Speeding Up Private Distributed Matrix Multiplication via Bivariate Polynomial Codes

Burak Hasırcıoğlu*, Jesús Gómez-Vilardebó†, and Deniz Gündüz*
*Imperial College London, UK, {b.hasircioglu18, d.gunduz}@imperial.ac.uk
†Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA), Barcelona, Spain, jesus.gomez@cttc.es

***Abstract*—We consider the problem of private distributed matrix multiplication under limited resources. Coded computation has been shown to be an effective solution in distributed matrix multiplication, both providing privacy against the workers and boosting the computation speed by efficiently mitigating stragglers. In this work, we propose the use of recently-introduced bivariate polynomial codes to further speed up private distributed matrix multiplication by exploiting the partial work done by the stragglers rather than completely ignoring them. We show that the proposed approach reduces the average computation time of private distributed matrix multiplication compared to its competitors in the literature while improving the upload communication cost and the workers' storage efficiency.**

I. Introduction

Matrix multiplication is a fundamental building block of many applications in signal processing and machine learning. For some applications, especially those involving massive matrices and stringent latency requirements, matrix multiplication in a single computer is infeasible, and distributed solutions need to be adopted. In such a scenario, the full multiplication task is first partitioned into smaller sub-tasks, which are then distributed across dedicated *workers*.

In this work, we address two main challenges in distributed matrix multiplication. The first one is the *straggler*s, which refers to unresponsive or very slow workers. If the completion of the full task requires the computations from all the workers, straggling workers become a significant bottleneck. To compensate for the stragglers, additional redundant computations can be assigned to workers. It has been recently shown that the use of error-correcting codes, by treating the slowest workers as erasures, instead of simply replicating tasks across workers, significantly lowers the overall computation time [1].

In the context of straggler mitigation, polynomial-type codes are studied in [2]–[5]. In these schemes, matrices are first partitioned and encoded using polynomial codes at the master server. Then, workers compute sub-products by multiplying these coded partitions and send the results back to the master for decoding. The minimum number of sub-tasks required to decode the result is referred to as the *recovery threshold*, and denoted by R_{th}. In these schemes, only one sub-product is assigned to each worker, and therefore, any work done by the workers beyond the fastest R_{th} is completely ignored. This is sub-optimal since the workers may have similar computational speeds, in which case most of the work done is lost.

In the *multi-message approach* [6]–[10], multiple sub-products are assigned to each worker and the result of each sub-product is communicated to the master as soon as it is completed. This results in faster completion of the full-computation as it allows to exploit partial computations completed by stragglers. Moreover, the multi-message approach makes finishing the task possible even if there are not as many available workers as the recovery threshold.

Another important parameter in distributed computation is the *upload cost*, which is defined as the number of bits sent from the master to each worker, or equivalently, the storage required per worker. As discussed in [9], simply assigning multiple sub-products to the workers using polynomial-type codes is not efficient in terms of the upload cost, as one matrix partition can only be used in the computation of one sub-product. To combat this limitation, product codes are considered in [6] for the multi-message distributed matrix multiplication problem. However, with product codes, sub-products are no longer fully one-to-any replaceable, which reduces the scheme's resource efficiency. To make the sub-products one-to-any replaceable, *bivariate polynomial codes* are introduced in [9], [10], which provides a better trade-off between the upload cost and expected computation time, by allowing a matrix partition to be used in the computation of several sub-tasks.

The second challenge we tackle in this paper is privacy. The multiplied matrices may contain sensitive information, and sharing these matrices even partially with the workers may cause a privacy breach. Moreover, in some settings, a number of workers can exchange information with each other to learn about the multiplied matrices. Such a collusion may result in leakage even if no information is revealed to individual workers. The first application of polynomial codes to privacy-preserving distributed matrix multiplication is presented in [11]. To hide the matrices from the workers, random matrix partitions are created, and linearly encoded together with the true matrix partitions using polynomial codes. The recovery threshold has been improved in subsequent works [12], [13], by carefully choosing the degrees of the encoding monomials so that the resultant decoding polynomial contains the minimum number of additional coefficients. In [14]–[16], lower recovery threshold values than [13] are obtained by

