithms for mobile IoT in Section V.

IV. Challenges and System Overview

A. Challenges

Preservation of data privacy in untrustworthy networks. Mobile IoT networks are especially vulnerable to various malicious attacks (e.g., DOS, eavesdropping, etc.) due to emphasis on convenience and ease of access. Therefore, data encoding and partitioning algorithms must be carefully designed to prevent data leakage when outsourcing to untrustworthy devices.

Optimizing matrix multiplication energy efficiency. Matrix multiplication is an intensive computational task for large dimensions. Deploying SDMM algorithms on mobile IoT devices is especially challenging because of constrained resources, such as CPU, RAM, and battery. Therefore, adaptations must consider time complexity of operations in the context of energy efficiency and latency on mobile devices.

Scalability for heterogeneous mobile IoT devices. Devices connected to the mobile IoT network are often frequently changing. Thus, the proposed algorithms should be generalizable to support different numbers of connected worker devices with various computational capabilities.

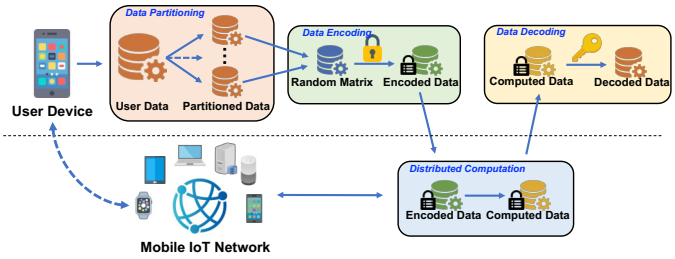


Figure 5. Interaction between the user and worker devices in the mobile IoT network using our proposed secure coded distributed learning framework.

B. System Overview

The proposed secure distributed learning framework comprises four phases: (1) data partitioning, (2) data encoding, (3) distributed computing, and (4) data decoding. We illustrate an overview of the proposal in Figure 5. We consider a typical distributed learning task based on logistic regression, where the privacy for both outsourced dataset and parameters (i.e., gradient vector) should be protected. We first perform stochastic data quantization on the user device following the method described in [12] to convert any given information in the real domain $\mathbb{R}$ into an independent and uniform finite field $\mathbb{F}_q$, where q is a sufficiently large number to prevent wraparound computation.

During data partitioning, the user first divides the data into multiple parts based on the number of available worker devices in the connected mobile IoT network. In data encoding, the user generates a random matrix with all entries subject to $\mathbb{F}_q$. The random matrix is used to mask the partitioned data in order to preserve data privacy. The masked data partitions will then be transmitted to a group of N workers following the SDMM algorithms introduced in Section V. We note that the design of our secure encoding algorithms guarantees that for any individual worker, deriving the original data is impossible on a information-theoretic security basis

During distributed computation, the worker device performs matrix multiplication on the received data partitions using its own local resources before transmitting back to the user. Once sufficient responses from the worker devices are received, the user reconstructs the computation results in the data decoding phase, detailed in Section V. By employing the gradient descent method as in [12], the decoding process reveals the computed gradients on the weights. Iterative weight updates via efficient distribution of matrix multiplication-based gradient computation enables faster and less energy-demanding distributed logistic regression.

Additionally, the proposed secure coded distributed learning framework should be able to accommodate heterogeneous mobile IoT devices varying significantly in computational capabilities (i.e., CPU clock frequency, number of CPU cores, RAM size, RAM frequency, and even software-level battery optimization algorithms). To tackle this issue, we design variants of the SDMM algorithms based on our observations on real-world experimental results. The devised algorithms are specifically optimized for mobile IoT devices to flexibly coordinate with different numbers of workers.

V. Matrix Encoding and Partitioning

We present our proposed SDMM coding scheme to be used within the distributed logistic regression algorithm. In order to update the learning model for each iteration, the algorithm must solve the following computationally intensive problem; given a data matrix $X \in \mathbb{F}_q^{r\times s}$ and a model vector $\omega_t \in \mathbb{F}_q^{s\times 1}$ at iteration t, distributively compute the model update

$$\omega_{t+1} = X^{\mathsf{T}} X \omega_t, \tag{1}$$

without leaking any information about X or ω_t to any of the workers. Here, X^{T} denotes the transpose of the matrix X.

In principle, this can be performed by any of the many known secure distributed matrix multiplication schemes. However, our objective is to design coding schemes with fast and lightweight encoding and decoding to better suit mobile IoT applications. We first present our secure lightweight codes for SDMM and then describe the secure model update to be used within the iterative learning algorithm.

