Adaptive Gap Entangled Polynomial Coding for Multi-Party Computation at the Edge

Elahe Vedadi
University of Illinois at Chicago
evedad2@uic.edu

Yasaman Keshtkarjahromi
Seagate Technology
yasaman.keshtkarjahromi@seagate.com

Hulya Seferoglu
University of Illinois at Chicago
hulya@uic.edu

***Abstract*—Multi-party computation (MPC) is promising for designing privacy-preserving machine learning algorithms at edge networks. An emerging approach is coded-MPC (CMPC), which advocates the use of coded computation to improve the performance of MPC in terms of the required number of workers involved in computations. The current approach for designing CMPC algorithms is to merely combine efficient coded computation constructions with MPC. Instead, we propose a new construction; Adaptive Gap Entangled polynomial (AGE) codes, where the degrees of polynomials used in computations are optimized for MPC. We show that MPC with AGE codes (AGE-CMPC) perform better than existing CMPC algorithms in terms of the required number of workers as well as storage, communication and computation load.**

I. Introduction

MASSIVE amount of data is generated at edge networks. For example, the data generated by IoT devices are expected to reach 73.1 ZB by 2025, growing from 18.3 ZB in 2019 [1]. This vast data is expected to be processed in real-time in many time sensitive edge applications, which is extremely challenging if not impossible with existing centralized cloud due to limited bandwidth between an edge network and centralized cloud [2]–[4].

We consider a distributed computing system at the edge, where data is generated and collected by end devices, Fig. 1. Computationally intensive aspects are distributively processed by edge servers and a central server collects the outcome of the processed data. In this context, it is crucial to design efficient computation mechanisms at edge servers by taking into account the limited resources, including the number of edge serves, computing power, storage, and communication cost, while preserving privacy of data at end devices.

Multi-party computation (MPC) is a privacy-preserving distributed computing framework [5]. In MPC, several parties (end devices in Fig. 1) have private data and the goal is to compute a function of data collectively with the participation of all parties, while preserving privacy, *i.e.,* each party only knows its own information. MPC can be categorized into cryptographic solutions [6], [7] and information-theoretic solutions [8]. In this paper, our focus is on the information-theoretic MPC solution; BGW (Ben-Or, Goldwasser and Widgerson) [8] using Shamir's secret-sharing scheme [9] thanks to its lower computational complexity and quantum safe nature [10].

This work was supported in parts by the Army Research Lab (ARL) under Grant W911NF-2120272 and National Science Foundation (NSF) under Grants CCF-1942878 and CNS-1801708.

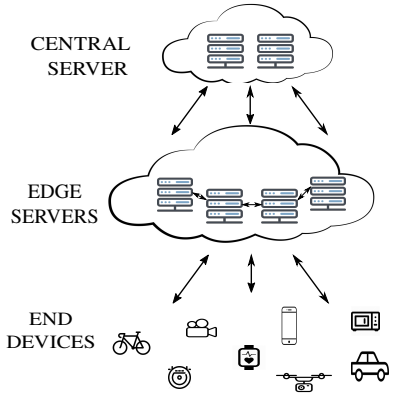


Fig. 1. An edge computing system. End devices generate and/or collect data, edge servers process data, and a central server collects the outcome of the processed data.

Despite its potential, BGW does not take into account the limited resources at the edge.

An emerging approach is coded-MPC (CMPC), which advocates the use of coded computation [11], [12] to improve the performance of BGW in terms of the required number of workers (edge servers in Fig. 1) involved in computationally intensive computations. For example, there may be two end devices in Fig. 1 possessing private matrices A and B. The goal is to calculate $Y = A^T B$ with the help of edge servers, while preserving privacy. This multiplication is a computationally intensive task when the sizes of A and B are large.

The common approach for designing CMPC algorithms is to merely combine efficient coded computation constructions with MPC. This approach fails short of being efficient as it does not take into account the interaction between coded computation and MPC [13]. Indeed, CMPC mechanisms based on Shamir's secret shares create a polynomial for each matrix for matrix multiplication with two parts; *coded* and *secret* terms [14]–[16]. The multiplication of matrices, *i.e.,* the multiplication of these polynomials create cross terms of coded and secret terms, some of which are not used in the decoding process, so named *garbage terms* [13].

The garbage terms are crucial for the performance of CMPC. In fact, even if a construction is optimized for coded computation, it may not perform well in CMPC. PolyDot coded MPC [13] is better than entangled polynomial coded MPC (Entangled-CMPC) [14] in terms of the required number of workers for a range of colluding workers. This result is surprising as entangled polynomial codes always outperform

PolyDot codes in terms of the number of required workers for coded computation [17]. Motivated by this observation, we propose a new construction in this paper; Adaptive Gap Entangled (AGE) codes, where the degrees of polynomials used in computations are optimized for MPC. We show through analysis and simulations that MPC with AGE codes performs better than existing CMPC algorithms including PolyDot-CMPC [13], Entangled-CMPC [14], SSMM [15], and GCSA-NA [16] in terms of the required number of workers as well as storage, communication and computation load.