This work was partially funded by the European Research Council (ERC) through Starting Grant BEACON (no. 677854) and by the UK EPSRC (grant no. EP/T023600/1) under the CHIST-ERA program. The work of J. Gómez-Vilardebó was supported in part by the Catalan Government under Grant SGR2017-1479, and by the Spanish Government under Grant RTI2018-099722-B-100 (ARISTIDES).

using different matrix partitioning techniques and different choices of encoding polynomials, but this is achieved at the expense of a considerable increase in the upload cost. In [17], a novel coding approach for distributed matrix multiplication is proposed based on polynomial evaluation at the roots of unity in a finite field. It has constant time decoding complexity and a low recovery threshold compared to traditional polynomial-type coding approaches, but the sub-tasks are not one-to-any replaceable and its straggler mitigation capability is limited. In [18], a multi-message approach is proposed for private distributed matrix multiplication by using rateless codes. Computations are assigned adaptively in rounds, and in each round, workers are classified into clusters depending on their computation speeds. Results from a worker in a cluster are useful for decoding only if the results of all the sub-tasks assigned to that cluster and also to the fastest cluster are collected, making computations not one-to-any replaceable.

In this work, we propose a multi-message, straggler-resistant, private distributed matrix multiplication scheme based on bivariate Hermitian polynomial codes. Our scheme works effectively even with a small number of workers and under a limited upload cost budget. We show that, especially when the number of fast workers is limited, our proposed method outperforms other schemes in the literature in terms of the average computation time under a given upload cost budget. We also show that our scheme retains its low expected computation time under both homogeneous and heterogeneous computation speeds across the workers.

II. Problem Setting

We study distributed matrix multiplication with strict privacy requirements. The elements of our matrices are in a finite field $\mathbb{F}_q$, where q is a prime number determining the size of the finite field. There is a master node that can access to the statistically independent matrices $A \in \mathbb{F}_q^{r\times s}$ and $B \in \mathbb{F}_q^{s\times c}$, $r, s, c \in \mathbb{Z}^+$. The master offloads the multiplication of matrices A and B to N workers, which possibly have heterogeneous computation speeds and storage capacities. We do not assume any statistics are known about the computation speeds of the workers. To offload the computation to several workers, the master divides the full multiplication task into smaller sub-tasks and then collects the responses from the workers. To define these sub-tasks, the master partitions A into K sub-matrices as $A = \begin{bmatrix} A_1^T & A_2^T & \cdots & A_K^T \end{bmatrix}^T$, where $A_i \in \mathbb{F}_q^{\frac{r}{K}\times s}$, $\forall i \in [K] \triangleq \{1, 2, \ldots, K\}$, and B into L sub-matrices as $B = \begin{bmatrix} B_1 & B_2 & \cdots & B_L \end{bmatrix}$, where $B_j \in \mathbb{F}_q^{s\times \frac{c}{L}}$, $\forall j \in [L]$. The master sends coded versions, i.e., linear combinations, of these partitions to the workers. Assuming worker i can store $m_{A,i}$ partitions of A and $m_{B,i}$ partitions of B, the master sends coded partitions $\tilde{A}_{i,k}$ and $\tilde{B}_{i,l}$ to worker i, where $i \in [N]$, $k \in [m_{A,i}]$ and $l \in [m_{B,i}]$. For simplicity, we describe a static setting, in which all the coded matrices are sent to the workers before they start computations. However, in a more dynamic setting, in which matrix partitions are sent when they are needed, the required memory at workers could be made smaller. The basic assumption is that $m_{A,i}$ partitions of A and $m_{B,i}$ partitions of B are available for worker i at some point, and thus they could exploit them to extract information on the original matrices A and B. The workers multiply the received coded partitions of A and B in a way depending on the underlying coding scheme and send the result of each computation to the master as soon as it is finished. Once the master receives a number of computations equal to the recovery threshold, R_{th}, it can decode the desired multiplication AB.

In our threat model, all the workers are honest but curious. That is, they follow the protocol but they can use the received encoded matrices to gain information about the original matrices, A and B. We also assume that any T workers can collude, i.e., exchange information among themselves. Our privacy requirement is such that no T workers are allowed to gain any information about the content of the multiplied matrices in the information-theoretic sense.