A. Lightweight Codes for SDMM

To distribute the update model of Eq. (1), we propose new lightweight codes for SDMM that securely distribute the two-matrix product AB to a number of workers. We assume that the user has two matrices $A \in \mathbb{F}_q^{r\times s}$ and $B \in \mathbb{F}_q^{s\times t}$ and wishes to compute their product $AB \in \mathbb{F}_q^{r\times t}$ with the assistance of N non-colluding workers without leaking any information about A or B. This is typically solved in literature by partitioning the matrices and using a polynomial code to protect the information. However, the resulting encoding and decoding processes involves dense linear combinations, which can be taxing on computing resources. Since our end goal is to apply this to matrix-vector multiplication, we consider the following inner product partitioning.

$$\begin{pmatrix} A_1 & \dots & A_L \end{pmatrix} \begin{pmatrix} B_1 \\ \vdots \\ B_L \end{pmatrix} = A_1B_1 + \dots + A_LB_L. \tag{2}$$

We propose Algorithm 1 as a general code for SDMM on $N = 2L$ workers. The main idea is to mix the data parts with noise and to divide the workers into L pairs, where each pair is responsible for computing the term A_iB_i in (2). Algorithm 1 is lightweight as both encoding and decoding phases are sparse and each coded sub-matrix involves a single matrix addition.

B. Updating the Model

We now address the problem of updating the model via the computation $\omega_{t+1} = X^{\mathsf{T}} X \omega_t$. The model must be updated in each iteration t. However, data is assigned only once at the beginning of the learning algorithm does not change. Algorithm 2 describes our secure data assignment scheme. Algorithm 3 gives how the model is updated at each iteration t via $\omega_{t+1} = X^{\mathsf{T}} X \omega_t$. At a high level, it does this by implementing Algorithm 1 twice: once to compute $D = X\omega_t$, and then to compute $\omega_{t+1} = X^{\mathsf{T}} D$. Note that the order in which we multiply $\omega_{t+1} = X^{\mathsf{T}} X \omega_t$ matters. Indeed,

Algorithm 1 Lightweight SDMM Code

Input: $A \in \mathbb{F}_q^{r\times s}$, $B \in \mathbb{F}_q^{s\times t}$ and $L \in \mathbb{N}$.
Output: $AB \in \mathbb{F}_q^{r\times t}$.

1: **Random matrix generation:** The user generates two random matrices $R \in \mathbb{F}_q^{r\times \frac{s}{L}}$ and $S \in \mathbb{F}_q^{\frac{s}{L}\times t}$.
2: **Matrix Partitioning:** The user partitions A, B, R, and S into L parts each as in (2).
3: **User Encoding:** For $i = 1, \ldots, L-1$, the user computes:
$W_{2i-1} = A_i + R, \quad V_{2i-1} = B_i + S$
$W_{2i} = A_i - R, \quad V_{2i} = B_i - S$
$W_{2L-1} = A_L + R, \quad V_{2L-1} = B_L + (L-1)S$
$W_{2L} = A_L - R, \quad V_{2L} = B_L - (L-1)S.$
4: **User Upload:** User sends W_i and V_i to each Worker i.
5: **Worker Computation:** Each Worker i computes: $Y_i = \frac{1}{2} W_i V_i$.
6: **User Download:** User downloads Y_i from Worker i.
7: **User Decoding:** User decodes: $AB = \sum_{i=1}^{2L} Y_i$.

performing $(X^\mathsf{T}X)\omega^{(t)}$ takes $\mathcal{O}(rs^2)$ operations as opposed to $X^\mathsf{T}(X\omega^{(t)})$ which takes $\mathcal{O}(rs)$ operations. Our assumption on the attack model is that workers do not collude. Therefore, perfect privacy is ensured in Algorithms 1, 2, and 3 since all the data received by the workers is mixed with uniformly distributed random noise having the same entropy as the data.

Algorithm 2 Secure Data Assignment

Input: Data matrix $X \in \mathbb{F}_q^{r\times s}$ and number of workers $N = 2L$.
Output: For $i = 1, \ldots, N$, Worker i stores coded versions of X and X^T.

1: **Random matrix generation:** User generates a random matrix $R \in \mathbb{F}_q^{r\times \frac{s}{L}}$.
2: **Matrix Partitioning:** User partitions X into L columns.
3: **User Encoding:** For $i = 1, \ldots, L$, the user computes: $W_{2i-1} = X_i + R$, $W_{2i} = X_i - R$.
4: **User Upload:** The user sends W_i to each Worker i.
5: **Worker Storage:** Each Worker i stores W_i and W_i^T.

Security Model: We assume the workers are honest but curious, in that they perform all operations correctly, but store all the information and try to decode the data. We also assume that no two workers collude. Under this assumption, we guarantee information-theoretic security, i.e. the scheme leaks no information, even statistical, about the data, even if the adversary has unlimited computing power.