II. Related Work

Coded computation advocates higher reliability and smaller delay in distributed computation by introducing redundancy in the offloaded sub-tasks to the workers [18]. Significant effort is being put on constructing codes for fast and distributed matrix-vector multiplication [18], [19], matrix-matrix multiplication [17], [20]–[22], dot product and convolution of two vectors [23], [24], gradient descent [25]–[27], distributed optimization [28], Fourier transform [29], and linear transformations [30]. As compared to this line of works, we consider privacy-preserving computation at edge networks.

Privacy is studied in coded computation. In [31]–[33], the problem of matrix-matrix multiplication is considered for the case that a master possesses the input data and would like to perform multiplication on the data with the help of parallel workers, while the data is kept confidential from the workers. In [34] and [35], privacy is addressed for the same system model of master-worker setup, but for matrix-vector multiplication. As compared to this line of work, we focus on the MPC system setup, where there are multiple sources each having private input data, and the goal is that a master learns the result of computation of matrix multiplication on the input data with the help of parallel workers. The input data should be kept confidential from workers and the master according to the information-theoretic security.

There is a line of work investigating CMPC. Lagrange Coded Computing (LCC) is designed [36] in a coded computation setup for security and privacy. This work is extended for MPC setup [37]. The problem of limited memory at each party in MPC setup is addressed in [38] by leveraging polynomial coded computation. This work is generalized using entangled polynomial codes for block-wise matrix multiplication [14]. Secure multi-party batch matrix multiplication is considered in [15], [16], which modify the MPC system setup by employing the idea of noise alignment to reduce the communication load among workers. As compared to this line of work, we design AGE-CMPC by taking into account the interaction of coded computation and MPC for the limited edge resources.

III. System Model

Notations. *Set of polynomial degrees:* The set of powers of the terms in a given polynomial $f(x) = \sum_{i=0}^{n} a_i x^i$ with non-zero coefficients is denoted by $\mathbf{P}(f(x))$, *i.e.,* $\mathbf{P}(f(x)) = \{i \in \mathbb{Z} : 0 \leq i \leq n,\ a_i \neq 0\}$.

Set definitions and operations: We use the following standard notations for arbitrary sets $\mathbf{A}$ and $\mathbf{B}$, where the elements of $\mathbf{A}$, $\mathbf{B}$ are integers, *i.e.,* $a, b \in \mathbb{Z}$; (i) $\mathbf{A} + \mathbf{B} = \{a + b : a \in \mathbf{A},\ b \in \mathbf{B}\}$, and (ii) $\mathbf{A} + b = \{a + b : a \in \mathbf{A}\}$. The cardinality of $\mathbf{A}$ is shown by $|\mathbf{A}|$. The set of integers between a and b is represented by Ω_a^b, *i.e.,* $\Omega_a^b = \{a, \dots, b\}$. Furthermore, $k|m$ means that m is divisible by k, *i.e.,* $\mod \{m, k\} = 0$.

Matrix splitting: If a matrix A is divided into s row-wise and t column-wise partitions, it is represented as

$$A = \begin{bmatrix} A_{0,0} & \dots & A_{0,t-1} \\ \vdots & \ddots & \vdots \\ A_{s-1,0} & \dots & A_{s-1,t-1} \end{bmatrix}, \tag{1}$$

where for $A \in \mathbb{F}^{m \times m}$, $A_{j,i} \in \mathbb{F}^{\frac{m}{s} \times \frac{m}{t}}$ for $j \in \Omega_0^{s-1}$ and $i \in \Omega_0^{t-1}$.

Setup. We consider a system setup with E end devices (sources), N edge servers (workers), and a central server (master) as shown in Fig. 1. Each source $e \in \mathcal{E}$, where $E = |\mathcal{E}|$, has private data $X_e \in \mathbb{F}^{\mu \times \nu}$, where $\mathbb{F}$ is a finite field. Each source is connected to all workers, and offloads its data to workers for privacy-preserving computation. Each worker $W_n, n \in \mathcal{N}$ ($|\mathcal{N}| = N$) is connected to other workers as well as the master. The sources, workers, and the master are all edge devices with limited available resources.

Application. The goal is to calculate a function of per source data; $Y = \gamma(X_1, \dots, X_E)$, while the privacy of data $X_1, \dots, X_E$ is preserved. While function $\gamma(.)$ could be any polynomial function in MPC setup, we focus on matrix multiplication as (i) we would like to present our ideas in a simple way, and (ii) matrix multiplication forms an essential building block of many signal processing and machine learning algorithms (gradient descent, classification, etc.) [11]. In particular, we consider $Y = \gamma(A, B) = A^T B$, where $X_1 = A$, $X_2 = B$, $A \in \mathbb{F}^{m \times m}$, $B \in \mathbb{F}^{m \times m}$. We note that we use square matrices from two sources for easy exposition, and it is straightforward to extend our results for more general matrices and larger number of sources.