III. Proposed Scheme

Our coding scheme is based on bivariate polynomial codes [9], [10]. Thanks to their lower upload cost, bivariate polynomial codes allow workers to complete more sub-tasks compared to their univariate counterparts under the same upload cost budget, which, in turn, improves the expected computation time and helps to satisfy the privacy requirements.

A. Encoding

In the proposed coding scheme, coded matrices are generated by evaluating the following polynomials and their derivatives:

$$A(x) = \sum_{i=1}^{K} A_i x^{i-1} + \sum_{i=1}^{T} R_i x^{K+i-1}, \tag{1}$$

$$B(x,y) = \sum_{i=1}^{L} B_i y^{i-1} + \sum_{i=1}^{T}\sum_{j=1}^{m} S_{i,j} x^{K+i-1} y^{j-1}, \tag{2}$$

where $m \leq L$ is the maximum number of sub-tasks any worker can complete. Matrices $R_i \in \mathbb{F}_q^{\frac{r}{K}\times s}$ and $S_{i,j} \in \mathbb{F}_q^{s\times \frac{c}{L}}$ are independent and uniform randomly generated from their corresponding domain for $i \in [T]$ and $j \in [m]$. For each worker i, the master evaluates $A(x)$ at x_i and the derivatives of $B(x,y)$ with respect to y up to the order $[m-1]$ at (x_i, y_i). We only require these evaluation points to be distinct. Thus, the master sends to worker i, $A(x_i)$ and $\mathcal{B}_i = \{B(x_i,y_i), \partial_1 B(x_i,y_i), \ldots, \partial_{m-1} B(x_i,y_i)\}$, where ∂_i is the i^{th} partial derivative with respect to y. Thus, we require $m_{A,i} = 1$ and $m_{B,i} = m$.

In (1) and (2), the role of R_i's and $S_{i,j}$'s is to mask the actual matrix partitions for privacy. The following theorem states that the evaluations of $A(x)$, $B(x,y)$ and its derivatives do not leak any information about A and B to any T colluding workers.

Theorem 1. *For the encoding scheme described above, we have*

$$I(A, B; \{(A(x_i), \mathcal{B}_i) \mid i \in \tilde{N}\}) = 0, \tag{3}$$

$\forall \tilde{N} \subset [N]$ *such that* $|\tilde{N}| = T$.

Proof. Since A and B are independent, we have

$$I(A, B; \{(A(x_i), \mathcal{B}_i) \mid i \in \tilde{N}\}) = \\ I(A; \{A(x_i) \mid i \in \tilde{N}\}) + I(B; \{\mathcal{B}_i \mid i \in \tilde{N}\}).$$

Moreover, every worker receives only one $A(x_i)$. Thus, in case of T colluding workers, the attackers can obtain at most T evaluations of $A(x)$. Since x_i's are distinct, only $T-1$ of R_i's can be eliminated. Thus, $I(A; \{A(x_i) \mid i \in \tilde{N}\}) = 0$ since R_i's are generated uniform randomly from $\mathbb{F}_q$. Similarly, the attackers can obtain at most mT evaluations of $B(x, y)$ and its y-directional derivatives in total. Since (x_i, y_i)'s are distinct for different i, and monomials $x^{K+i-1}y^{i-1}$ and their derivatives with respect to y up to the m^{th} order are linearly independent, the attackers can eliminate at most $mT - 1$ of $S_{i,j}$'s. Thus, $I(B; \{\mathcal{B}_i \mid i \in \tilde{N}\}) = 0$ since $S_{i,j}$'s are generated uniform randomly from $\mathbb{F}_q$. □

B. Computation

Worker i multiplies $A(x_i)$ and $\partial_{j-1}B(x_i, y_i)$ with the increasing order of $j \in [m]$. That is, j^{th} completed computation is $A(x_i)\partial_{j-1}B(x_i, y_i)$. As soon as each multiplication is completed, its result is communicated back to the master.

C. Decoding

After collecting sufficiently many computations from the workers, the master can interpolate $A(x)B(x, y)$. Note that, in our scheme, every computation is equally useful, i.e., the sub-tasks are one-to-any replaceable. In the following theorem, we give the recovery threshold expression, which specifies the minimum number of required computations and characterizes the probability of decoding failure, i.e., bivariate polynomial interpolation, due to the use of finite field.