Our proposal is provably secure because data is padded by a uniformly random key before being sent to a worker. In Algorithm 1, for example, Worker 1 receives the matrices $W_1 = A_1 + R$ and $V_1 = B_1 + S$. Then, $H(A, B|W_1, V_1) = H(A, B)$, since W_1 and V_1 are uniformly random and independent from A and B. A formal proof can be found in Eq. 26 of [41].

Quantization: In order to guarantee information-theoretic privacy, it is necessary that the random keys being used to pad the information be uniformly distributed. As their is no uniform distribution over the real numbers $\mathbb{R}$, we quantize our data to elements of some finite field $\mathbb{F}_q$. We can then obtain our random keys by uniformly sampling over $\mathbb{F}_q$. To avoid biases, we perform stochastic quantization similar to existing techniques such as in [12].

Algorithm 3 Secure Model Update

Input: Model $\omega_t \in \mathbb{F}_q^{s\times 1}$, number of workers $N = 2L$, data matrix X assigned to workers according to Alg. 2
Output: $\omega_{t+1} = X^\mathsf{T}X\omega_t$.

1: **Random matrix generation:** The user generates a random vector $s_1 \in \mathbb{F}_q^{\frac{s}{L}\times 1}$.
2: **Vector Partitioning:** The user partitions ω_t into L rows.
3: **User Encoding:** For $i = 1, \ldots, L-1$, the user computes:
$b_{2i-1} = \omega_{t,2i-1} + s_1,$
$b_{2i} = \omega_{t,2i} - s_1,$
$b_{2L-1} = \omega_{t,2L-1} + (L-1)s_1,$
$b_{2L-1} = \omega_{t,2L} - (L-1)s_1.$
4: **User Upload:** The user sends b_i to each Worker i.
5: **Worker Computation:** Each Worker i computes: $y_i = \frac{1}{2} W_i b_i$.
6: **User Download:** User downloads y_i from each Worker i.
7: **User Decoding:** User decodes: $y = \sum_{i=1}^{2L} y_i = X\omega_t$.
8: **Random matrix generation:** User generates a random vector $s_2 \in \mathbb{F}_q^{r\times 1}$.
9: **User Encoding:** For $i = 1, \ldots, L$, the user computes: $c_{2i-1} = y + s_2$, $c_{2i} = y - s_2$.
10: **User Upload:** User sends c_i to each Worker i.
11: **Worker Computation:** Each Worker i computes: $z_i = \frac{1}{2} W_i^\mathsf{T} c_i$.
12: **User Download:** User downloads z_i from each Worker i.
13: **User Computation (can be precomputed locally, partitioned and uploaded to workers, or outsourced to an extra $N+1th$ worker):** The user computes: $R^\mathsf{T}s_2$.
14: **User Decoding:** The user decodes: $(X^\mathsf{T}X\omega_t)_i = z_{2i-1} + z_{2i} - R^\mathsf{T}s_2$.

VI. Secure Distributed Learning on Mobile IoT

We introduce the implementation of our proposed algorithms of Section V, designed to maintain the accuracy of existing distributed learning algorithms while simultaneously minimizing time spent on computing and communicating in the IoT network.

A. Android Implementation Framework

We develop software prototypes of our proposed algorithms for Android and perform evaluations using commercial smartphones and tablets as example target devices. Particularly, we develop two Android applications for user and worker devices using Java with JDK version 11.0.3 based on the Android Studio 4.0 platform. We use Android SDK version 28, with backward compatibility support for version 17 and later.

The communication model between the user and workers follows the standard server-client model, where a mobile server is established on the user device, with multiple worker

devices acting as clients that download encoded data, perform computations, then transmit back the computed results. Our design constructs the communication module between the user and worker devices through Java Socket API. The user device creates a server socket, listening on a designated port to connect with eligible devices. For simplicity, we assume workers are aware of the IP address and communication port of the user device. As a result, each worker device holds a dedicated communication link with the user. This mechanism is especially important in the rapidly changing mobile IoT network environment, as it allows the user to flexibly add or drop new connections. Hotspot-based communication, as one of the most common real-world mobile IoT network environments, is utilized in our framework, however our implementation can be easily extended to other systems, such as peer-to-peer.