Attack Model. We assume a semi-honest system model, where the sources, the workers, and the master follow the defined protocols, but they are curious about the private data. We assume that z among N workers can collude to maximize the information that they can access. We design our AGE-CMPC mechanism against z colluding workers to provide privacy-preserving computation.

Privacy Requirements. We define the privacy requirements from the perspective of the sources, workers, and the master.

Source perspective: Source devices should not learn anything about the private data of any other sources. This requirement is satisfied in our system as there is no communication among the source devices. Also, the workers and the master do not send any information to sources.

Worker perspective: Workers should not learn anything when they communicate with each other as well as when they receive data from the sources, *i.e.,* $\tilde{H}(\chi_1, \dots, \chi_E | \bigcup_{n \in \mathcal{N}_c} (\{G_{n'}(\alpha_n), n' \in \Omega_1^N\}, \bigcup_{e \in \Omega_1^E} F_e(\alpha_n))) = \tilde{H}(\chi_1, \dots, \chi_E)$, where $\tilde{H}$ denotes the Shannon entropy, α_n

is a a priori parameter associated by worker W_n, $G_{n'}(\alpha_n)$ is the data each worker W_n receives from another worker $W_{n'}$, $F_e(\alpha_n)$ is the data received by each worker W_n from source e for $n \in \mathcal{N}_c$, and $\mathcal{N}_c$ is any subset of $\mathcal{N}$ satisfying $|\mathcal{N}_c| \leq z$.

Master perspective: The master node should not learn anything more than the final result Y, *i.e.,* $\tilde{H}(\chi_1, \ldots, \chi_E | Y, \bigcup_{n \in \mathcal{N}} I(\alpha_n)) = \tilde{H}(\chi_1, \ldots, \chi_E | Y)$, where $I(\alpha_n)$ is the data that the master receives from W_n.

IV. Adaptive Gap Entangled Polynomial Coding

In this section, we introduce Adaptive Gap Entangled polynomial (AGE) codes and present our CMPC design with AGE codes; AGE-CMPC. We provide the proofs of all our theorems and lemmas in the extended version of this paper [39].

A. AGE Codes

We consider the generalized formulation [17] for coded computation of matrices A and B, where A and B are represented with the following polynomials.

$$C_A(x) = \sum_{i=0}^{t-1} \sum_{j=0}^{s-1} A_{i,j} x^{j\alpha + i\beta},$$
$$C_B(x) = \sum_{k=0}^{s-1} \sum_{l=0}^{t-1} B_{k,l} x^{(s-1-k)\alpha + \theta l} \quad (2)$$

where $\alpha, \beta, \theta \in \mathbb{N}$, $A_{i,j} \in A^T$ and $B_{k,l} \in B$. In this setup, instead of multiplying $Y = A^T B$, we can multiply $C_A(x)$ and $C_B(x)$, which can be decomposed to support distributed computing. Several codes that have been designed for coded computation can be considered as the special case of (2) by considering different values of (α, β, θ). For example, PolyDot codes [22] correspond to $(\alpha, \beta, \theta) = (t, 1, t(2s-1))$, while Generalized PolyDot codes [40] and entangled polynomial codes [17] follow $(\alpha, \beta, \theta) = (1, s, ts)$, where t is the number of column-wise partitions and s is the number of row-wise partitions of matrices A and B.

We design our AGE codes by considering $(\alpha, \beta, \theta) = (1, s, ts+\lambda)$, where λ is a parameter in the range of $0 \leq \lambda \leq z$, which we optimize to achieve the minimum required number of workers for MPC. We note that entangled polynomial codes [17] also follows (2) for $\alpha = 1$ and $\beta = s$, but as they are designed for coded computation, θ is optimized to achieve the minimum recovery threshold, so θ is set to $\theta = ts$. Instead, we set $\theta = ts + \lambda$ and optimize λ. Next, we prove the decodability of our AGE codes.

Theorem 1: AGE code guarantees the decodability of $Y = A^T B$ from the polynomial $C_Y(x) = C_A(x) C_B(x)$. □

B. AGE-CMPC

Phase 1 - Sources Share Data with Workers. In the first phase, sources split their matrices A and B into $s \geq 1$ row-wise and $t \geq 1$ column-wise partitions, where $s|m$ and $t|m$ are satisfied. We note that in AGE-CMPC, we exclude the case of no partitioning, *i.e.,* $s = t = 1$, where coding is not required. Assuming $A_{i,j} \in A^T$ and $B_{k,l} \in B$, where $i, l \in \Omega_0^{t-1}$, $j, k \in \Omega_0^{s-1}$, the sources create polynomials $F_A(x)$ and $F_B(x)$, which comprise coded and secret terms; *i.e.,* $F_{i'}(x) = C_{i'}(x) + S_{i'}(x)$, $i' \in \{A, B\}$, where $C_{i'}(x)$'s are the coded terms defined by AGE codes and $S_{i'}(x)$'s are the secret terms which are defined by our AGE-CMPC design, which we explain next.