Theorem 2. *Assume the evaluation points (x_i, y_i) are chosen uniform randomly over the elements of $\mathbb{F}_q$. If the number of computations of sub-tasks received from the workers, which obey the computation order specified in Subsection III-B is greater than the recovery threshold $R_{th} \triangleq (K+T)L + m(K+T-1)$, then with probability at least $1 - d/q$, the master can interpolate the unique polynomial $A(x)B(x, y)$, where*

$$d \triangleq \frac{m}{2}\left(3(K+T)^2 + m(K+T) - 8K - 6T - m + 3\right) \\ + \frac{(K+T)L}{2}(K+L+T-2). \quad (4)$$

We give the proof sketch of Theorem 2 in the Appendix. Theorem 2 says that we can make the probability of failure arbitrarily small by increasing the order q of the finite field.

IV. Discussion

The recovery threshold of our scheme is comparable to that of the multi-message extension of [13], in which each worker is assigned m computations. In this case, the number of evaluations of each encoding polynomial collectively obtained by T colluding workers is mT. Thus, the recovery threshold of such an extension is obtained by substituting mT for T in the recovery threshold expression of [13]. That yields $R_{th}^{GASP} =$

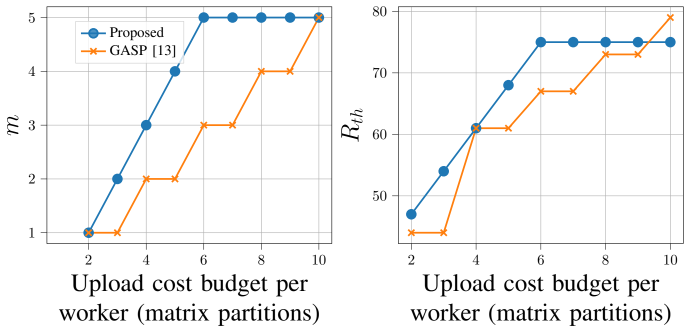

Fig. 1: Change of m with upload cost budget and the comparison of the recovery thresholds of the proposed scheme and the multi-message extension of [13], when $K = L = 5, T = 3$.

$$\begin{cases} KL + K + L & 1 = mT < L \leq K \\ KL + K + L + (mT)^2 + mT - 3 & 1 < mT < L \leq K \\ (K + mT)(L + 1) - 1 & L \leq mT < K \\ 2KL + 2mT - 1 & L \leq K \leq mT. \end{cases}$$

In multi-message schemes, using univariate polynomial codes, including [18] and the multi-message extension of [13], if a worker is assigned m sub-tasks, then m coded partitions of both A and B are required. Thus, the total upload cost for the univariate polynomial codes is $Nm\,(rs/K + sc/L)\log_2(q)$ bits. On the other hand, in the proposed scheme, we need one coded partition of A and m coded partitions of B. Thus, the upload cost of our scheme is $N\,(rs/K + msc/L)\log_2(q)$ bits, which is much less than that of univariate polynomial codes.

Comparing R_{th}^{GASP} and R_{th} of the proposed scheme for the same m value might be misleading since, under the same upload cost budget, these schemes might have different m values. Given an upload cost budget, we take the largest possible m since it is the limiting factor for the maximum number of computation a worker can provide. In Fig. 1, for $K = L = 5, T = 3$ we show how m changes with the upload cost budget, and we compare R_{th} and R_{th}^{GASP} under a fixed upload cost budget, which is given in the number of matrix partitions for simplicity, assuming the partitions of A and B have the same size. Observe that in the proposed scheme, since m cannot exceed $L = 5$, for the upload cost values greater than 6, the values of m and R_{th} stays the same.

The upload cost could be further improved by employing the schemes in [9], [10], in which for a worker to complete m computations, as few as $\sqrt{m}$ coded partitions of A and B may be enough. However, the extensions of these schemes to the private case have a large privacy overhead in the recovery threshold. For example, [9] originally has $R_{th} = KL$, but for its direct privacy extension, we have $R_{th} \approx (K + \sqrt{m}T)(L + \sqrt{m}T)$, which requires much larger computation time.