B. Android Implementation for User Device

We first load the dataset from the user device's local flash storage onto RAM using the RandomAccessFile API. In order to quantize the data matrix $\hat{X}$ into the finite field $\mathbb{F}_q$, we require a sufficiently large prime divisor to ensure stochastic distribution. In particular, we leverage the quantization algorithm as in [12], where the largest 24-bit prime number $q = 15485863$ is determined to be a suitable quantization threshold to avoid wraparound computation. Therefore, for any given data point $\hat{X}_{i,j}$ in the real domain, i.e., $\hat{X}_{i,j} \in \mathbb{R}$, where i stands for the row in the matrix and j is the column, we quantize $\hat{X}_{i,j}$ using the following rule:

$$X_{i,j} = \begin{cases} \lfloor 2^l \hat{X}_{i,j} \rfloor \mod q, \hat{X}_{i,j} - \lfloor 2^l \hat{X}_{i,j} \rfloor < 0.5, \\ \lfloor 2^l \hat{X}_{i,j} + 1 \rfloor \mod q, \hat{X}_{i,j} - \lfloor 2^l \hat{X}_{i,j} \rfloor \geq 0.5, \end{cases} \tag{3}$$

where $X_{i,j} \in \mathbb{F}_q$ is the quantized data, $l = 2$ is empirically selected to control quantization level. Similarly, we quantize the initial weight vector $\hat{w} \in \mathbb{R}$ into the finite field $w \in \mathbb{F}_q$.

Next, the user device partitions the quantized dataset X and weight vector w into $X = [X_1, X_2, ..., X_i]$, $w^\mathsf{T} = [w_1, w_2, ..., w_i]$, where i is the number of workers. The user then generates random matrix $R_i \in \mathbb{F}_q$ and $s_i \in \mathbb{F}_q$ to encode the partitioned data $X_i \in \mathbb{F}_q$ and weight vector $w_i \in \mathbb{F}_q$. The naive approach is to sequentially encode and upload data for each worker one at a time, however this leaves workers unnecessarily idle. To minimize wait time, we parallelize the data encoding and distribution process through multithreading. Specifically, we create a thread pool of i threads at the user device using the Java ExecutorService API, where each thread prepares data for the Nth worker to enable simultaneous data preparation. Workers must then solve the model update challenge posed in of Eq. 1 through logistic regression. Algorithm 3 optimizes this computation such that partitions cannot be used to deduce the pre-partitioned dataset, reducing time complexity and guaranteeing user privacy. After downloading results from the workers, the user performs decoding to restore the gradient vector to optimize the logistic regression model through gradient descent as in [12].

C. Android Implementation for Worker Device

Unlike the user device implementation, which is knowledgeable of all information shared in the mobile IoT network, worker devices must operate in relative isolation. Thereby, any sensitive information from the user side, such as the quantization method, encoding schemes, or number of participating devices, will be hidden from the workers to protect data privacy. To participate in a distributed learning task, eligible worker devices can connect to the user's socket server based on IP address and port number. After receiving the encoded data partitions through byte streams, the worker device can then compute the matrix operations using our schemes introduced in Section V. The computation results will be sent back to the user through the maintained socket connection for decoding.

VII. Evaluation

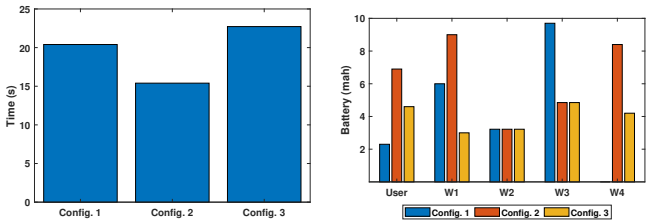
We extend our preliminary study from Section III using binary image classification for a conventional demonstration of distributed machine learning and for its relevance to daily mobile device usage (e.g., mobile gaming, tagging photos).

A. Experimental Setup

Hardware Specifications. Our primary mobile IoT configuration consists of commercially available devices to act as our user (Nexus 5) and workers (Galaxy Note 5, Nexus 6, Lenovo Tab 4 8, and Galaxy Tab A). Other configurations were also studied, including homogeneous networks (i.e. using duplicate models), elaborated on in Section VIII. For our experiments, no special considerations are made when assigning roles to different devices. While the theoretical optimal solution is to assign the most powerful devices to worker roles, device specifications are not necessarily predictable by the user.

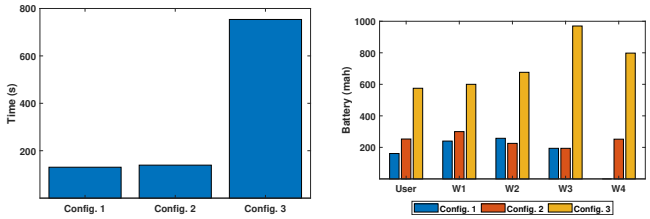
Algorithm Configurations. Several parameters of our proposed algorithms, particularly Algorithm 3, are freely adjustable and may have considerable impact on the overall classification performance. Thus, we study three specific configurations to provide a more comprehensive analysis. In *Configuration 1*, we consider the case of $L = 1$, dividing the data and assigning it to workers according to Algorithm 1 with N=2. We enlist a third worker to handle the computation of the random matrix product RS, making this configuration a 3-worker scheme with relatively asymmetric worker responsibilities. For *Configuration 2* and *Configuration 3*, we consider $L = 2$ utilizing four workers and handle the random matrix product via precomputation or via partition and distribution, respectively.