Let $\mathbf{P}(C_A(x))$ and $\mathbf{P}(C_B(x))$ be the set of all powers in the polynomials $C_A(x)$ and $C_B(x)$, with non-zero coefficients.

$$\mathbf{P}(C_A(x)) = \{j + si : i \in \Omega_0^{t-1},\ j \in \Omega_0^{s-1}\} = \{0, \ldots, ts-1\}, \quad (3)$$

$$\mathbf{P}(C_B(x)) = \{(s-1-k) + l(ts+\lambda) : k \in \Omega_0^{s-1},\ l \in \Omega_0^{t-1}\}, \quad (4)$$

where, $s, t \in \mathbb{N}$ and $\lambda \in \Omega_0^z$. $S_A(x)$ and $S_B(x)$ are defined such that $\mathbf{P}(C_A(x)S_B(x))$, $\mathbf{P}(S_A(x)C_B(x))$, and $\mathbf{P}(S_A(x)S_B(x))$ do not have common terms with the important powers of $\mathbf{P}(C_A(x)C_B(x))$, which are equal to $(s-1) + si + (ts+\lambda)l$ for $i, l \in \Omega_0^{t-1}$. The reason is that $\{(s-1-k+j)\alpha + i\beta + \theta l :\ i, l \in \Omega_0^{t-1},\ j, k \in \Omega_0^{s-1}\ s, t \in \mathbb{N}\}$ is the set of powers of polynomial $C_A(x)C_B(x)$. The components of the desired product $Y = A^T B$ are equal to $Y_{i,l} = \sum_{j=0}^{s-1} A_{ij} B_{jl}$, for $i, l \in \Omega_0^{t-1}$, that are the summation of the coefficients of the terms with $j = k \in \Omega_0^{s-1}$. Therefore $\{(s-1)\alpha + i\beta + \theta l :\ i, l \in \Omega_0^{t-1},\ s, t \in \mathbb{N}\}$ is the set of important powers of $C_A(x)C_B(x)$, and for successful recovery of Y, these components should not have any overlap with the other components, *i.e.,* garbage terms. In other words, the following conditions should be satisfied:

$$\begin{aligned}
&\text{C1: } (s-1) + si + (ts+\lambda)l \notin \mathbf{P}(S_B(x)) + \mathbf{P}(C_A(x)),\\
&\text{C2: } (s-1) + si + (ts+\lambda)l \notin \mathbf{P}(S_A(x)) + \mathbf{P}(C_B(x)),\\
&\text{C3: } (s-1) + si + (ts+\lambda)l \notin \mathbf{P}(S_A(x)) + \mathbf{P}(S_B(x)). \quad (5)
\end{aligned}$$

Our strategy for determining $\mathbf{P}(S_A(x))$ and $\mathbf{P}(S_B(x))$ is as follows. First, we set the elements of $\mathbf{P}(S_B(x))$ as z consecutive elements starting from the maximum important power, *i.e.,* $s-1+s(t-1)+(ts+\lambda)(t-1)$ plus one; $\mathbf{P}(S_B(x)) = \{ts + (ts+\lambda)(t-1), \ldots, ts + (ts+\lambda)(t-1) + z - 1\}$ or equivalently: $\mathbf{P}(S_B(x)) = \{ts + \theta(t-1) + r, r \in \Omega_0^{z-1}\}$. We note that the elements of $\mathbf{P}(C_A(x))$ and $\mathbf{P}(S_A(x))$ are powers of polynomials, so they are non-negative. Therefore, by starting the elements of $\mathbf{P}(S_B(x))$ from the maximum important power plus one, C1 and C3 are satisfied. Then, we find all elements of the subset of $\mathbf{P}(S_A(x))$, starting from the minimum possible element, that satisfies C2 in (5). Using this rule, we can determine $S_A(x)$ and $S_B(x)$ as

$$S_A(x) = \begin{cases} S_{A_1}(x) & z > \lambda, \text{ and } t \neq 1 \\ S_{A_2}(x) & z = \lambda, \text{ or } t = 1 \end{cases} \quad (6)$$

where $S_{A_1}(x) = \sum_{w=0}^{\lambda-1} \sum_{l=0}^{q-1} \bar{A}_{(w+\theta l)} x^{ts+\theta l+w} + \sum_{u=0}^{z-1-q\lambda} \bar{A}_{(u+\lambda+\theta(q-1))} x^{ts+\theta q+u}$, $S_{A_2}(x) = \sum_{u=0}^{z-1} \bar{A}_u x^{ts+u}$, and $\bar{A}_{(w+\theta l)}$, $\bar{A}_{(u+\lambda+\theta(q-1))}$, and $\bar{A}_u$ are chosen independently and uniformly at random in $\mathbb{F}^{\frac{m}{t} \times \frac{m}{s}}$, and

$q = \min\{\lfloor \frac{z-1}{\lambda} \rfloor, t-1\}$.

$$S_B(x) = \sum_{r=0}^{z-1} \bar{B}_r x^{ts+\theta(t-1)+r}, \quad (7)$$

where $\bar{B}_r$ is chosen independently and uniformly at random in $\mathbb{F}^{\frac{m}{s} \times \frac{m}{t}}$.