V. Simulation Results

In this section, we compare our work with the previous works in the literature [13], [18] in terms of average computa-

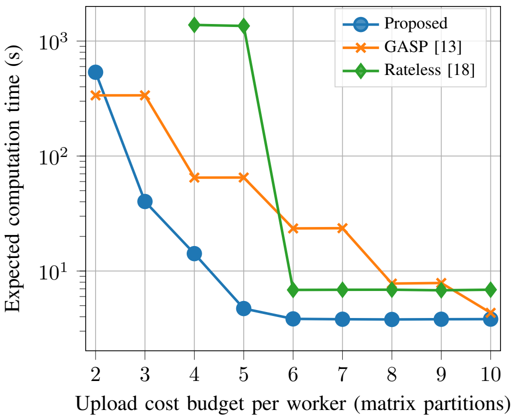

Fig. 2: Comparison of expected computation time as a function of the upload cost budget per worker when $K = L = 5$ with heterogeneous workers.

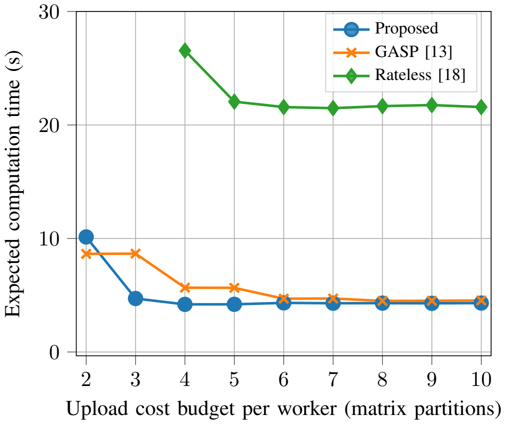

Fig. 3: Comparison of expected computation time as a function of the upload cost budget per worker when $K = L = 5$ with homogeneous workers.

tion time. We define the *computation time* as the time required by the workers to complete the number of computations specified by the recovery threshold. Our scenario consists of a limited number of workers and limited upload resources from the master to the workers.

Following the literature [1], [19], we assume that the time for a worker to finish one sub-task is distributed as a shifted exponential random variable with the density $\lambda e^{-\lambda(t-\nu)}$, where the scale parameter λ controls the speed of the worker and the shift parameter ν is the minimum time duration for a task to be completed. Smaller λ implies slower workers and thus more straggling.

In our experiments, all the workers have a common shift parameter of $\nu = 10/(KL)$ seconds. We assume $T = 3$. We consider both matrices A and B are divided into $K = L = 5$ partitions and assume that the partitions of matrices A and B have the same size, i.e., $\frac{r}{K} = \frac{c}{L}$.

We consider two scenarios, workers with heterogeneous and homogeneous computational speeds, respectively. In the heterogeneous case, we group the workers into three classes, each consisting of 17 workers, and the computation speed of the workers in each class is specified by $\lambda_1 = 10^{-1} \times KL$, $\lambda_2 = 10^{-3} \times KL$ and $\lambda_3 = 10^{-4} \times KL$, respectively. In the homogeneous case, all the workers have $\lambda = 10^{-2} \times KL$. Note that the coding scheme is agnostic to these values in both scenarios.

Since [13] was not originally proposed as a multi-message solution, we use its multi-message extension as described in Section IV. For [18], we do not limit the upload cost per worker but we limit the total upload cost since it allocates the computation load to workers adaptively in rounds. Moreover, since there are three groups of workers with different scale parameters, we use three clusters in the rounds after the first round. In the first round, we assign every worker one computation in accordance with the description in [18].

In Fig. 2 and Fig. 3, we plot the expected computation time versus the upload cost per worker for heterogeneous and homogeneous cases, respectively. For simplicity, the upload cost is given in terms of the maximum number of total matrix partitions that can be sent to each worker, instead of the number of bits. As observed in Fig. 2, in the heterogeneous case, even if the recovery threshold of the proposed scheme is close to that of GASP codes [13] (see Fig. 1), the proposed scheme's ability to generate more computations with the same upload cost budget, i.e. number of encoded matrix partitions, allows faster completion of the overall task. Rateless codes [18], on the other hand, perform very close to the proposed scheme when the upload cost budget is large, but for the smaller values, either it could not complete, e.g., upload cost budgets 2 and 3, or it took too long to complete the task. This is because the scheme assigns new sub-tasks whenever a cluster completes its assignment greedily. Thus, the upload cost budget is mostly invested in the fastest cluster in the heterogeneous setting. However, despite its speed, the fastest cluster does not provide many useful computations compared to the slower clusters.

