entropy

Article

Straggler- and Adversary-Tolerant Secure Distributed Matrix Multiplication Using Polynomial Codes

Eimear Byrne [1], Oliver W. Gnilke [2] and Jörg Kliewer [3,*]

[1] School of Mathematics and Statistics, University College Dublin, D04 V1W8 Dublin, Ireland
[2] Department of Mathematical Sciences, Aalborg University, 9220 Aalborg, Danmark
[3] Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07410, USA
* Correspondence: jkliewer@njit.edu; Tel.: +1-973-596-3519

Abstract: Large matrix multiplications commonly take place in large-scale machine-learning applications. Often, the sheer size of these matrices prevent carrying out the multiplication at a single server. Therefore, these operations are typically offloaded to a distributed computing platform with a master server and a large amount of workers in the cloud, operating in parallel. For such distributed platforms, it has been recently shown that coding over the input data matrices can reduce the computational delay by introducing a tolerance against straggling workers, i.e., workers for which execution time significantly lags with respect to the average. In addition to exact recovery, we impose a security constraint on both matrices to be multiplied. Specifically, we assume that workers can collude and eavesdrop on the content of these matrices. For this problem, we introduce a new class of polynomial codes with fewer non-zero coefficients than the degree $+1$. We provide closed-form expressions for the recovery threshold and show that our construction improves the recovery threshold of existing schemes in the literature, in particular for larger matrix dimensions and a moderate to large number of colluding workers. In the absence of any security constraints, we show that our construction is optimal in terms of recovery threshold.

Keywords: distributed computation; matrix multiplication; distributed learning; information theoretic security; polynomial codes

check for updates

Citation: Byrne, E.; Gnilke, O.W.; Kliewer, J. Straggler- and Adversary-Tolerant Secure Distributed Matrix Multiplication Using Polynomial Codes. *Entropy* **2023**, *25*, 266. https://doi.org/10.3390/e25020266

Academic Editor: Syed A. Jafar

Received: 1 November 2022
Revised: 16 January 2023
Accepted: 20 January 2023
Published: 31 January 2023

1. Introduction

Recently, tensor operations have emerged as an important ingredient of many signal processing and machine learning applications [1]. These operations are typically complex due to the large size of the associated tensors. Therefore, in the interest of a low execution time, such computations are often performed in a distributed fashion and outsourced to a cloud of multiple workers that operate in parallel over the distributed data set. These workers in many cases consist of commercial off-the-shelf servers that are characterized by failures and varying execution times. Such straggling servers are handled by state-of-the art cloud computation platforms via a repetition of the computation task at hand. However, recent work has shown that encoding the input data may help alleviate the straggler problem and thus reduce the computation latency, which mainly depends on the amount of stragglers present in the cloud computing environment; see [2,3]. More generally, it has been shown that coding can control the trade-off between computational delay and communication load between workers and master server [3–6]. In addition, the workers in the cloud may not be trustworthy, so the input and output of the partial computations need to be protected against unauthorized access. To this end, it has been shown that stochastic coding can help keep both input and output data secure from eavesdropping and colluding workers (see, for example, [7–14]).

In this work, we focus on the canonical problem of distributing the multiplication of two matrices A and B, i.e., $C = AB$, whose content should be kept secret from a prescribed number of colluding workers in the cloud. Our goal is to minimize the number of workers from which the partial result must be downloaded, the so-called *recovery threshold*, to recover the correct matrix product C.

Coded matrix computation was first addressed in the non-secure case by applying separate MDS codes to encode the two matrices [3]. In [5], polynomial codes have been introduced, which improves on the recovery threshold of [3]. The recovery threshold was further improved by the so-called MatDot and PolyDot codes [15,16] at the expense of a larger download rate. In particular, PolyDot codes allow a flexible trade-off between the recovery threshold and the download rate, depending on the application at hand.

In [17,18] two different schemes are presented, an explicit scheme that improves on the recovery thereshold of PolyDot codes and a construction based on the tensor rank of matrix multiplication, which is optimal up to a factor of 2. In [19] a new construction for private and secure matrix multiplication is proposed based on entangled polynomial codes, which allows for a flexible trade-off between the upload rate and the download rate (equivalently, the recovery threshold). For small numbers of stragglers [20] constructs schemes that outperform the entangled polynomial scheme. Recently, several attempts have been made to design coding schemes to further reduce upload and download rates, the recovery threshold, and computational complexity for both workers and server (see, for example, [21–27]). For example, in [21], bivariate polynomial codes were used to reduce the recovery threshold in specific cases. In [22], the authors considered new schemes for the private and secure case which outperform [19] for specific parameter regions. The work in [23] considered distributed storage repair codes, so-called field-trace polynomial codes, to reduce the download rate for specific partitions of matrices A and B. Very recently, the authors in [24] proposed a black-box coding scheme based on star products, which subsumes several existing works as special cases. In [25], a discrete Fourier transform-based scheme with low upload rates and encoding complexity is proposed. The work in [26] focused on selecting the evaluation points for the polynomial codes, providing a better upload rate than [9], but worse than [25].

In the following, we propose a new scheme for secure matrix multiplication, which provides explicit evaluation points for the polynomial codes, but unlike the work in [26], is also able to tolerate stragglers. Specifically, we exploit gaps in the underlying polynomial code. This is motivated by the observation that the recovery threshold can be improved by selecting the number of evaluation points to be equal to the number of only the *non-zero* coefficients in the polynomial [9,19]. In addition, selecting dedicated evaluation points has the advantage that the condition for security against colluding workers is automatically satisfied (see, for example, condition C2 in [27]). As such, our approach is able to provide a constructive scheme with provable security guarantees. Further, our coding scheme provides an advantage in terms of download rate in some cases, and is both straggler-tolerant and robust against Byzantine attacks on the workers.

This paper is organized as follows. In Section 2, the problem statement and the background is highlighted. Section 3 discusses design and properties of our proposed scheme and provides performance guarantees with respect to the number of helper nodes needed for recovery, security, straggler tolerance and under Byzantine attacks. Section 4 extends the scheme of Section 4 by introducing gaps into the code polynomials and by studying its properties. Finally, Section 5 presents numerical results and comparisons with state-of-the-art schemes from the literature.

2. Problem Statement and Background

Let A and B be a pair of matrices over the finite field $\mathbb{F}_q$, whose product is well defined. We consider the problem of computing the product $C = AB$. The computation will be distributed among a number of helper nodes, each of which will execute a portion of the total calculation. We also assume that the user wishes to hide the data contained in the matrices A and B and that up to T honest but curious helper nodes may collude to deduce information about the contents of A and B. To divide the work among the helper nodes, the matrices A and B are each divided into KM and ML blocks, respectively, of compatible dimensions, say $a \times r$ and $r \times b$. The matrices are also assumed to have independent and identically distributed uniformly distributed entries from a sufficiently large field of cardinality $q > N$, where N denotes the number of servers to be employed (in fact, we will require q to exceed the degree of a polynomial $P(x)Q(x)$, central to this scheme). Hence, for given matrix partition of A and B according to

$$A = \begin{bmatrix} A_{1,1} & \cdots & A_{1,M} \\ \vdots & \ddots & \vdots \\ A_{K,1} & \cdots & A_{K,M} \end{bmatrix}, \quad B = \begin{bmatrix} B_{1,1} & \cdots & B_{1,L} \\ \vdots & \ddots & \vdots \\ B_{M,1} & \cdots & B_{M,L} \end{bmatrix},$$

we obtain

$$C = AB = \begin{bmatrix} C_{1,1} & \cdots & C_{1,L} \\ \vdots & \ddots & \vdots \\ C_{K,1} & \cdots & C_{K,L} \end{bmatrix} \text{ where } C_{i,j} = \sum_{m=1}^{M} A_{i,m} B_{m,j}.$$

The system model is displayed in Figure 1. We consider a distributed computing system with a master server and N helper nodes or workers. The master server is interested in computing the product $C = AB$. In Figure 1, the worker receives matrices A and B and T random uniformly independent and identically distributed matrices of size $R_t \in \mathbb{F}_q^{a\times r}$ and $S_t \in \mathbb{F}^{r\times b}$ for $t \in [T]$. To keep the data secure and to leverage possible computational redundancy at the workers, the server sends encoded versions of the input matrices to the workers. This security constraint imposes the mutual information condition

$$I(A_{\mathcal{T}}, B_{\mathcal{T}}; A, B) = 0 \tag{1}$$

between the pair (A, B) and their encodings $(A_{\mathcal{T}}, B_{\mathcal{T}})$ for all subsets $\mathcal{T} \subset [N]$ of maximum cardinality T. The server generates a polynomial representation of A and R_t by constructing a polynomial $P(x) \in \mathbb{F}_q^{a\times r}[x]$. Likewise, a polynomial representation of B and Q_t results in a polynomial $Q(x) \in \mathbb{F}_q^{r\times b}[x]$. The polynomial encodings that the p-th worker receives comprise the two polynomial evaluations $P(\alpha_p)$ and $Q(\alpha_p)$, for distinct evaluation points $\alpha_p \in \mathbb{F}_q$ with $p \in [N]$. It then computes the matrix product $P(\alpha_p)Q(\alpha_p)$ and sends it back to the server. The server collects a subset of $N_R \le N$ outputs from the workers as defined by the evaluation points in the subset $\{P(\alpha_p)Q(\alpha_p)\}_{p\in\mathcal{N}_R}$ with $|\mathcal{N}_R| = N_R$. The size of the smallest possible subset N_R for which perfect recovery is obtained, i.e.,

$$H(AB|\{P(\alpha_p)Q(\alpha_p) : p \in \mathcal{N}_R\}) = 0, \tag{2}$$

where H denoted the entropy function, is defined as the *recovery threshold*. The server then interpolates the underlying polynomial such that the correct product $C = AB$ can be assembled from a combination of the interpolated polynomial coefficients $C_{i,j}$ (see Section 3 for details).