Theorem 2: The polynomials $S_A(x)$ and $S_B(x)$ defined in (6) and (7) satisfy the conditions in (5). □

In phase 1, source 1 shares $F_A(\alpha_n)$ and source 2 shares $F_B(\alpha_n)$ with each worker W_n. Due to using z random terms in constructing $F_A(x)$ and $F_B(x)$, no information about A and B is revealed to any workers.

Phase 2 - Workers Compute and Communicate. The second phase consists of workers processing data received from the sources and sharing the results with each other. In this phase, each worker W_n calculates $H(\alpha_n) = F_A(\alpha_n)F_B(\alpha_n)$, where $H(x)$ is defined as:

$$H(x) = \sum_{n=0}^{\deg(F_A(x))+\deg(F_B(x))} H_n x^n = F_A(x)F_B(x), \quad (8)$$

where $H_u = \sum_{j=0}^{s-1} A_{i,j}B_{j,l}$ are the coefficients that are required for calculating A^TB, *i.e.,* $u = si + (s-1) + \theta l$ for $i, l \in \Omega_0^{t-1}$. Each worker W_n has the knowledge of one point from $H(x)$ through calculation of $H(\alpha_n) = F_A(\alpha_n)F_B(\alpha_n)$. By applying Lagrange interpolation on (8), there exist $r_n^{(i,l)}$'s such that

$$H_u = \sum_{j=0}^{s-1} A_{ij}B_{jl} = \sum_{n=1}^{N} r_n^{(i,l)} H(\alpha_n). \quad (9)$$

Thus, each worker W_n multiplies $r_n^{(i,l)}$'s with $H(\alpha_n)$ and shares them with the other workers, securely. In particular, for each worker W_n, there are t^2 coefficients of $r_n^{(i,l)}$. Therefore, each worker W_n creates a polynomial $G_n(x)$ with the first t^2 terms allocated to multiplication of $r_n^{(i,l)}$ with $H(\alpha_n)$ and the last z terms allocated to random coefficients to keep $H(\alpha_n)$ confidential from z colluding workers:

$$G_n(x) = \sum_{i=0}^{t-1}\sum_{l=0}^{t-1} r_n^{(i,l)} H(\alpha_n) x^{i+tl} + \sum_{w=0}^{z-1} R_w^{(n)} x^{t^2+w}, \quad (10)$$

where $R_w^{(n)}, w \in \Omega_0^{z-1}$ are chosen independently and uniformly at random from $\mathbb{F}^{\frac{m}{t} \times \frac{m}{t}}$. Each worker W_n sends $G_n(\alpha_{n'})$ to all other workers $W_{n'}$. After all the data exchanges, each worker $W_{n'}$ has the knowledge of $G_n(\alpha_{n'})$, which sums them up and sends it to the master in the last phase. The following equation represent the polynomial that is equal to the summation of $G_n(x)$:

$$I(x) = \sum_{n=1}^{N} G_n(x), \quad (11)$$

which can be equivalently written as:

$$\begin{aligned} I(x) &= \sum_{i=0}^{t-1}\sum_{l=0}^{t-1}\sum_{n=1}^{N} r_n^{(i,l)} H(\alpha_n) x^{i+tl} + \sum_{w=0}^{z-1}\sum_{n=1}^{N} R_w^{(n)} x^{t^2+w} \\ &= \sum_{i=0}^{t-1}\sum_{l=0}^{t-1}\sum_{j=0}^{s-1} A_{ij}B_{jl} x^{i+tl} + \sum_{w=0}^{z-1}\sum_{n=1}^{N} R_w^{(n)} x^{t^2+w}. \quad (12) \end{aligned}$$

Phase 3 - Master Node Reconstructs $Y = A^TB$. As seen in (12), the coefficients for the first t^2 terms of $I(x)$ represent the components of the matrix $Y = A^TB$. On the other hand, the degree of $I(x)$ is $t^2 + z - 1$, therefore, the master can reconstruct $I(x)$ and extract $Y = A^TB$ after receiving $I(\alpha_n)$ from $t^2 + z$ workers.

Theorem 3: The total number of workers required to compute $Y = A^TB$ using AGE-CMPC, when there exist z colluding workers and each worker can work on at most $\frac{1}{st}$ fraction of data from each source due to the computation or storage constraints, is expressed as

$$N_{\text{AGE-CMPC}} = \begin{cases} \min_{\lambda} \Gamma(\lambda) & t \neq 1 \\ 2s + 2z - 1 & t = 1 \end{cases} \quad (13)$$