On the other hand, for the homogeneous case, as observed in Fig. 3, the improvement of our scheme over GASP codes is limited. In this case, since workers' computation speeds are similar, faster workers do not compensate for slower ones. Still, we observe that due to the one-to-any replaceability of our scheme and GASP, both perform much better than rateless codes [18].

VI. Conclusion

We have proposed storage- and upload-cost-efficient bivariate Hermitian polynomial codes for straggler exploitation in private distributed matrix multiplication. Previous works usually assume the availability of at least as many workers as the recovery threshold, but if the number of workers is not sufficient, the multi-message approach can allow the completion of the task. Compared to prior work, the proposed coding scheme has lower upload cost and less storage requirement, making the assignment of several sub-tasks to each worker more practical. Thanks to these properties, the proposed bivariate polynomial codes improve the average computation time of the private distributed matrix multiplication, especially when the number of workers, the upload cost budget or the storage capacity is limited.

Appendix: Proof Sketch of Theorem 2

In Fig. 4, we visualize the degrees of the monomials of $A(x)B(x,y)$. We see that the number of monomials of $A(x)B(x,y)$ is $(K+T)L+m(K+T-1)$. We need to show that every possible combination of so many responses from the workers interpolates to a unique polynomial, implying $(K+T)L+m(K+T-1)$ is the recovery threshold.

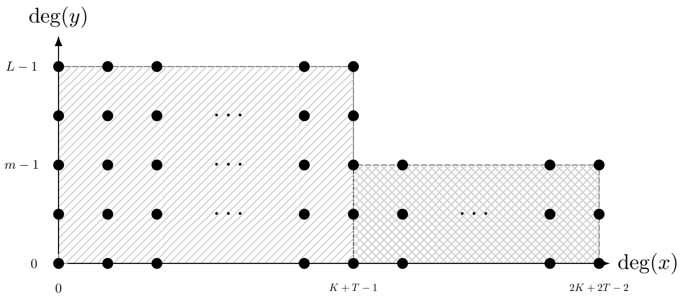

Fig. 4: The visualization of the degrees of the monomials of $A(x)B(x,y)$ in the $\deg(x)-\deg(y)$ plane.

Definition 1. Bivariate polynomial interpolation problem can be formulated as solving a linear system of equations, whose unknowns are the coefficients of $A(x)B(x,y)$ and whose coefficient matrix consists of the monomials of $A(x)B(x,y)$ and their derivatives with respect to y, which are evaluated at the evaluation points of responded workers. We refer to this coefficient matrix as **interpolation matrix** and denote it by M. For example, when $K=L=2, m=2, T=1, N=4$, M would be as follows.

$$\begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 & x_1^4 & y_1 & x_1y_1 & x_1^2y_1 & x_1^3y_1 & x_1^4y_1 \\ 0 & 0 & 0 & 0 & 0 & 1 & x_1 & x_1^2 & x_1^3 & x_1^4 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_4 & x_4^2 & x_4^3 & x_4^4 & y_4 & x_4y_4 & x_4^2y_4 & x_4^3y_4 & x_4^4y_4 \\ 0 & 0 & 0 & 0 & 0 & 1 & x_4 & x_4^2 & x_4^3 & x_4^4 \end{bmatrix}$$

Observe that for worker 1, the first and the second rows correspond to $A(x)B(x,y)$ and $A(x)\partial_1 B(x,y)$, evaluated at (x_1,y_1), respectively.

The problem of showing that any R_{th} responses from the workers interpolates to a unique polynomial is equivalent to showing that the corresponding interpolation matrix is non-singular. The theorem claims that this is the case with high probability. First, we need to show that there exist some evaluation points for which the determinant of the interpolation matrix is not zero. That is equivalent to showing that $\det(M)$ is not the zero polynomial of the evaluation points.