We further define the *upload rate* R_u per worker as the sum of the dimensions of $P(\alpha_p)$ and $Q(\alpha_p)$, i.e., $R_u = (a+b)r$ field elements of $\mathbb{F}_q$. Likewise, the *download rate* or *communication load* R_d is defined as the total number of field elements to be downloaded from the workers such that (2) is satisfied, i.e., $R_d = abN_R$.

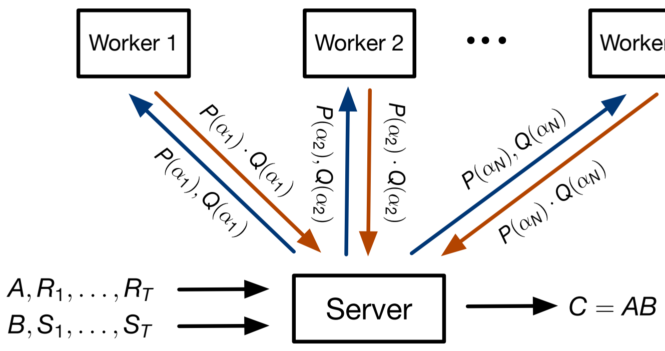
Worker 1
Worker 2
Worker N
$P(\alpha_1) \cdot Q(\alpha_1)$
$P(\alpha_1), Q(\alpha_1)$
$P(\alpha_2), Q(\alpha_2)$
$P(\alpha_2) \cdot Q(\alpha_2)$
$P(\alpha_N), Q(\alpha_N)$
$P(\alpha_N) \cdot Q(\alpha_N)$
$A, R_1, \ldots, R_T$
$B, S_1, \ldots, S_T$
Server
$C = AB$

Figure 1. System model for secure matrix multiplication.

Notation. For the remainder, we fix A, B, C to be matrices over $\mathbb{F}_q$ such that $C = AB$, and we fix K, M, L, a, b, r to be the integers as defined above. We define $[n] := \{1, \ldots, n\}$ for any positive integer n. For each $k \in [K], \ell \in [L]$, and $m \in [M]$, we write $A_{k,m}$, $B_{m,\ell}$, and $C_{k,\ell}$ to denote the (k, m), (m, ℓ), and (k, ℓ) blocks of A, B, and C, respectively. The transpose of a matrix Z is denoted by Z^t.

3. Proposed Scheme

The scheme we propose uses a similar approach to the schemes in [9,19,27]. We will begin with the choices for exponents in $P(x)$ and $Q(x)$ and show that the desired blocks of C appear as coefficients of the product PQ. We discuss the maximum possible degree of PQ since it gives us an upper bound on the necessary evaluations, and hence workers, needed to interpolate PQ. In Section 3.3, we give explicit criteria for choices of evaluation points and prove that the scheme protects against collusion of up to T servers. Section 3.4 discusses the option to query additional servers to provide resilience against stragglers and Byzantine servers.

Section 4 uses ideas from the GASP scheme [9] to reduce the recovery threshold by examining how many coefficients in the product are already known to be zero.

3.1. Choice of Exponents and Maximal Degree

We propose the following scheme to outsource the computation among the worker servers. The model will incorporate methods to secure the privacy of the data held by the matrices A, B, and C.

Let $D := M + 2$. For the given A and B, we define the polynomials:

$$\bar{P}(x) := \sum_{k=1}^{K} x^{D(k-1)} \sum_{m=1}^{M} x^m A_{k,m} \quad \text{and} \quad \bar{Q}(x) := \sum_{\ell=1}^{L} x^{DK(\ell-1)} \sum_{m=1}^{M} x^{M+1-m} B_{m,\ell}.$$

We now define polynomials

$$P(x) := \bar{P}(x) + R(x) \quad \text{and} \quad Q(x) := \bar{Q}(x) + S(x),$$

where and $R(x), S(x)$ are a pair of matrix polynomials:

$$R(x) := \sum_{t=1}^{T} x^{D(t-1)} R_t \quad \text{and} \quad S(x) := \sum_{t=1}^{T} x^{D(t-1)} S_t,$$

whose coefficients are $a \times r$ and $r \times b$ matrices over $\mathbb{F}_q$, respectively, chosen uniformly at random.

In the next theorem, we show that the desired matrices $C_{k,\ell}$ appear as coefficients of the product PQ and can hence be retrieved by inspection of this product.

Theorem 1. *For each pair $(k, \ell) \in [K] \times [L]$, the block $C_{k,\ell}$ arising in the product $C = AB$ appears as the coefficient of $x^{D((k-1)+K(\ell-1))+M+1}$ in the product PQ.*

Proof. We calculate the product

$$\begin{aligned}PQ =&\bar{P}\bar{Q}+\bar{P}S+R\bar{Q}+RS\\ =&\sum_{k=1}^{K}\sum_{\ell=1}^{L}x^{D((k-1)+K(\ell-1))}\sum_{m=1}^{M}\sum_{m'=1}^{M}A_{k,m}B_{m',\ell}x^{M+1+m-m'}\\ &+\sum_{k=1}^{K}\sum_{t'=1}^{T}x^{D(k+t'-2)}\sum_{m=1}^{M}A_{k,m}S_{t'}x^{m}\\ &+\sum_{\ell=1}^{L}\sum_{t=1}^{T}x^{D(K(l-1)+(t-1))}\sum_{m'=1}^{M}R_tB_{m',\ell}x^{M+1-m'}\\ &+\sum_{t=1}^{T}\sum_{t'=1}^{T}R_tS_{t'}x^{D(t+t'-2)}.\end{aligned}$$

Consider the exponents modulo D. The first term in the sum of terms above is the product $\bar{P}\bar{Q}$. Any of the exponents of x in this term are equal to $D-1\equiv M+1 \mod D$ if and only if $m=m'$, in which case its corresponding coefficient is $C_{k,\ell}$. In particular, the matrix block $C_{k,\ell}$ appears in the product $\bar{P}\bar{Q}$ as the coefficient of $x^{D((k-1)+K(\ell-1))+M+1}$.

We claim that no other exponent of x in $PQ-\bar{P}\bar{Q}$ is equal to $M+1 \mod D$, from which the result will follow. Observe that the exponents in the second and third term of the product (i.e. those of $\bar{P}S+R\bar{Q}$) are all between 1 and M modulo D, while every exponent of x in the fourth term, which is RS, is a multiple of D. □

In order to retrieve the polynomial PQ, we may evaluate P and Q at a number of distinct values $\alpha_1,\dots,\alpha_{N+1}$ in $\mathbb{F}_q^{\times}$. The values $P(\alpha_i)$ and $Q(\alpha_i)$ are found at a cost of zero non-scalar operations. Define

$$V(\alpha_1,\dots,\alpha_{N+1}):=\begin{pmatrix}1 & \alpha_1 & \alpha_1^2 & \cdots & \alpha_1^N\\ 1 & \alpha_2 & \alpha_2^2 & \cdots & \alpha_2^N\\ \vdots & \ddots & \vdots & & \\ 1 & \alpha_N & \alpha_N^2 & \cdots & \alpha_N^N\\ 1 & \alpha_{N+1} & \alpha_{N+1}^2 & \cdots & \alpha_{N+1}^N\end{pmatrix}.$$

The (i,j)-entries of the coefficients of $PQ\in\mathbb{F}_q^{a\times b}[x]$ can be retrieved by computing the product

$$V(\alpha_1,\dots,\alpha_{N+1})^{-1}((P(\alpha_1)Q(\alpha_1))_{i,j},\dots,(P(\alpha_{N+1})Q(\alpha_{N+1}))_{i,j})^t,$$

if the degree of PQ is at most N. Since this computation involves only $\mathbb{F}_q$-linear computations, the total non-scalar cost is the total cost of performing the $N+1$ matrix products $P(\alpha_i)Q(\alpha_i)$. In the distributed computation scheme as shown in Figure 1, the server uploads each pair of evaluations $P(\alpha_i),Q(\alpha_i)$ to the i-th worker node, which then computes the product $P(\alpha_i)Q(\alpha_i)$ and returns it to the server.