where $\Gamma(\lambda)$ is defined as

$$\Gamma(\lambda) = \begin{cases} \Upsilon_1(\lambda), & z > ts - s,\ \lambda = 0 \\ \Upsilon_2(\lambda), & z \leq ts - s,\ \lambda = 0 \\ \Upsilon_3(\lambda), & \lambda = z \\ \Upsilon_4(\lambda), & z > ts,\ 0 < \lambda < z \\ \Upsilon_5(\lambda), & z \leq ts,\ 0 < \lambda < z,\ ts < \lambda + s - 1 \\ \Upsilon_6(\lambda), & \lambda + s - 1 < z \leq ts,\ 0 < \lambda < z,\ q\lambda \geq s \\ \Upsilon_7(\lambda), & \lambda + s - 1 < z \leq ts,\ 0 < \lambda < z,\ q\lambda < s \\ \Upsilon_8(\lambda), & z \leq \lambda + s - 1 \leq ts,\ 0 < \lambda < z,\ q\lambda \geq s \\ \Upsilon_9(\lambda), & z \leq \lambda + s - 1 \leq ts,\ 0 < \lambda < z,\ q\lambda < s, \end{cases} \quad (14)$$

and $\Upsilon_1(0) = 2st^2+2z-1$, $\Upsilon_2(0) = st^2+3st-2s+t(z-1)+1$, $\Upsilon_3(z) = 2ts + (ts+z)(t-1) + 2z - 1$, $\Upsilon_4(\lambda) = (q+2)ts + \theta(t-1) + 2z - 1$, $\Upsilon_5(\lambda) = 3ts + \theta(t-1) + 2z - 1$, $\Upsilon_6(\lambda) = 2ts + \theta(t-1) + (q+2)z - q - 1$, $\Upsilon_7(\lambda) = \theta(t+1) + q(z-1) - 2\lambda + z + ts + \min\{0, z + s(1-t) - \lambda q - 1\}$, $\Upsilon_8(\lambda) = 2ts + \theta(t-1) + 3z + (\lambda+s-1)q - \lambda - s - 1$, $\Upsilon_9(\lambda) = \theta(t+1) + q(s-1) - 3\lambda + 3z - 1 + \min\{0, ts - z + 1 + \lambda q - s\}$, $s \geq 1$, $t \geq 2$, $s|m$, $t|m$ are satisfied, $\theta = ts + \lambda$. □

AGE-CMPC satisfies privacy requirements stated in Section III. The proof directly follows from [38] (Theorem 3).

C. AGE-CMPC in Perspective

In this section we compare AGE-CMPC with Entangled-CMPC [14], SSMM [15], GCSA-NA [16] (considering batch size as one), and PolyDot-CMPC [13] in terms of the number of required workers.

Lemma 4: $N_{\text{AGE-CMPC}}$ is always less than or equal to the number of workers required by Entangled-CMPC [14], SSMM [15], GCSA-NA (for one matrix multiplication) [16], and PolyDot-CMPC [13].

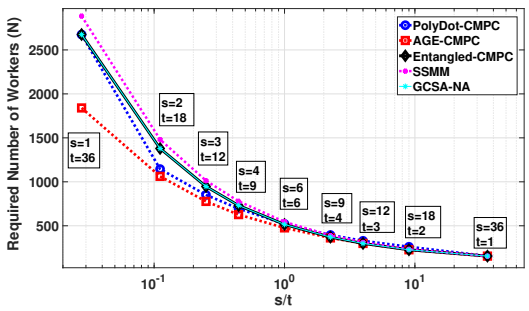


Fig. 2. Required number of workers.

V. Performance Evaluation

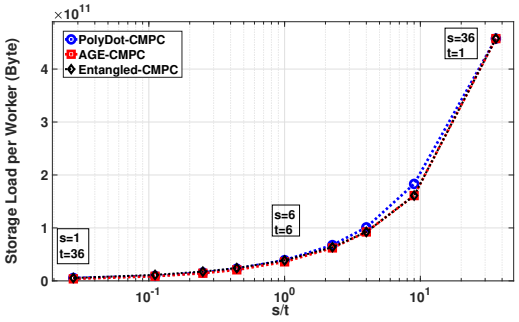
We evaluate the performance of our algorithm, and compare with the baselines, (i) PolyDot-CMPC [13], (ii) Entangled-CMPC [14], (iii) SSMM [15], and (iv) GCSA-NA [16]. The system model parameters are considered as follows: the size of each matrix A and B is $m \times m = 36000 \times 36000$, the number of colluding workers is $z = 42$, the number of partitions of matrices A and B is $st = 36$.

Fig. 2 shows the required number of workers needed to compute $Y = A^T B$ versus s/t, the number of row partitions over the number of column partitions. As seen, the required number of workers of AGE-CMPC is less than or equal to the other baselines, which confirms our Lemmas 4.

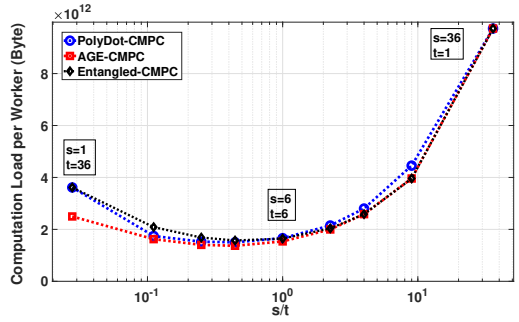
Fig. 3(a) shows the storage cost per worker, where the size of each stored scalar is 1 Byte, versus s/t. We note that the detailed calculations of storage, computation, and communication load is provided in the extended version of this paper [39]. AGE-CMPC reduces the storage load per worker as compared to baselines. The reason is that there is a direct relationship between the required number of workers and storage load per worker in CMPC setup when we fix s and t. As we described in phase 2, each worker n needs to compute polynomial $G_n(\alpha_{n'})$ and send it to workers n' for $n' \in \{0, \ldots, N\} \setminus \{n\}$, also it needs to receive $G_{n'}(\alpha_n)$ from worker n'. Therefore, the smaller required number of workers of AGE-CMPC results in the smaller storage load per worker as compared to PolyDot-CMPC and Entangled-CMPC.