In [20] and [10], such a result for the same type of interpolation matrices is shown for the real field $\mathbb{R}$. We omit it here for space restrictions. The result in [20] and [10] is based on Taylor series expansion, which is also applicable in $\mathbb{F}_q$, if the degree of the polynomial $A(x)B(x,y)$ is smaller than q. This can be guaranteed by choosing a large q. For a discussion of the applicability of Taylor series expansion in finite fields, see [21], and [22]. From these results, we can conclude that $\det(M)$ is not the zero polynomial for large enough q. Next, we need to find the upper bound on the probability $\det(M)=0$, when the evaluation points are sampled uniform randomly from $\mathbb{F}_q$.

Lemma 1. [23, Lemma 1] *Assume P is a non-zero, v-variate polynomial of variables $\alpha_i, i \in [v]$. Let d_1 be the degree of α_1 in $P(\alpha_1,\dots,\alpha_v)$, and $P_2(\alpha_2,\dots,\alpha_v)$ be the coefficient of $\alpha_1^{d_1}$in $P(\alpha_1,\dots,\alpha_v)$. Inductively, let d_j be the degree of α_j in $P_j(\alpha_j,\dots,\alpha_v)$ and $P_{j+1}(\alpha_{j+1},\dots,\alpha_v)$ be the coefficient of α_j in $P_j(\alpha_j,\dots,\alpha_v)$. Let S_j be a set of elements from a field $\mathbb{F}$, from which the coefficients of P are chosen. Then, in the Cartesian product set $S_1\times S_2\times\cdots\times S_v$, $P(\alpha_1,\dots,\alpha_v)$ has at most $|S_1\times S_2\times\cdots\times S_v|\left(\frac{d_1}{|S_1|}+\frac{d_2}{|S_2|}+\cdots+\frac{d_v}{|S_v|}\right)$ zeros.*

In our case, since the elements of M are the monomials of $A(x)B(x,y)$ and their derivatives with respect to y, evaluated at some (x_i,y_i), $\det(M)$ is a multivariate polynomial of the evaluation points (x_i,y_i). Thus, v is the number of different evaluation points in M. We choose the evaluation points from the whole field $\mathbb{F}_q$. Thus, $S_j=\mathbb{F}_q$ and $|S_j|=q, \forall j\in[v]$, and $|S_1\times S_2\times\cdots\times S_v|=q^v$. Then, the number of zeros of $\det(M)$ is at most $q^{v-1}(d_1+d_2+\cdots+d_v)$. If we sample the evaluation points uniform randomly, then the probability that $\det(M)=0$ is $(d_1+d_2+\cdots+d_v)/q$, since we sample a v-tuple of evaluation points from $S_1\times S_2\times\cdots\times S_v$. To find $d_1+d_2+\cdots+d_v$, we resort to the definition of determinant, that is $\det(M)=\sum_{i=1}^{R_{th}}(-1)^{1+i}m_{1,i}M_{1,i}$, where $m_{1,i}$ is the element of M at row 1 and column i and $M_{1,i}$ is the minor of M when row 1 and column i are removed [24, Corollary 7.22]. Thus, to identify the coefficients in Lemma 1, in the first row of M, we start with the monomial with the largest degree. Assuming the monomials are placed in an increasing order of their degrees, the largest degree monomial is at column R_{th}. If that monomial is univariate, then d_1 is the degree of the monomial and the coefficient of $\alpha_1^{d_1}$ is $P_2(x_2,\dots,x_v)=\det(M_{1,1})$. If the monomial is bivariate, then we take the degree of the corresponding evaluation of x, i.e., α_1, as d_1, and the degree of the corresponding evaluation of y, i.e., α_2, as d_2. In this case, the coefficient of α^{d_2} is $P_3(\alpha_3,\dots,\alpha_v)=\det(M_{1,1})$. Next, we take $M_{1,1}$, and repeat the same procedure. We do so until we reach a monomial of degree zero. In this procedure since we visit all the monomials of $A(x)B(x,y)$ evaluated at different evaluation points, i.e., α_i's, the sum $d_1+d_2+\cdots+d_v$ becomes the sum of degrees of all the monomials of $A(x)B(x,y)$. The next lemma helps us in computing this.