In this approach to reconstructing PQ, we require the participation of $N+1$ worker nodes, where N is the degree of PQ. For this reason, we study this degree. Since

$$\deg(PQ)\le\max(\deg(\bar{P}\bar{Q}),\deg(\bar{P}S),\deg(R\bar{Q})\deg(RS)),$$

we have the following result, wherein each of the values $N_1(K,L,M;T)$ to $N_4(K,L,M;T)$ correspond to the maximum possible degrees of $\bar{P}\bar{Q},\bar{P}S,R\bar{Q}$, and RS, respectively. We write $N(A,B;K,L,M;T)$ to denote the maximum possible degree of the polynomial PQ, as the A,B,R,S range over all possible matrices of the stated sizes.

Proposition 1. *The degree of PQ is upper bounded by $N(A,B;K,L,M;T)$, where*

$$N(A,B;K,L,M;T) = \max \begin{cases} N_1(K,L,M;T) := D(KL-1) + 2M & (3) \\ N_2(K,L,M;T) := D(K+T-2) + M & (4) \\ N_3(K,L,M;T) := D(K(L-1)+T-1) + M & (5) \\ N_4(K,L,M;T) := 2D(T-1) & (6) \end{cases}$$

Proposition 2. *The following are equivalent.*

1. $T > K$,
2. $N_3(K,L,M;T) > N_1(K,L,M;T)$,
3. $N_4(K,L,M;T) > N_2(K,L,M;T)$.

Proof. First note that $T > K \Leftrightarrow T - K \geq 1$ and that $1 = \lceil \frac{M}{D} \rceil > \frac{M}{D}$. Since $T - K$ is an integer, we thus have that the following inequalities are equivalent to $T > K$:

$$T - K > \frac{M}{D},$$
$$D(T-K) > M,$$
$$D(K(L-1)+T-1) + M > D(KL-1) + 2M.$$

This shows that $N_3(K,L,M;T) > N_1(K,L,M;T)$ if and only if $T > K$. Similarly, using the 2nd and 3rd inequalities just above, we have

$$\begin{aligned} T > K &\Leftrightarrow DT > DK + M, \\ &\Leftrightarrow 2D(T-1) > D(T+K-2) + M, \end{aligned}$$

from which we see that $N_4(K,L,M;T) > N_2(K,L,M;T)$ if and only if $T > K$. □

Proposition 3. *The following are equivalent.*

1. $T > K(L-1) + 1$,
2. $N_4(K,L,M;T) > N_3(K,L,M;T)$,
3. $N_2(K,L,M;T) > N_1(K,L,M;T)$.

Proof. We have the following inequalities:

$$\begin{aligned} T > K(L-1) + 1 &\Leftrightarrow T - K(L-1) - 1 \geq 1 > \frac{M}{D}, \\ &\Leftrightarrow D(T - K(L-1) - 1) > M, \\ &\Leftrightarrow D(2T-2) > D(K(L-1)+T-1) + M, \end{aligned}$$

from which we deduce that $N_4(K,L,M;T) > N_3(K,L,M;T)$. We now show that $N_2(K,L,M;T) > N_1(K,L,M;T)$. We have:

$$\begin{aligned} T > K(L-1) + 1 &\Leftrightarrow D(T - K(L-1) - 1) > M, \\ &\Leftrightarrow D(K+T-2) + M > D(KL-1) + 2M. \end{aligned}$$

□

We tabulate (see Table 1) the value of $N(K,L,M;T)$ based on the observations of Propositions 2 and 3.

Table 1. Summary table of maximal degree of PQ.

	$\boldsymbol{T > K(L-1)+1}$		$\boldsymbol{T \leq K(L-1)+1}$	
$T > K$	$2D(T-1)$	(6)	$D(K(L-1)+T-1)+M$	(5)
$T \leq K$	$D(K+T-2)+M$	(4)	$D(KL-1)+2M$	(3)

3.2. *AB versus B^tA^t*

We compare the recovery threshold cost of calculating B^tA^t rather than AB. It can be shown that it is always better to calculate AB whenever $K \geq L$. That is, we show that $N(A,B;K,L,M;T) \leq N(B^t,A^t;L,K,M;T)$ for $K \geq L$. We consider all possible cases for the maximal degree in the following two theorems and remarks.

Theorem 2. *1. Let $T > K,L$. Suppose that $T < K(L-1)+1$ and $T < L(K-1)+1$. We have that*

$$N(A,B;K,L,M;T) = N_3(K,L,M;T) < N_3(L,K,M;T) = N(B^t,A^t;L,K,M;T),$$

if and only if $L < K$.

2. *Let $K \geq T > L$. Suppose that $T < K(L-1)+1$ and $T < L(K-1)+1$. We have that*

$$N(A,B;K,L,M;T) = N_1(K,L,M;T) < N_3(L,K,M;T) = N(B^t,A^t;L,K,M;T).$$

3. *Let $T > L,K$ and suppose that $L(K-1)+1 \geq T > K(L-1)+1$. We have that*

$$N(A,B;K,L,M;T) = N_4(K,L,M;T) < N_3(L,K,M;T) = N(B^t,A^t;L,K,M;T).$$

4. *Let $T > K \geq L$ and suppose that $T > L(K-1)+1$. We have that*

$$N(A,B;K,L,M;T) = N_4(K,L,M;T) = N_4(L,K,M;T) = N(B^t,A^t;L,K,M;T).$$

5. *Let $T \leq L \leq K$ and suppose that $T \leq K(L-1)+1$. We have that*

$$N(A,B;K,L,M;T) = N_1(K,L,M;T) = N_1(L,K,M;T) = N(B^t,A^t;L,K,M;T).$$

Proof. 1. Since $T > K$, and $T < K(L-1)+1$ by Propositions 2 and 3 we have that

$$N_3(K,L,M;T) > N_4(K,L,M;T) > N_2(K,L,M;T), N_1(K,L,M;T)$$

and so $N(A,B;K,L,M;T) = N_3(K,L,M;T)$.
Similarly, since $T > L$, and $T < L(K-1)+1$, we have that $N(B^t,A^t;L,K,M;T) = N_3(L,K,M;T)$. Clearly, $L < K$ if and only if:

$$\begin{aligned} N_3(K,L,M;T) &= D(K(L-1)+T-1)+M \\ &< D(L(K-1)+T-1)+M = N_3(L,K,M;T). \end{aligned}$$

2. By Propositions 2 and 3, the assumptions $K \geq T$ and $T < K(L-1)+1$ imply that $N(A,B;K,L,M;T) = N_1(K,L,M;T)$, while the assumptions $T > L$ and $T < L(K-1)+1$ yield that $N(B^t,A^t;K,L,M;T) = N_3(L,K,M;T)$.
Clearly, since $T > L$, we have $M < D(T-L)$ and

$$N_1(K,L,M;T) = D(KL-1)+2M < D(L(K-1)+T-1)+M = N_3(L,K,M;T).$$

3. From the given assumptions, by Propositions 2 and 3, we have $N(A, B; K, L, M; T) = N_4(K, L, M; T)$ and $N(B^t, A^t; L, K, M; T) = N_3(L, K, M; T)$. Since $L(K-1)+1 \geq T$, as in the proof of Proposition 3, we have

$$N_4(K, L, M; T) = 2D(T-1) = N_4(L, K, M; T) \leq N_3(L, K, M; T).$$

4. For the given assumptions the statement follows immediately from Propositions 2 and 3.
5. From the given assumptions, by Propositions 2 and 3, we have $N(A, B; K, L, M; T) = N_1(K, L, M; T)$ and $N(B^t, A^t; L, K, M; T) = N_1(L, K, M; T)$. The rest follows immediately from $N_1(K, L, M; T) = D(KL-1) + 2M = D(LK-1) + 2M = N_1(L, K, M; T)$. □

Remark 1. *Clearly, if $T \leq K$ and $T > K(L-1)+1$ then $L = 1$. In this case, from Propositions 3 and 2, we have that $N(A, B; K, 1, M; T) = N_2(K, 1, M; T)$.*

Theorem 3. *Let $T \leq K$ and $T > K(L-1)+1$.*

(i) *Assume $T > L$ and $T \leq L(K-1)+1$ then $N(A, B; K, L, M; T) = N_2(K, 1, M; T) = N_3(1, K, M; T) = N(B^t, A^t; L, K, M; T)$.*
(ii) *Assume $T = 1 \leq L$ and $T \leq L(K-1)+1$ then $N(A, B; K, L, M; T) = N_2(K, 1, M; 1) < N_1(1, K, M; 1) = N(B^t, A^t; L, K, M; T)$.*

Proof. (i) Since $L = 1$ we have that
$N_2(K, 1, M; T) = D(K+T-2) + M = D(L(K-1) + T - 1) + M = N_3(1, K, M; T)$
and so the result follows.
(ii) We see that
$N_2(K, 1, M; 1) = D(K-1) + M < D(K-1) + 2M = N_1(1, K, M; 1)$
□

Remark 2. *The remaining two cases lead to a contradiction and can hence never occur. Let $T \leq K$ and $T > K(L-1)+1$ and $T > L(K-1)+1$. By Remark 1, we have that $L = 1$ and we obtain the contradiction $T \leq K < T$.*

3.3. T-Collusion

Each query is masked with a polynomial of the form $\sum_{i=0}^{T-1} x^{iD} R_i$, where R_i is chosen uniformly at random. A query is private in the case of T servers colluding if and only if the matrix

$$M(x_1, \ldots, x_T) := \begin{pmatrix} 1 & \cdots & 1 \\ x_1^D & \cdots & x_T^D \\ \vdots & \ddots & \vdots \\ x_1^{D(T-1)} & \cdots & x_T^{D(T-1)} \end{pmatrix}$$

has full rank for any subset of T evaluation points. This is the same as condition C2 in [27]. Because of the very specific set of exponents used, we can give a more explicit condition for the invertibility of this matrix.