Fig. 3(b) shows the computation cost per worker versus s/t. Similar to the discussion in Fig. 3(a), for the fixed amounts of s and t, required number of workers has a direct relation with computation load per worker, *i.e.,* larger amounts of workers results in larger computation load per worker. Therefore, computation load per worker of AGE-CMPC is less than or equal to the other methods. However, as seen in Fig. 3(b), computation load per worker does not have a monotonic behavior as different s and t partitions lead to different amount of computations.

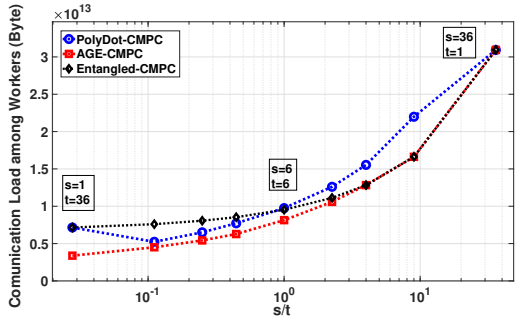
Fig. 3(c) shows the communication overhead versus s/t. The communication overhead consists of; (i) from sources to workers in phase 1, (ii) among workers in phase 2, and (iii) from workers to the master in phase 3. We consider the communication cost among workers as it is the dominating communication cost in this system. We assume that each



(a) Storage



(b) Computation



(c) Communication

Fig. 3. (a) Storage, (b) computation, and (c) communication loads.

scalar that is transmitted among workers is 1 Byte. Similar to our discussions for storage and computation, for the fixed amounts of s and t, required number of workers has a direct relationship with the communication load among workers, *i.e.,* larger amounts of workers results in larger communication load. Therefore, communication load among workers of AGE-CMPC is less than or equal to the other methods.

VI. Conclusion

We have investigated coded privacy-preserving computation using Shamir's secret sharing. We have designed a novel coded computation method; AGE codes that can be customized for coded privacy-preserving computations. We also designed a coded privacy-preserving computation mechanism; AGE coded MPC (AGE-CMPC) by employing AGE codes. We designed our algorithm such that it takes advantage of the "garbage terms". Also, we have analyzed AGE-CMPC in terms of the required number of workers as well as its computation, storage, and communication overhead, and shown that AGE-CMPC provides significant improvement.

REFERENCES

[1] R. Swearingen, "Idc report 2020: Iot growth demands rethink of long-term storage strategies, says idc," 2020.

[2] L. Peterson, T. Anderson, S. Katti, N. McKeown, G. Parulkar, J. Rexford, M. Satyanarayanan, O. Sunay, and A. Vahdat, "Democratizing the network edge," *SIGCOMM Comput. Commun. Rev.*, vol. 49, no. 2, pp. 31–36, May 2019. [Online]. Available: http://doi.acm.org/10.1145/3336937.3336942

[3] P. Levine and A. Horowitz, "Return to the edge and the end of cloud computing," 2017. [Online]. Available: https://www.youtube.com/watch?v=-QRXQTSZxdQ

[4] G. M. Research, "The edge will eat the cloud," 2017.

[5] J. Saia and M. Zamani, "Recent results in scalable multi-party computation," in *SOFSEM 2015: Theory and Practice of Computer Science*, G. F. Italiano, T. Margaria-Steffen, J. Pokorný, J.-J. Quisquater, and R. Wattenhofer, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 24–44.

[6] A. C.-C. Yao, "How to generate and exchange secrets," in *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, 1986, pp. 162–167.

[7] S. M. O. Goldreich and A. Wigderson, "How to play any mental game," in *Proc. of the 19th STOC*, 1987, pp. 218–229.

[8] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computation," in *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, 2019, pp. 351–371.

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

[10] U. Maurer, "Information-theoretic cryptography," in *Advances in Cryptology — CRYPTO' 99*, M. Wiener, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 47–65.

[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, March 2018.

[12] 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, Jan 2018.

[13] E. Vedadi, Y. Keshtkarjahromi, and H. Seferoglu, "Polydot coded privacy preserving multi-party computation at the edge," in *IEEE Signal Processing Advances in Wireless Communications (SPAWC) (invited paper)*, 2022. [Online]. Available: https://https://nrl.ece.uic.edu/

[14] H. A. Nodehi, S. R. H. Najarkolaei, and M. A. Maddah-Ali, "Entangled polynomial coding in limited-sharing multi-party computation," in *2018 IEEE Information Theory Workshop (ITW)*, 2018, pp. 1–5.