Data Collection. Similar to our preliminary study, we leverage publicly available datasets to evaluate the performance of our algorithms. In addition to the previously mentioned EMNIST dataset, we also leverage the CIFAR dataset, utilizing 10000 images for training and 2000 for testing. Image resolution is $32 \times 32px$, making the CIFAR dataset significantly larger in both scale and sample size than the EMNIST dataset. We conduct 10 trials of the classification task for Configurations 1-3. In total, our evaluation contains over 60 real-world trials.



(a) Average runtime. (b) Average battery consumption.

Figure 6. Performance using the EMNIST dataset.



(a) Average runtime. (b) Average battery consumption.

Figure 7. Performance using the CIFAR dataset.

Evaluation Metrics. We measure the efficacy of our proposal by considering classification runtime and battery consumption in the mobile IoT. Specifically, we define our evaluation metrics as: 1) *runtime*, the time spent on one classification trial, including both the training and testing stages; 2) *communication time*, the overhead or indirect amount of time needed to coordinate transmissions between devices; 3) *computation time*, the amount of time spent on performing matrix operations for the classification output; 4) *battery consumption*, the milliamp hours expended for the trial. Runtime, communication time, and computation time are recorded by timestamps as devices progress through the classification task. Battery consumption is calculated from the percentage battery loss during execution multiplied by the total battery capacity of the device. To reduce other sources of battery drainage, we disable non-essential applications during the experiment.

B. Experimental Results

Overall Performance. The average runtime and battery consumption for our EMNIST trials are provided in Figures 6(a) and 6(b), respectively. In contrast to the performance of existing algorithms observed in Figure 2, where runtime ranged from 200 to 500 seconds, we find that our proposed algorithms are capable of completing the same task within 15 to 25 seconds, showing speedup over $13\times$. A proportional effect on battery consumption is also observed, showing an average $10\times$ reduction in milliamp hours expended. We observe an average classification accuracy of 95%, however *we note that accuracy does not quantify efficiency in this study.*

Impact of problem size. We find the performance enhancements are more apparent when evaluating on the larger CIFAR dataset. As part of our preliminary experiments, we also evaluated performance of the existing CodedPrivateML and GASP algorithms with the CIFAR dataset, however, we found that even a single classification trial could exceed 90 minutes and deplete over 10% of a given device's battery, making collection of a sample size equivalent to our EMNIST studies significantly more challenging. We believe this illustrates a compelling need for distributed learning algorithms optimized for mobile hardware. Results for the CIFAR dataset, shown in Figure 7, suggest that our proposed algorithms are able to reduce runtime down to under 12 minutes using Configuration 3, or even under 3 minutes in the case of Configuration 1 and 2, equivalent to a speedup of $7\times$ and $30\times$, respectively. The drastic difference in performance observed for Configuration 3 relative to Configuration 1 or 2 is likely due to the absence of precomputation, elaborated further in Section VIII. The relative battery consumption for each individual device is unchanged (i.e. worker 3 consistently consumes the most, user device consumes the least), suggesting that energy costs are fixed to hardware specifications and scale based on problem size. Battery costs for all devices are more stable for the CIFAR dataset as computational resources are under steady demand over a longer duration.

Impact of number of workers. When using Configuration 2 or 3, the user enlists the computing capabilities of 4 mobile devices as workers, whereas Configuration 1 uses only 3. We find 3-worker schemes to require less battery consumption, particularly for the user device. This can be partially attributed to the user device overseeing a smaller IoT network, and thus wasting less energy due to overhead. Leveraging the computational resources of a 4th device incurs a minimum fixed battery cost, regardless of problem size, meaning a user interested in conserving power for all devices, not just the local device, may prefer using 3 workers. However, a smaller network may also be more susceptible to outside threats as every worker has a comparatively large partition of the original data, which adversaries may wish to interfere with through eavesdropping, stalling, etc. We discuss robustness to these factors in Section VIII. Furthermore, it is possible for 4-worker schemes to match or surpass 3-worker schemes in terms of runtime via precomputation.