Proposition 4. *The matrix $M(x_1, \ldots, x_T)$ is invertible if and only if the elements $x_1^D, \ldots, x_T^D$ are distinct.*

Proof. $M(x_1, \ldots, x_T)$ is a Vandermonde matrix with entries $x_1^D, \ldots, x_T^D$. □

Proposition 5. *A set of elements of $\mathbb{F}_q$ such that their D^{th} powers are pairwise different has size at most $N = \frac{q-1}{\gcd(q-1, D)} + 1$.*

Proof. Fix a generator γ of $\mathbb{F}_q^*$. Then the image of the map $x \mapsto x^D$ from $\mathbb{F}_q$ to $\mathbb{F}_q$ is given by 0 together with all powers γ^{Di} where $0 \leq i < q-1$. □

Corollary 1. *Let $T < q$. If $\gcd(q-1, D) = 1$, then the scheme in Section 3 is secure against T-collusion for any choice of evaluation points.*

3.4. Stragglers and Byzantine Servers

Considering the scheme as described in the previous section, we see that the responses are the coordinates of a codeword of a Reed–Solomon code. The polynomial that needs to be interpolated has degree at most $N = N(K, L, M; T)$, and hence $N+1$ evaluation points suffice for reconstruction. Any $N+1$ evaluation points are admissible and hence we have the following theorem.

Theorem 4. *The scheme in Section 3 is straggler resistant against S stragglers if $N+1+S$ helper nodes are used.*

Proof. The responses can be considered as a codeword in an $[N+1+S, N+1, S+1]$ RS code, with S erasures. Since S is smaller than the minimum distance of the code, the full codeword and hence the interpolating polynomial can be recovered. □

Similarly, we can use additional helper nodes to account for possible Byzantine servers whose responses are incorrect.

Theorem 5. *The scheme in Section 3 is resistant against Byzantine attacks of up to B helper nodes if $N+1+2B$ helper nodes are used.*

Proof. The responses can be considered as a codeword in an $[N+1+2B, N+1, 2B+1]$ RS code, with B errors. Since $2B$ is smaller than the minimum distance of the code, the full codeword and hence the interpolating polynomial can be recovered. □

Combining both theorems give us the following corollary.

Corollary 2. *The scheme in Section 3 is resistant against S stragglers and B Byzantine helper nodes if $N+1+S+2B$ helper nodes are used.*

4. Gaps in the Polynomial

The upper bound on the recovery threshold given by the maximum degree of the product PQ can actually be improved if we choose instead to use the fact that we need only as many servers as non-zero coefficients. Similar to considerations in [9], as a basic observation of linear algebra, we note that only as many evaluation points as there are possible non-zero coordinates are required to retrieve the required matrix coefficients of PQ. Let PQ have degree $r-1$ and suppose that $q \geq r+1$. Let $\alpha_1, \ldots, \alpha_r$ be distinct elements of $\mathbb{F}_q^\times$. Suppose that the zero coefficients of PQ are indexed by $\mathcal{I}$ and let $i = r - |\mathcal{I}|$. There exist $j_1, \ldots, j_i \in \{1, \ldots, r\}$ such that the $i \times i$ matrix V, found by deleting the columns of $V(\alpha_{j_1}, \ldots, \alpha_{j_i})$ indexed by $\mathcal{I}$, is invertible. Then, each (s,t)-entry of the unknown coefficients of the polynomial $PQ \in \mathbb{F}_q^{a \times b}[x]$ can be retrieved by computing the product

$$V^{-1}((P(\alpha_j)Q(\alpha_j))_{s,t} : j \in [r] \backslash \mathcal{I})^t.$$

Theorem 6. *Let $M \geq 2$, $D = M + 2$. Let*

$$\bar{P}(x) := \sum_{k=1}^{K} x^{D(k-1)} \sum_{m=1}^{M} x^m A_{k,m}, \quad R(x) := \sum_{t=1}^{T} x^{D(t-1)} R_t,$$
$$\bar{Q}(x) := \sum_{\ell=1}^{L} x^{DK(\ell-1)} \sum_{m=1}^{M} x^{M-m+1} B_{m,\ell}, \quad S(x) := \sum_{t=1}^{T} x^{D(t-1)} S_t.$$

The number N of non-zero terms in the product PQ satisfies

$$N \leq \begin{cases} N_1(K,L,M;T)+1 & \text{if } M>2, T \leq K, L \geq 2 \text{ or } L=1, T=1; \\ 3LK+K-T+LT+1 & \text{if } M=2, T\leq K, L\geq 2; \\ ((L-1)K+T)M+2LK+1 & \text{if } K+1 \leq T \leq \lfloor LK/2 \rfloor +1, L\geq 2; \\ ((L-1)K+T)M+LK+2T-1 & \text{if } T > \lfloor LK/2 \rfloor +1, L \geq 2; \\ (K+T-1)M+2K+1 & \text{if } 2 \leq T \leq \lfloor K/2 \rfloor +1, L=1; \\ (K+T-1)M+K+2T-1 & \text{if } T > \lfloor K/2 \rfloor +1, L=1. \end{cases}$$

Proof. We have $P(x) = \bar{P}(x) + R(x)$ and $Q(x) = \bar{Q}(x) + S(x)$. Recall that $\bar{P}(x)$ and $R(x)$ have disjoint support, as do $\bar{Q}(x)$ and $S(x)$. From Theorem 1, for each each $k \in [K], \ell \in [L]$, the matrix

$$C_{k\ell} = A_{k,1}B_{1,\ell} + \cdots + A_{k,M}B_{M,\ell}$$

is the coefficient of x^h in $\bar{P}\bar{Q}$ for

$$h = (k-1)D + (\ell-1)KD + M + 1 = (k + (\ell-1)K)D - 1.$$

Clearly, each such coefficient $h \equiv M+1 \mod D$. The degrees of terms arising in the product PQ are given by

$$(i+zK)D + j + y + 2, \tag{7}$$
$$(i+t)D + j + 1, \tag{8}$$
$$(u+zK)D + y + 1, \tag{9}$$
$$(u+t)D. \tag{10}$$

for $i \in \{0,...,K-1\}, z \in \{0,...,L-1\}, j,y \in \{0,...,M-1\}$ and $u,t \in \{0,...,T-1\}$. The sequence (7) corresponds to terms that appear in the product $\bar{P}\bar{Q}$. By inspection, we see that no element θ in any of the sequences (8)–(10) satisfies $\theta \equiv -1 \mod D$: in (8) this would require $j = M$ and in (9) this would require $y = M$, contradicting our choices of j, y. The total number of distinct terms to be computed is the number of distinct integers appearing in the union $\mathcal{T}$ of the elements of the sequences (7)–(10). Let $\mathcal{U}_0$ denote the set of integers appearing in (7). Observe that $\mathcal{U}_0 = \{2, \ldots, (LK+1)D-4\}$, unless $M = 2$, in which case $\mathcal{U}_0 = \{j : 2 \leq j \leq 4LK, j \not\equiv 1 \mod 4\}$. Consider the set

$$\mathcal{U} := \{0, 1, 2, \ldots, (LK+1)D - 4\}.$$

We make the following observations with respect to $\mathcal{U}$.

- If $M > 2$, then $\mathcal{U} = \mathcal{U}_0 \cup \{0,1\} \subset \mathcal{T}$,
- $\mathcal{U}$ contains the elements of (8) $\iff T \leq (L-1)K + 1$,
- $\mathcal{U}$ contains the elements of (9) $\iff T \leq K$,
- $\mathcal{U}$ contains the elements of (10) $\iff T \leq \lfloor LK/2 \rfloor + 1$.