[15] J. Zhu, Q. Yan, and X. Tang, "Improved constructions for secure multi-party batch matrix multiplication," *IEEE Transactions on Communications*, vol. 69, pp. 7673–7690, 2021.

[16] Z. Chen, Z. Jia, Z. Wang, and S. A. Jafar, "Gcsa codes with noise alignment for secure coded multi-party batch matrix multiplication," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 1, pp. 306–316, 2021.

[17] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding," *IEEE Transactions on Information Theory*, vol. 66, no. 3, pp. 1920–1933, 2020.

[18] 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, 2018.

[19] N. S. Ferdinand and S. C. Draper, "Anytime coding for distributed computation," in *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2016, pp. 954–960.

[20] Q. Yu, M. A. Maddah-Ali, and S. Avestimehr, "Polynomial codes: an optimal design for high-dimensional coded matrix multiplication," in *NIPS*, 2017, pp. 4406–4416.

[21] K. Lee, C. Suh, and K. Ramchandran, "High-dimensional coded matrix multiplication," in *2017 IEEE International Symposium on Information Theory (ISIT)*, 2017, pp. 2418–2422.

[22] M. Fahim, H. Jeong, F. Haddadpour, S. Dutta, V. Cadambe, and P. Grover, "On the optimal recovery threshold of coded matrix multiplication," in *2017 55th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2017, pp. 1264–1270.

[23] S. Dutta, V. Cadambe, and P. Grover, ""short-dot": Computing large linear transforms distributedly using coded short dot products," *IEEE Transactions on Information Theory*, vol. 65, no. 10, pp. 6171–6193, 2019.

[24] ——, "Coded convolution for parallel and distributed computing within a deadline," in *2017 IEEE International Symposium on Information Theory (ISIT)*, 2017, pp. 2403–2407.

[25] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 06–11 Aug 2017, pp. 3368–3376. [Online]. Available: http://proceedings.mlr.press/v70/tandon17a.html

[26] W. Halbawi, N. Azizan, F. Salehi, and B. Hassibi, "Improving distributed gradient descent using reed-solomon codes," in *2018 IEEE International Symposium on Information Theory (ISIT)*, 2018, pp. 2027–2031.

[27] N. Raviv, I. Tamo, R. Tandon, and A. G. Dimakis, "Gradient coding from cyclic mds codes and expander graphs," *IEEE Transactions on Information Theory*, vol. 66, no. 12, pp. 7475–7489, 2020.

[28] C. Karakus, Y. Sun, S. Diggavi, and W. Yin, "Redundancy techniques for straggler mitigation in distributed optimization and learning," *Journal of Machine Learning Research*, vol. 20, no. 72, pp. 1–47, 2019. [Online]. Available: http://jmlr.org/papers/v20/18-148.html

[29] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Coded fourier transform," in *2017 55th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2017, pp. 494–501.

[30] Y. Yang, P. Grover, and S. Kar, "Computing linear transformations with unreliable components," *IEEE Transactions on Information Theory*, vol. 63, no. 6, pp. 3729–3756, 2017.

[31] 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, Jan 2019.

[32] J. Kakar, S. Ebadifar, and A. Sezgin, "On the capacity and straggler-robustness of distributed secure matrix multiplication," *IEEE Access*, vol. 7, pp. 45 783–45 799, 2019.

[33] R. G. L. D'Oliveira, S. El Rouayheb, and D. Karpuk, "Gasp codes for secure distributed matrix multiplication," *IEEE Transactions on Information Theory*, vol. 66, no. 7, pp. 4038–4050, 2020.

[34] R. Bitar, Y. Xing, Y. Keshtkarjahromi, V. Dasari, S. El Rouayheb, and H. Seferoglu, "Private and rateless adaptive coded matrix-vector multiplication," *EURASIP Journal on Wireless Communications and Networking*, 2021.

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

[36] Q. Yu, N. Raviv, J. So, and A. S. Avestimehr, "Lagrange coded computing: Optimal design for resiliency, security and privacy," *arXiv preprint, arXiv:1806.00939*, 2018.

[37] Q. Yu, N. Raviv, and A. S. Avestimehr, "Coding for private and secure multiparty computing," in *2018 IEEE Information Theory Workshop (ITW)*, 2018, pp. 1–5.

[38] H. Akbari-Nodehi and M. A. Maddah-Ali, "Secure coded multi-party computation for massive matrix operations," *IEEE Transactions on Information Theory*, vol. 67, no. 4, pp. 2379–2398, 2021.

[39] E. Vedadi, Y. Keshtkarjahromi, and H. Seferoglu, "Adaptive gap entangled polynomial coding for multi-party computation at the edge," 2022. [Online]. Available: https://arxiv.org/abs/2203.06759

[40] S. Dutta, Z. Bai, H. Jeong, T. M. Low, and P. Grover, "A unified coded deep neural network training strategy based on generalized polydot codes," *2018 IEEE International Symposium on Information Theory (ISIT)*, pp. 1585–1589, 2018.