Impact of precomputation. As described in Section V, security is provided in part by the computation of two random matrices, R and S, which functions effectively as a key. Because the product of RS does not depend on any input data (and vice versa), it is not mandatory for workers to receive this information, thus the user is free to compute this locally. Precomputation contributes to considerable improvements in performance for large problem sizes, based on our evaluation of the CIFAR dataset in Figure 7. Configuration 2 with precomputation is over $5\times$ faster and exhausts half as much battery compared to Configuration 3, completing the classification task in $139.15s$ on average. The reduction in battery consumption is likely due to a much shorter overall runtime rather than an indication of less exertion by the devices, based on experiments using the EMNIST dataset, shown in Figure 6(a). Although Configuration 2 is still 32% faster than Configuration 3, average battery consumption is nearly 50% higher. This illustrates a trade-off dilemma for precomputation, where battery expended to achieve speedup must not exceed battery saved by reducing runtime. We note

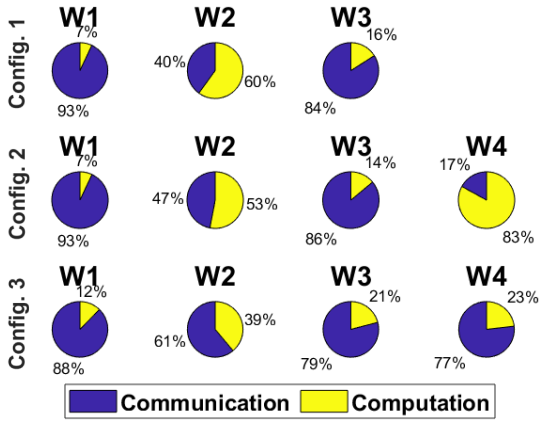


Figure 8. Comparison of average communication and computation time for worker devices using our proposed algorithms.

that precomputation was repeated for each trial of Configuration 2, however *this is only needed if secrecy of the key is at risk*. For most use cases, periodic precomputation is sufficient, similar to periodically updating passwords. In other words, our results represent worst-case scenario battery consumption.

Impact on communication and computation time. Our preliminary experiments in Section III show an overwhelming majority of runtime is spent on computation over communication, however our proposed algorithms are able to reverse this effect under most circumstances as shown in Figure 8. Compared to existing works, our proposed algorithms introduce more iterations of user uploads and downloads to facilitate matrix multiplication reordering, reducing computational complexity at the cost of increasing communication time. However, the magnitude of computational speedup significantly outweighs the communication cost as a linear increase in number of transmissions can yield exponential reduction in runtime and multiplication complexity (i.e. from $\mathcal{O}(rs^2)$ to $\mathcal{O}(rs)$). We find that conditions where the computation-communication time ratio does not change as significantly are mostly tied to specific devices (i.e. worker 2), suggesting that computational speedup is partially dependent on hardware specifications. The long communication time for worker 4 under Configuration 2 is a notable outlier, possibly due to a weak Wi-Fi connection during experimentation. Optimizing device connectivity is itself a large research area, however some works [42] suggest access point selection assisted by machine learning can further reduce communication times.

VIII. Discussion

Worker scalability. Our experiments used 3-worker and 4-worker setups as examples, however, our approach is scalable to serve any number of workers based on the number of partitions desired and devices available. Algorithm 3, for example, is designed based on the preservation of data privacy for even-numbered partitions. However, odd-numbered worker configurations are possible by tasking the odd worker (i.e. the $[2L+1]th$ device) with computing the product RS. In all other cases, RS can be partitioned like the input data or precomputed locally by the user. 2-worker configurations are also possible by considering the same experimental setup as Configuration 1, $L = 1$ (i.e. no partitioning), without enlisting a third device for computing RS.

Worker robustness. Overall performance is constrained by capabilities of the least-powerful worker (i.e., the straggler device). While devices can generally operate asynchronously, the user must wait for at least i workers to reply before starting a new iteration. For a simple network, i must equal N if every worker is considered necessary for the computation process. However, it is possible to introduce redundancy in the system such that $i < N$. One possibility is to leverage more workers than partitions, assigning duplicate data to extraneous devices. This scheme is more robust as it can tolerate the worst-case response (i.e., worker never replies). Alternatively, faster workers can volunteer to accept additional partitions if they are able to compute the initial assignment quickly enough. We note that such schemes may adversely affect tolerance to collusion, or the cooperation of workers to deduce the original input data without the user's knowledge. If we suppose that T workers are curious and attempt to uncover the identities of matrix A or B, the solution is only possible if $T = N$, assuming $N = i$. If multiple copies of a partition W_i exist, or worker i is able to obtain a copy of partition w_j where $j \neq i$, then the minimum T workers needed to uncover A or B decreases. We leave these scenarios for future work.