Consider the following sets.

$$\begin{aligned} \mathcal{U}_1 &:= \{\alpha D + i : 0 \leq \alpha \leq K+T-2, 1 \leq i \leq M\}, |\mathcal{U}_1| = (K+T-1)M; \\ \mathcal{U}_2 &:= \{\beta D + j : 0 \leq \beta \leq T-1+(L-1)K, 1 \leq j \leq M\}, |\mathcal{U}_2| = ((L-1)K+T)M; \\ \mathcal{U}_3 &:= \{\gamma D : 0 \leq \gamma \leq 2T-2\}, |\mathcal{U}_3| = 2T-1. \end{aligned}$$

Clearly, $\mathcal{U}_1$ comprises the elements of the sequence (8) and the members of $\mathcal{U}_3$ are exactly those of the sequence (10). For $T \geq K+1$, we have

$$\{u + xK : 0 \leq u \leq T-1, 0 \leq x \leq L-1\} = \{\beta : 0 \leq \beta \leq T-1+(L-1)K\},$$

in which case $\mathcal{U}_2$ is exactly the set of elements of (9). It follows that $\mathcal{U}_1 \cup \mathcal{U}_2 \cup \mathcal{U}_3 \subseteq \mathcal{U}$ if and only if $T \leq \min\{(L-1)K+1, K, \lfloor LK/2 \rfloor + 1\}$. This minimum is K if $L \geq 2$ and is 1 if $L = 1$. Furthermore, $\mathcal{U}_3$ is disjoint from $\mathcal{U}_1$ and from $\mathcal{U}_2$. If $L \geq 2$ or if $L = K = 1$, then $\mathcal{U}_1 \subset \mathcal{U}_2$, while if $L = 1$, then $\mathcal{U}_2 \subset \mathcal{U}_1$.

Suppose first that $M > 2$. We thus have that $\mathcal{U} = \mathcal{T}$ if $L \geq 2$ and $T \leq K$, or if $L = T = 1$; in either of these cases, PQ has at most

$$|\mathcal{T}| = |\mathcal{U}| = (LK+1)D - 3 = (LK-1)D + 2M + 1 = N_1(K, L, M; T) + 1$$

non-zero terms. We summarize these observations as follows.

$$\mathcal{T} \quad = \quad \begin{cases} \mathcal{U} & \text{if } L \geq 2 \text{ and } T \leq K, \text{ or if } L = T = 1; \\ \mathcal{U} \cup \mathcal{U}_1 \cup \mathcal{U}_3 & \text{if } L = 1 \\ \mathcal{U} \cup \mathcal{U}_2 \cup \mathcal{U}_3 & \text{if } L \geq 2 \text{ or if } L = K = 1. \end{cases}$$

Furthermore,

$$\begin{aligned} \mathcal{U} \cap \mathcal{U}_3 &= \{\gamma D : 0 \leq \gamma \leq \min\{2T-2, LK\}\}, \\ \mathcal{U} \cap \mathcal{U}_2 &= \{\beta D + j : 0 \leq \beta \leq \min\{LK, T-1+(L-1)K\}, 1 \leq j \leq M\} \\ &\quad \setminus \{LKD + M - 1, LKD + M\}, \\ \mathcal{U} \cap \mathcal{U}_1 &= \{\alpha D + i : 0 \leq \alpha \leq \min\{LK, T+K-2\}, 1 \leq i \leq M\} \\ &\quad \setminus \{LKD + M - 1, LKD + M\} \end{aligned}$$

Hence $|\mathcal{U} \cap \mathcal{U}_3| = \min\{2T-1, LK+1\}$. If $T \geq K+1$ then $|\mathcal{U} \cap \mathcal{U}_2| = M(LK+1) - 2$ and so, applying inclusion–exclusion, we see that, if $L \geq 2$, then

$$|\mathcal{T}| \quad = \quad \begin{cases} |\mathcal{U}| = (LK+1)D - 3 = (LK+1)(M+2) - 3 & \text{if } K \geq T; \\ |\mathcal{U} \cup \mathcal{U}_2| = ((L-1)K+T)M + 2LK + 1 & \text{if } K+1 \leq T \leq \lfloor LK/2 \rfloor + 1; \\ |\mathcal{U} \cup \mathcal{U}_2 \cup \mathcal{U}_3| = ((L-1)K+T)M + LK + 2T - 1 & \text{otherwise}. \end{cases}$$

In the case $L = 1$, we have $\mathcal{U}_2 \subseteq \mathcal{U}_1$, while if $T \leq K$ then the elements of (9) are contained in $\mathcal{U}$. Therefore, $\mathcal{T} = \mathcal{U} \cup \mathcal{U}_1 \cup \mathcal{U}_3$ and so for $T \geq 2$ we have

$$|\mathcal{T}| \quad = \quad \begin{cases} (K+T-1)M + 2K + 1 & \text{if } T \leq \lfloor K/2 \rfloor + 1; \\ (K+T-1)M + K + 2T - 1 & \text{otherwise}. \end{cases}$$

Finally, suppose that $M = 2$. If $L = 1$ then, since $U_2 \subset \mathcal{U}_1$ we have $\mathcal{T} = \mathcal{U}_0 \cup \mathcal{U}_1 \cup \mathcal{U}_3$. Similar to previous computations, we see $|\mathcal{T}|$ takes the same values as in the case for $M > 2$. If $L \geq 2$ and $T \geq K+1$ then $\mathcal{T} = \mathcal{U}_0 \cup \mathcal{U}_2 \cup \mathcal{U}_3$. Again using similar computations as before, we see in this case that $|\mathcal{T}|$ takes the same values as in the case for $M > 2$. Suppose that $L \geq 2$ and $T \leq K$. In this case, the integers appearing in (9) comprise the set

$$\mathcal{U}_2' := \{4(u+zK) + j : 0 \leq u \leq T-1, 0 \leq z \leq L-1, 1 \leq j \leq 2\}, |\mathcal{U}_2'| = 2TL.$$

We have $|\mathcal{U}_0| = 3KL$ and moreover,

$$\begin{aligned}
\mathcal{U}_0 \cap \mathcal{U}_2' &= \{4(u+zK)+2 : 0 \le u \le T-1, 0 \le z \le L-1\}, |\mathcal{U}_0 \cap \mathcal{U}_2'| = TL;\\
\mathcal{U}_0 \cap \mathcal{U}_1 &= \{4\alpha+2 : 0 \le \alpha \le K+T-2\}, |\mathcal{U}_0 \cap \mathcal{U}_1| = K+T-1;\\
\mathcal{U}_0 \cap \mathcal{U}_3 &= \{4(\alpha+1) : 0 \le \alpha \le 2T-3\}, |\mathcal{U}_0 \cap \mathcal{U}_3| = 2T-2;\\
\mathcal{U}_1 \cap \mathcal{U}_2' &= \{4(u+zK)+j : 0 \le u \le T-1, 0 \le z \le 1, 1 \le j \le 2\}, |\mathcal{U}_1 \cap \mathcal{U}_2'| = 4T;\\
\mathcal{U}_0 \cap \mathcal{U}_1 \cap \mathcal{U}_2' &= \{4(u+zK)+2 : 0 \le u \le T-1, 0 \le z \le 1\}, |\mathcal{U}_0 \cap \mathcal{U}_1 \cap \mathcal{U}_2'| = 2T.
\end{aligned}$$

Therefore, $|\mathcal{T}| = 3LK + K - T + TL + 1$.
$\square$

Example 1. *Let $M = 3, K = 3, L = 2$, that is:*

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,1} & A_{3,2} & A_{3,3} \end{bmatrix}, \quad B = \begin{bmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \\ B_{3,1} & B_{3,2} \end{bmatrix}.$$

We will compute the product AB using 32 helper nodes, assuming that $T = 3$ servers may collude. Choose a pair of polynomials

$$R(z) = R_1 + R_6x^5 + R_{11}x^{10} \text{ and } S(z) = S_1 + S_6x^5 + S_{11}x^{10},$$

whose non-zero matrix coefficients are chosen uniformly at random over $\mathbb{F}_q$. We have

$$\begin{aligned}
\bar{P}(x) &= x(A_{1,1} + A_{1,2}x + A_{1,3}x^2) + x^6(A_{2,1} + A_{2,2}x + A_{2,3}x^2) + x^{11}(A_{3,1} + A_{3,2}z + A_{3,3}z^2)\\
\bar{Q}(x) &= x(B_{3,1} + B_{2,1}x + B_{1,1}x^2) + x^{16}(B_{3,2} + B_{2,2}x + B_{1,2}x^2).
\end{aligned}$$

Define $P(x) := \bar{P}(x) + R(x)$ and $Q(x) := \bar{Q}(x) + S(x)$. In Table 2, we show the exponents that arise in the product $P(x)Q(x)$. The monomials corresponding to the computed data are $4, 9, 14, 19, 24, 29$, shown in blue. The coefficients of $x^4, x^9, x^{14}, x^{19}, x^{24}$ and x^{29} are, respectively, given by

$$\begin{aligned}
C_{1,1} &= A_{1,1}B_{1,1} + A_{1,2}B_{2,1} + A_{1,3}B_{3,1},\\
C_{1,2} &= A_{1,1}B_{1,2} + A_{1,2}B_{2,2} + A_{1,3}B_{3,2},\\
C_{2,1} &= A_{2,1}B_{1,1} + A_{2,2}B_{2,1} + A_{2,3}B_{3,1},\\
C_{2,2} &= A_{2,1}B_{1,2} + A_{2,2}B_{2,2} + A_{2,3}B_{3,2},\\
C_{3,1} &= A_{3,1}B_{1,1} + A_{3,2}B_{2,1} + A_{3,3}B_{3,1},\\
C_{3,2} &= A_{3,1}B_{1,2} + A_{3,2}B_{2,2} + A_{3,3}B_{3,2}.
\end{aligned}$$

Note that the total number of non-zero terms in PQ is $LKD + M - 1 = 32$, as predicted by Theorem 6. This also corresponds to the case for which PQ has degree $N_1(K, L, M; T) = N_1(3, 2, 3; 3) = 31$, which is consistent with Theorem 2. Therefore, 32 helper nodes are required to retrieve PQ and hence the coefficients $C_{k,m}$. If the matrices have entries over $\mathbb{F}_q$ with $q = 64$, then since $\gcd(q-1, D) = \gcd(63, 5) = 1$, the user can retrieve the data securely in the presence of 3 colluding workers.