Lemma 2. *Consider the polynomial $P(x,y)=\sum_{i=0}^{a}\sum_{j=0}^{b}c_{ij}x^iy^j$, where $c_{i,j}$'s are scalars. The sum of degrees of all the monomials of $P(x,y)$ is given by $\xi(a,b)\triangleq\frac{a(a+1)}{2}(b+1)+\frac{b(b+1)}{2}(a+1)$.*

The proof of Lemma 2 is based on Gauss's trick, and is omitted due to space restrictions. By using Lemma 2, we can find the sum of monomial degrees in the diagonally shaded rectangle and the rectangle shaded by crosshatches in Fig. 4, separately, and by summing them we find $d_1+d_2+\cdots+d_v$ to be equal to (4), which concludes the proof.

References

[1] 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.

[2] Q. Yu, M. Maddah-Ali, and S. Avestimehr, "Polynomial codes: an optimal design for high-dimensional coded matrix multiplication," in *Advances in Neural Information Processing Systems*, 2017, pp. 4403–4413.

[3] S. Dutta, M. Fahim, F. Haddadpour, H. Jeong, V. Cadambe, and P. Grover, "On the optimal recovery threshold of coded matrix multiplication," *IEEE Transactions on Information Theory*, vol. 66, no. 1, pp. 278–301, 2019.

[4] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding," in *IEEE International Symposium on Information Theory*, 2018.

[5] Z. Jia and S. A. Jafar, "Cross subspace alignment codes for coded distributed batch computation," *arXiv*, pp. arXiv–1909, 2019.

[6] S. Kiani, N. Ferdinand, and S. C. Draper, "Exploitation of stragglers in coded computation," in *IEEE International Symposium on Information Theory*, 2018.

[7] M. M. Amiri and D. Gündüz, "Computation scheduling for distributed machine learning with straggling workers," *IEEE Transactions on Signal Processing*, vol. 67, no. 24, pp. 6270–6284, 2019.

[8] E. Ozfatura, S. Ulukus, and D. Gündüz, "Straggler-aware distributed learning: Communication–computation latency trade-off," *Entropy*, vol. 22, no. 5, p. 544, 2020.

[9] B. Hasircioglu, J. Gomez-Vilardebo, and D. Gunduz, "Bivariate polynomial coding for straggler exploitation with heterogeneous workers," *IEEE International Symposium on Information Theory*, 2020.

[10] B. Hasircioglu, J. Gomez-Vilardebo, and D. Gunduz, "Bivariate hermitian polynomial coding for efficient distributed matrix multiplication," in *IEEE Global Communications Conference (GLOBECOM)*, 2020.

[11] 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.

[12] J. Kakar, S. Ebadifar, and A. Sezgin, "Rate-efficiency and straggler-robustness through partition in distributed two-sided secure matrix computation," *arXiv preprint arXiv:1810.13006*, 2018.

[13] 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.

[14] M. Aliasgari, O. Simeone, and J. Kliewer, "Private and secure distributed matrix multiplication with flexible communication load," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2722–2734, 2020.

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

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

[17] N. Mital, C. Ling, and D. Gunduz, "Secure distributed matrix computation with discrete fourier transform," *arXiv preprint arXiv:2007.03972*, 2020.

[18] R. Bitar, M. Xhemrishi, and A. Wachter-Zeh, "Rateless codes for private distributed matrix-matrix multiplication," *arXiv preprint arXiv:2004.12925*, 2020.

[19] G. Liang and U. C. Kozat, "Tofec: Achieving optimal throughput-delay trade-off of cloud storage using erasure codes," in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE, 2014, pp. 826–834.

[20] B. Hasircioglu, J. Gomez-Vilardebo, and D. Gunduz, "Bivariate polynomial coding for exploiting stragglers in heterogeneous coded computing systems," *arXiv preprint arXiv:2001.07227*, 2020.

[21] K. Hoffman and R. Kunze, "Linear algebra," *Englewood Cliffs, New Jersey*, 1971.

[22] F. Fontein, "The hasse derivative," Aug 2009. [Online]. Available: https://math.fontein.de/2009/08/12/the-hasse-derivative/

[23] J. T. Schwartz, "Fast probabilistic algorithms for verification of polynomial identities," *Journal of the ACM (JACM)*, vol. 27, no. 4, pp. 701–717, 1980.

[24] J. Liesen and V. Mehrmann, *Linear algebra*, ser. Springer undergraduate mathematics series.