Hardware robustness. We also perform additional evaluations using the same procedure in Section VII with a secondary set of homogeneous (i.e. featuring duplicate models) mobile IoT devices. For the EMNIST dataset, the average runtimes for Configurations 1-3 are $22.64s$, $34.73s$, and $53.42s$, showing speedup of $10\times$, $13\times$, and $11\times$ respectively. This reinforces our previous observations, where our algorithms outperform existing secure coded distributed algorithms by $7 \sim 10\times$. Meanwhile, we are able to reduce battery consumption by $4 \sim 17\times$ across all configurations, with 4-worker schemes showing the most improvement. These results suggest our proposed algorithms can efficiently accomplish distributed learning tasks while agnostic to hardware models.

IX. Conclusion

We present the first step towards bringing secure coded distributed computation algorithms onto mobile IoT, proposing lightweight secure coding algorithms specifically optimized for commercial hardware. We develop an Android-based framework to evaluate SDMM algorithms under diverse real-world settings and studied performance of existing proposals to identify limitations to improve upon. Experiments on multiple commercial mobile IoT networks show our proposed algorithms can protect the privacy of outsourced data while reducing both runtime and battery consumption by over $10\times$.

Acknowledgment

This work was partially supported by the National Science Foundation Grants CCF1909963, CNS1801630, and CNS1801708.

REFERENCES

[1] X. L. Li, B. Veeravalli, and C. Ko, "Distributed image processing on a network of workstations," *International Journal of Computers and Applications*, vol. 25, no. 2, pp. 136–145, 2003.

[2] H. B. Prajapati and S. K. Vij, "Analytical study of parallel and distributed image processing," in *2011 International Conference on Image Information Processing*. IEEE, 2011, pp. 1–6.

[3] K. G. Narra, Z. Lin, G. Ananthanarayanan, S. Avestimehr, and M. Annavaram, "Collage inference: Using coded redundancy for low variance distributed image classification," *arXiv preprint arXiv:1904.12222*, 2019.

[4] A. M. Elmisery and H. Fu, "Privacy preserving distributed learning clustering of healthcare data using cryptography protocols," in *2010 IEEE 34th Annual Computer Software and Applications Conference Workshops*. IEEE, 2010, pp. 140–145.

[5] M. Maity, D. Dhane, T. Mungle, A. K. Maiti, and C. Chakraborty, "Web-enabled distributed health-care framework for automated malaria parasite classification: An e-health approach," *Journal of Medical Systems*, vol. 41, no. 12, p. 192, 2017.

[6] B. Tang, Z. Chen, G. Hefferman, T. Wei, H. He, and Q. Yang, "A hierarchical distributed fog computing architecture for big data analysis in smart cities," in *Proceedings of the ASE BigData & SocialInformatics 2015*, 2015, pp. 1–6.

[7] P. Chamoso, A. González-Briones, F. De La Prieta, G. K. Venyagamoorthy, and J. M. Corchado, "Smart city as a distributed platform: Toward a system for citizen-oriented management," *Computer Communications*, vol. 152, pp. 323–332, 2020.

[8] L. Zhou, Y. Zhang, K. Song, W. Jing, and A. V. Vasilakos, "Distributed media services in p2p-based vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 60, no. 2, pp. 692–703, 2010.

[9] J. Wang, C. Jiang, Z. Han, Y. Ren, R. G. Maunder, and L. Hanzo, "Taking drones to the next level: Cooperative distributed unmanned-aerial-vehicular networks for small and mini drones," *Ieee vehIcular technology magazIne*, vol. 12, no. 3, pp. 73–82, 2017.

[10] F. Dressler, G. S. Pannu, F. Hagenauer, M. Gerla, T. Higuchi, and O. Altintas, "Virtual edge computing using vehicular micro clouds," in *2019 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2019, pp. 537–541.

[11] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1514–1529, 2017.

[12] J. So, B. Guler, A. S. Avestimehr, and P. Mohassel, "Codedprivateml: A fast and privacy-preserving framework for distributed machine learning," *arXiv preprint arXiv:1902.00641*, 2019.

[13] R. G. L. D'Oliveira, S. El Rouayheb, and D. Karpuk, "Gasp codes for secure distributed matrix multiplication," in *2019 IEEE International Symposium on Information Theory (ISIT)*, 2019, pp. 1107–1111.

[14] R. Bitar, Y. Xing, Y. Keshtkarjahromi, V. Dasari, S. El Rouayheb, and H. Seferoglu, "Prac: private and rateless adaptive coded computation at the edge," in *Disruptive Technologies in Information Sciences II*, vol. 11013. International Society for Optics and Photonics, 2019, p. 110130T.

[15] Q. H. Zhenhua Liu, Bin Li, "Secure and verifiable outsourcing protocol for non-negative matrix factorisation," *International Journal of High Performance Computing and Networking*, 2017.