Suppose now that we have $T = 6$ colluding servers. In this case, we have $T = 6 > 4 = \lfloor LK/2 \rfloor + 1$ and $L > 1$ and so from Theorem 6, we expect the polynomial PQ to have at most $(LK + T)D - K(M + L) - 1 = 44$ non-zero coefficients. These exponents are shown in the corresponding degree table for our scheme (see Table 3). In this case, to protect against collusion by 6 workers, we require a total of 44 helpers. While the degree of PQ in this case is 50 (see Table 1), the coefficients corresponding to the exponents $E = \{34, 39, 44, 46, 47, 48, 49\}$ are zero, and hence known a priori to the user. Let α be a root of $x^6 + x^4 + x^3 + x + 1 \in \mathbb{F}_2[x]$, so that α generates $\mathbb{F}_{64}^{\times}$.

Let V be the 44×44 matrix obtained from $V(\alpha^i : i \in [63])$ by deleting the columns and rows indexed by $E \cup \{51, \ldots, 62\}$. It is readily checked (e.g., as here, using MAGMA [28]) that the determinant of V is α^{11} and in particular is non-zero. Therefore, we can solve the system to find the unknown coefficients of PQ via the computation $V^{-1}(P(\alpha^{ij})Q(\alpha^{ij}) : i, j \in [63] \backslash (E \cup \{51, \ldots, 62\}))^t$.

Table 2. Exponents of $P(x)Q(x)$ for $K = 3, L = 2, M = 3, T = 3$. The monomial exponents which correspond to the computed data are shown in blue. The grey background marks noise exponents.

	0	1	2	3	5	16	17	18	10
0	0	1	2	3	5	16	17	18	10
1	1	2	3	4	6	17	18	19	11
2	2	1	4	5	7	18	19	20	12
3	3	4	5	6	8	19	20	21	13
5	5	6	7	8	10	21	22	23	15
6	6	7	8	9	11	22	23	24	16
7	7	8	9	10	12	23	24	25	17
8	8	9	10	11	13	24	25	26	18
10	10	11	12	13	15	26	27	28	20
11	11	12	13	14	16	27	28	29	21
12	2	13	14	15	17	28	29	30	22
13	3	14	15	16	18	29	30	31	23

Table 3. Exponents of $P(x)Q(x)$ for $K = 3, L = 2, M = 3, T = 6$. The monomial exponents which correspond to the computed data are shown in blue. The grey background marks noise exponents.

	0	1	2	3	5	16	17	18	10	15	20	25
0	0	1	2	3	5	16	17	18	10	15	20	25
1	1	2	3	4	6	17	18	19	11	16	21	26
2	2	3	4	5	7	18	19	20	12	17	22	27
3	3	4	5	6	8	19	20	21	13	18	23	28
5	5	6	7	8	10	21	22	23	15	20	25	30
6	6	7	8	9	11	22	23	24	16	21	26	31
7	7	8	9	10	12	23	24	25	17	22	27	32
8	8	9	10	11	13	24	25	26	18	23	28	33
10	10	11	12	13	15	26	27	28	20	25	30	35
11	11	12	13	14	16	27	28	29	21	26	31	36
12	2	13	14	15	17	28	29	30	22	27	32	37
13	3	14	15	16	18	29	30	31	23	28	33	38
15	15	16	17	18	20	31	32	33	25	30	35	40
20	20	21	22	23	25	36	37	38	30	35	40	45
25	25	26	27	28	30	41	42	43	35	40	45	50

We remark that for the case of no collusion, Theorem 6 does not yield an optimal scheme. The proposition below outlines a modified scheme with a lower recovery threshold if secrecy is not a consideration.

Proposition 6. *Define the polynomials:*

$$\tilde{P}(x) := \sum_{k=1}^{K} x^{(k-1)M} \sum_{m=1}^{M} x^m A_{k,m},$$
$$\tilde{Q}(x) := \sum_{\ell=1}^{L} x^{(K+\ell-1)M} \sum_{m=1}^{M} x^{M+1-m} B_{m,\ell}.$$

The following hold:

1. *For each $(i, j) \in [K] \times [L]$, C_{ij} is the coefficient of $z^{M(i+j+K-1)+1}$ in $\tilde{P}\tilde{Q}$.*
2. *The number N of non-zero terms in the product $\tilde{P}\tilde{Q}$ satisfies*

$$N \leq KLM + M - 1.$$

Proof. For each $(i,j) \in [K] \times [L]$, define the following:

- $(c_{ij}) := (M(K+i+j-1)+1)$,
- $B_M(c_{ij}) := \{c_{ij} - M + 1, \dots, c_{ij} + M - 1\} = \{c_{ij} + u : -(M-1) \leq u \leq M-1\}$.

We have

$$\tilde{P}\tilde{Q} = \sum_{k=1}^{K} \sum_{\ell=1}^{L} \sum_{m=1}^{M} \sum_{m'=1}^{M} x^{M(K+\ell+k-1)+1+m-m'} A_{k,m} B_{m',\ell}.$$

The distinct monomials arising in the product $\tilde{P}\tilde{Q}$ are those indexed by the distinct elements of $\cup_{(i,j)\in[K]\times[L]} B_M(c_{ij})$. It is straightforward to check that for each $(i,j) \in [K] \times [L]$, the integer c_{ij} is not contained in $B_m(c_{ut})$ for any $(u,t) \neq (i,j)$ and hence the required coefficients C_{ij} that appear in the product $\tilde{P}\tilde{Q}$, which are indexed by the c_{ij}, can be uniquely retrieved. We compute the number of workers required by this scheme. We have

$$\begin{aligned} V &:= \left| \bigcup_{(i,j)\in[K]\times[L]} B_M(c_{ij}) \right| \\ &= KL(2M-1) - \sum_{(i,j)\neq(u,t)} \left| B_M(c_{ij}) \cap B_M(c_{st}) \right| \\ &= KL(2M-1) - (KL-1)(M-1) = KLM + M - 1. \end{aligned}$$

□

The recovery threshold of this scheme takes the same value as the recovery threshold of the poly-entangled scheme of Theorem 1 [18].

5. Results and Comparison with the State-of-the-Art

We provide some comparison plots that highlight parameter regions of interest. In Figure 2, we compare the two variants of our own scheme. The recovery threshold when considering the maximal degree of the resulting product polynomial is shown alongside the count of possibly non-zero coefficients. We see that significant gains can be achieved, especially in the higher collusion number region.

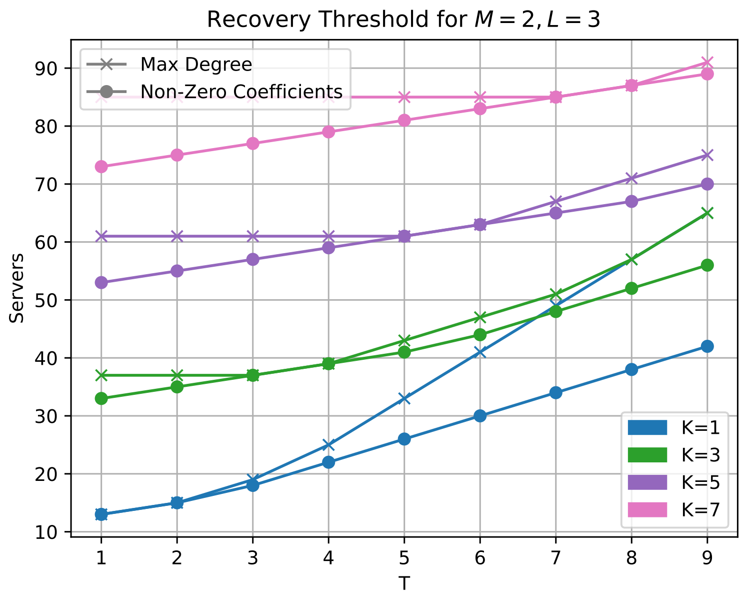

Figure 2. Comparison of maximal degree with non-zero coefficient.

In Figure 3, we compare our (non-zero coefficient) scheme with the SGPD scheme presented in [19]. For $K > 1$, we see that, except for very low values of T, our new scheme outperforms the SGPD scheme. This comparison of the recovery threshold for the two schemes is well justified since they use the same division of the matrices and will have identical upload and download costs per server.

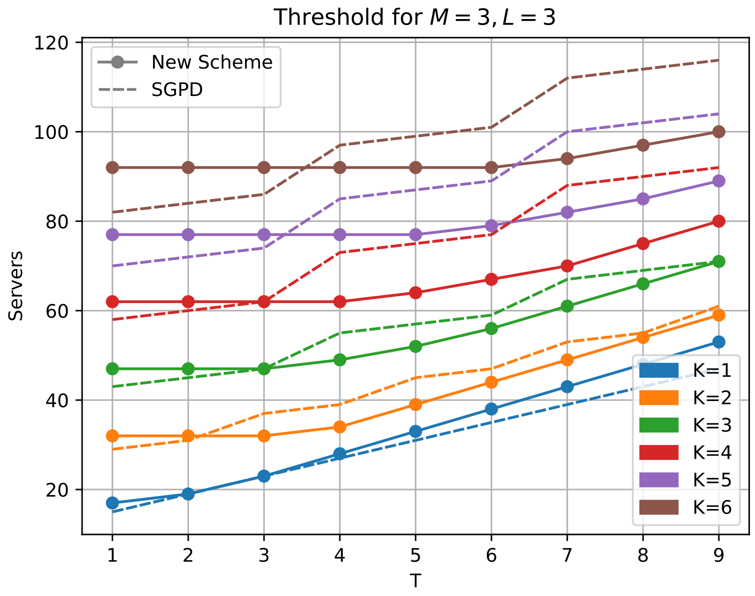

Figure 3. Comparison with [19].

The comparison in Figure 4 with the entangled codes scheme [17] and a newer scheme using roots of unity [26] shows that our new codes have lower recovery threshold for low number of colluding servers. Calculating the actual number of servers needed for the entangled scheme requires knowledge of the tensor rank of matrix multiplication. These ranks, or their best known upper bounds, are taken from [29,30]. It should be noted that the scheme in [26] requires that either $((L+1)(K+T)-1) \mid q$ or $(KML+LT+KM+T) \mid q$ where q is the field size. The requirements for our scheme outlined in Proposition 5 and Corollary 1 (i.e., that $\gcd(q-1, D) = 1, q > N$) are much less restrictive.

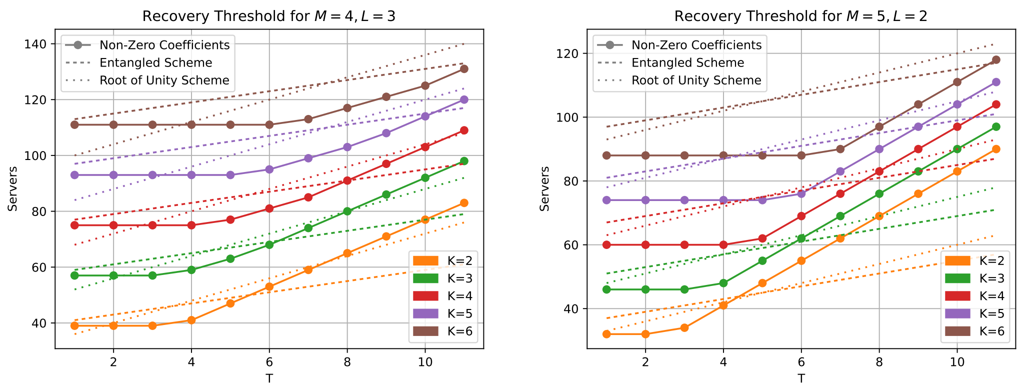

Figure 4. Comparison with [17,26] for the cases $M = 4, L = 3$ and $M = 5, L = 2$.

The comparison with the GASP scheme is less straightforward since the partitioning in GASP has a fixed value of $M = 1$. The plot in Figure 5 shows the recovery thresholds for the GASP scheme with partitioning $K = L = 3M$ as well as the recovery thresholds of our scheme for $K = L = 3$ and varying M from 1 to 5. We compare here with the maximal degree of our scheme, not the non-zero coefficients, to show that the variant of our scheme that is able to mitigate stragglers and Byzantine servers achieve much lower recovery thresholds. Fixing K and L to be the same value across this comparison means that the download cost per server is the same for all our schemes and the $K = L = 3$ GASP scheme. Note that in the $M = 1$ case, we have identical partition and hence upload cost per server as the $K = L = 3$ GASP scheme, while for $M = 2$, we have identical upload cost with the $K = L = 6$ GASP scheme, and $M = 5$ corresponds to the $K = L = 15$ GASP scheme. We can see that the grid partitioning allows for a much lower recovery threshold when the upload cost is fixed. The outer partitioning of the GASP scheme allows for low download cost per server that makes up for the higher recovery threshold. Explicitly, the outer partition into KM and LM blocks allows for a download rate of $N_{GASP}(\frac{ab}{M^2})$, where N_{GASP} is the recovery threshold for the GASP scheme. In contrast, the scheme presented in this paper will have a download rate of Nab if we partition into $K \times M$ and $M \times L$ blocks.

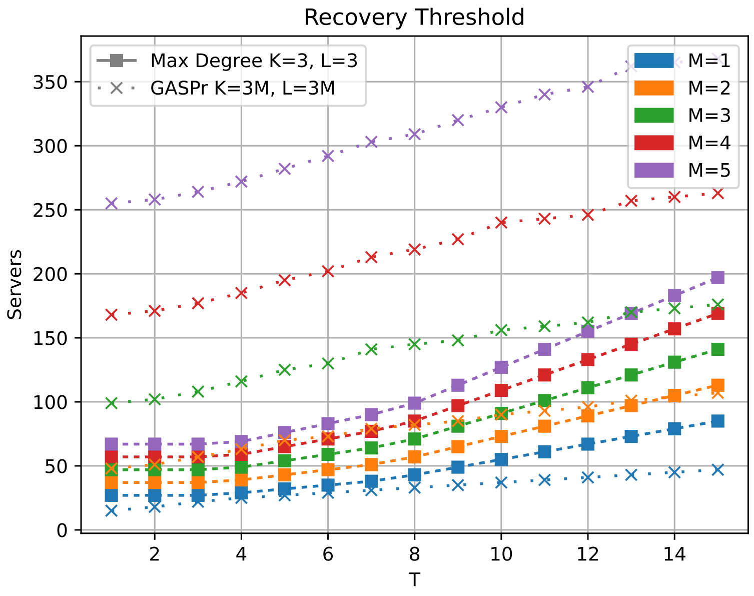

Figure 5. Comparison of the maximal degree with the $GASP_r$ scheme from [10].

It should be noted though that our construction allows to explicitly control the field size needed. In contrast, the GASP scheme might have to choose its evaluations points from an extension field Theorem 1 [9] if the base field is fixed by the entries of the matrices A and B, or just requires a very large base field. This would greatly increase the computational cost and the rates at all steps of the scheme. For example, for $K = 3, L = 3, T = 3$, GASP$_r$ uses $N = 22$ servers and the exponents for the randomness in one of the polynomials are $9, 10, 12$. Then, there are no suitable evaluation points for $q = 23, 25, 27, 29, 31, 32, 37, 41, 43$ and so for these values of q, an extension field is required.

Furthermore, the scheme presented in this paper can be used in situations where stragglers or Byzantine servers are expected as described in Corollary 2.

Complexity

We summarize the cost of $\mathbb{F}_q$-arithmetic operations and transmission of $\mathbb{F}_q$ elements associated with this scheme, using N servers. We refer the reader to ([25], Table 1) and ([26], Table 1) to view the complexity of other schemes in the literature (note that the costs defined in [25] are normalized). There are various trade-offs in costs depending on the partitioning chosen (the proposed scheme is completely flexible in this respect), ability to handle stragglers and Byzantine servers, and constraints on the field size q.

We remark that additions in general are much less costly than $\mathbb{F}_q$-multiplications in terms of space and time: for example, if $q = 2^\ell$, then an addition has space complexity (number of AND and XOR gates) $\mathcal{O}(\ell)$ and costs 1 clock in time, while multiplication has space complexity $\mathcal{O}(\ell^2)$ and time complexity $\mathcal{O}(\log_2(\ell))$ [31,32].

The encoding complexity of our scheme comes at the cost of evaluating the pair of polynomials $P(x)$ and $Q(x)$ each at N distinct elements of $\mathbb{F}_q$. This is equivalent to performing $Nr(a+b)$ (scalar) polynomial evaluations in $\mathbb{F}_q$. Given $\alpha \in \mathbb{F}_q$, the (i,j)-entry of $P(\alpha)$ is an evaluation of an $\mathbb{F}_q$-polynomial with $KM+T$ coefficients, while the (i,j)-entry of $Q(\alpha)$ is an evaluation of an $\mathbb{F}_q$-polynomial with $KL+T$ coefficients. The decoding complexity is the cost of interpolating the polynomial $PQ \in \mathbb{F}_q^{a\times b}[x]$ using N evaluation points, when PQ has at most N unknown coefficients.