[16] K. Li, M. Tao, and Z. Chen, "Exploiting computation replication for mobile edge computing: A fundamental computation-communication tradeoff study," *IEEE Transactions on Wireless Communications*, vol. 19, no. 7, pp. 4563–4578, 2020.

[17] X. Hu, K. Wong, and K. Yang, "Wireless powered cooperation-assisted mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 17, no. 4, pp. 2375–2388, 2018.

[18] L. Wang, D. Zhang, Y. Wang, C. Chen, X. Han, and A. M'hamed, "Sparse mobile crowdsensing: challenges and opportunities," *IEEE Communications Magazine*, vol. 54, no. 7, pp. 161–167, 2016.

[19] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2017.

[20] T. Alladi, V. Chamola, B. Sikdar, and K.-K. R. Choo, "Consumer iot: Security vulnerability case studies and solutions," *IEEE Consumer Electronics Magazine*, vol. 9, no. 2, pp. 17–25, 2020.

[21] U. Fiore, A. Castiglione, A. De Santis, and F. Palmieri, "Exploiting battery-drain vulnerabilities in mobile smart devices," *IEEE Transactions on Sustainable Computing*, vol. 2, no. 2, pp. 90–99, 2017.

[22] J. Kakar, A. Khristoforov, S. Ebadifar, and A. Sezgin, "Uplink-downlink tradeoff in secure distributed matrix multiplication," *arXiv preprint arXiv:1910.13849*, 2019.

[23] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[24] F. Tramer and D. Boneh, "Slalom: Fast, verifiable and private execution of neural networks in trusted hardware," *arXiv preprint arXiv:1806.03287*, 2018.

[25] A. Shamir, "How to share a secret," in *Communications of the ACM*, vol. 22, no. 11, 1979, pp. 612–613.

[26] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, "Emnist: Extending mnist to handwritten letters," in *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017, pp. 2921–2926.

[27] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[28] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[29] S. Dixit, D. K. Jain, and A. Saxena, "An approach for secret sharing using randomised visual secret sharing," in *2014 Fourth International Conference on Communication Systems and Network Technologies*, 2014, pp. 847–850.

[30] L. Bai, "A reliable (k, n) image secret sharing scheme," in *2006 2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing*, 2006, pp. 31–36.

[31] S. Washio and Y. Watanabe, "Security of audio secret sharing scheme encrypting audio secrets with bounded shares," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 7396–7400.

[32] N. Al Ebri, Joonsang Baek, and C. Y. Yeun, "Study on secret sharing schemes (sss) and their applications," in *2011 International Conference for Internet Technology and Secured Transactions*, 2011, pp. 40–45.

[33] R. Bitar, P. Parag, and S. El Rouayheb, "Minimizing latency for secure distributed computing," in *2017 IEEE International Symposium on Information Theory (ISIT)*, 2017, pp. 2900–2904.

[34] Z. Jia and S. A. Jafar, "On the capacity of secure distributed matrix multiplication," *arXiv preprint arXiv:1908.06957*, 2019.

[35] W. Chang and R. Tandon, "On the capacity of secure distributed matrix multiplication," in *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1–6.

[36] H. Yang and J. Lee, "Secure distributed computing with straggling servers using polynomial codes," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 1, pp. 141–150, 2019.

[37] R. G. L. D'Oliveira, S. E. Rouayheb, D. Heinlein, and D. Karpuk, "Degree tables for secure distributed matrix multiplication," in *2019 IEEE Information Theory Workshop (ITW)*, 2019, pp. 1–5.

[38] ——, "Notes on communication and computation in secure distributed matrix multiplication," *arXiv preprint arXiv:2001.05568*, 2020.

[39] U. Sheth, S. Dutta, M. Chaudhari, H. Jeong, Y. Yang, J. Kohonen, T. Roos, and P. Grover, "An application of storage-optimal matdot codes for coded matrix multiplication: Fast k-nearest neighbors estimation," in *2018 IEEE International Conference on Big Data (Big Data)*, 2018, pp. 1113–1120.

[40] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, "A fundamental tradeoff between computation and communication in distributed computing," *IEEE Transactions on Information Theory*, vol. 64, no. 1, pp. 109–128, 2018.

[41] W.-T. Chang and R. Tandon, "On the capacity of secure distributed matrix multiplication," in *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2018, pp. 1–6.

[42] C. Pei, Z. Wang, Y. Zhao, Z. Wang, Y. Meng, D. Pei, Y. Peng, W. Tang, and X. Qu, "Why it takes so long to connect to a wifi access point," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, 2017, pp. 1–9.