The cost of either polynomial evaluation at N points or interpolation of a polynomial of degree at most $N-1$ has complexity $\mathcal{O}(N\log^2 N\log\log N)$. Therefore, we have the following statement.

Proposition 7.

1. *The encoding phase of the scheme presented in Section 3, using N servers, has complexity* $\mathcal{O}((a+b)rN\log^2 N\log\log N)$.
2. *The decoding phase of the scheme presented in Section 3, using N servers, has complexity* $\mathcal{O}(abN\log^2 N\log\log N)$.
3. *The total upload cost of the scheme presented in Section 3, using N servers, is* $r(a+b)N$.
4. *The total download cost of the scheme presented in Section 3, using N servers, is* abN.

6. Conclusions

In this work, we addressed the problem of secure distributed matrix multiplication for $C=AB$ in terms of designing polynomial codes for this setting. In particular, we assumed that A and B contain confidential data, which must be kept secure from colluding workers. Similar to some previous work also employing polynomial codes for distributed matrix multiplication, we proposed to deliberately leave gaps in the polynomial coefficients for certain degrees and provided a new code construction which is able to exploit these gaps to lower the recovery threshold. For this construction, we also presented new closed-form expressions for the recovery threshold as a function of the number of colluding workers and the specific number of submatrices that the matrices A and B are partitioned into during encoding. Further, in the absence of any security constraints, we showed that our construction is optimal in terms of recovery threshold. Our proposed scheme improves on the recovery threshold of existing schemes from the literature in particular for large dimensions of A and a larger number of colluding workers, in some cases, even by a large margin.

Author Contributions: Writing—original draft, E.B., O.W.G., and J.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by U.S. National Science Foundation grants 1815322, 1908756, 2107370 in addition to the UCD Seed Funding- *Horizon Scanning* scheme (grant no. 54584).

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Janzamin, M.; Sedghi, H.; Anandkumar, A. Beating the perils of non-convexity: Guaranteed training of neural networks using tensor methods. *arXiv* **2015**, arXiv:1506.08473.
2. Joshi, G.; Soljanin, E.; Wornell, G. Efficient redundancy techniques for latency reduction in cloud systems. *ACM Trans. Model. Perform. Eval. Comput. Syst. (TOMPECS)* **2017**, *2*, 12:1–12:30. [CrossRef]
3. Lee, K.; Suh, C.; Ramchandran, K. High-dimensional coded matrix multiplication. In Proceedings of the IEEE International Symposium on Information Theory (ISIT), Aachen, Germany, 25–30 June 2017; pp. 2418–2422.
4. Lee, K.; Lam, M.; Pedarsani, R.; Papailiopoulos, D.; Ramchandran, K. Speeding Up Distributed Machine Learning Using Codes. *IEEE Trans. Inf. Theory* **2018**, *64*, 1514–1529. [CrossRef]

5. Yu, Q.; Maddah-Ali, M.; Avestimehr, S. Polynomial codes: An optimal design for high-dimensional coded matrix multiplication. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 4403–4413.
6. Li, S.; Maddah-Ali, M.A.; Yu, Q.; Avestimehr, A.S. A fundamental tradeoff between computation and communication in distributed computing. *IEEE Trans. Inform. Theory* **2017**, *64*, 109–128. [CrossRef]
7. Aliasgari, M.; Simeone, O.; Kliewer, J. Distributed and Private Coded Matrix Computation with Flexible Communication Load. *arXiv* **2019**, arXiv:1901.07705.
8. Yang, H.; Lee, J. Secure Distributed Computing With Straggling Servers Using Polynomial Codes. *IEEE Trans. Inf. Forensics Secur.* **2019**, *14*, 141–150. [CrossRef]
9. D'Oliveira, R.G.L.; El Rouayheb, S.; Karpuk, D. GASP Codes for Secure Distributed Matrix Multiplication. *IEEE Trans. Inf. Theory* **2020**, *66*, 4038–4050. [CrossRef]
10. D'Oliveira, R.G.L.; El Rouayheb, S.; Heinlein, D.; Karpuk, D. Degree Tables for Secure Distributed Matrix Multiplication. *IEEE J. Sel. Areas Inf. Theory* **2021**, *2*, 907–918. [CrossRef]
11. Yu, Q.; Raviv, N.; So, J.; Avestimehr, A.S. Lagrange Coded Computing: Optimal Design for Resiliency, Security and Privacy. *arXiv* **2018**, arXiv:1806.00939.
12. Kakar, J.; Ebadifar, S.; Sezgin, A. On the Capacity and Straggler-Robustness of Distributed Secure Matrix Multiplication. *IEEE Access* **2019**, *7*, 45783–45799. [CrossRef]
13. Chang, W.T.; Tandon, R. On the capacity of secure distributed matrix multiplication. In Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 9–13 December 2018; pp. 1–6.
14. Chang, W.T.; Tandon, R. On the Upload versus Download Cost for Secure and Private Matrix Multiplication. In Proceedings of the 2019 IEEE Information Theory Workshop (ITW), Gotland, Sweden, 25–28 August 2019; pp. 1–5.
15. Dutta, S.; Bai, Z.; Jeong, H.; Low, T.M.; Grover, P. A unified coded deep neural network training strategy based on generalized PolyDot codes. In Proceedings of the 2018 IEEE International Symposium on Information Theory (ISIT), Vail, CO, USA, 17–22 June 2018; pp. 1585–1589.
16. Dutta, S.; Fahim, M.; Haddadpour, F.; Jeong, H.; Cadambe, V.; Grover, P. On the Optimal Recovery Threshold of Coded Matrix Multiplication. *IEEE Trans. Inf. Theory* **2020**, *66*, 278–301. [CrossRef]
17. Yu, Q.; Avestimehr, A.S. Entangled Polynomial Codes for Secure, Private, and Batch Distributed Matrix Multiplication: Breaking the "Cubic" Barrier. In Proceedings of the 2020 IEEE International Symposium on Information Theory (ISIT), Los Angeles, CA, USA, 21–26 June 2020; pp. 245–250.
18. Yu, Q.; Maddah-Ali, M.A.; Avestimehr, A.S. Straggler Mitigation in Distributed Matrix Multiplication: Fundamental Limits and Optimal Coding. *IEEE Trans. Inf. Theory* **2020**, *66*, 1920–1933. [CrossRef]
19. Aliasgari, M.; Simeone, O.; Kliewer, J. Private and Secure Distributed Matrix Multiplication With Flexible Communication Load. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 2722–2734. [CrossRef]
20. Wang, H.-P.; Duursma, I. Parity-Checked Strassen Algorithm. *arXiv* **2020**, arXiv:2011.15082.
21. Hasirciolu, B.; Gomez-Vilardebo, J.; Gunduz, D. Bivariate Polynomial Codes for Secure Distributed Matrix Multiplication. *IEEE J. Sel. Areas Commun.* **2022**, *40*, 955–967. [CrossRef]
22. Li, J.; Hollanti, C. Private and Secure Distributed Matrix Multiplication Schemes for Replicated or MDS-Coded Servers. *IEEE Trans. Inf. Forensics Secur.* **2022**, *17*, 659–669. [CrossRef]
23. Machado, R.A.; D'Oliveira, R.G.L.; Rouayheb, S.E.; Heinlein, D. Field Trace Polynomial Codes for Secure Distributed Matrix Multiplication. In Proceedings of the 2021 XVII International Symposium "Problems of Redundancy in Information and Control Systems" (REDUNDANCY), Prague, Czech Republic, 23–25 November 2021.
24. Makkonen, O.; Hollanti, C. General Framework for Linear Secure Distributed Matrix Multiplication with Byzantine Servers. *arXiv* **2022**, arXiv:2205.07052.
25. Mital, N.; Ling, C.; Gündüz, D. Secure Distributed Matrix Computation With Discrete Fourier Transform. *IEEE Trans. Inf. Theory* **2022**, *68*, 4666–4680. [CrossRef]
26. Machado, R.A.; Manganiello, F. Root of Unity for Secure Distributed Matrix Multiplication: Grid Partition Case. *arXiv* **2022**, arXiv:2206.01559.
27. Zhu, J.; Li, S. A Systematic Approach towards Efficient Private Matrix Multiplication. *IEEE J. Sel. Areas Inf. Theory* **2022**, *3*, 257–274. [CrossRef]
28. Bosma, W.; Cannon, J.; Playoust, C. The Magma algebra system. I. The user language. *J. Symb. Comput.* **1997**, *24*, 235–265.
29. Sedoglavic, A. Yet Another Catalogue of Fast Matrix Multiplication Algorithms. Available online: https://fmm.univ-lille.fr/ (accessed on 28 October 2022).
30. Fawzi, A.; Balog, M.; Huang, A.; Hubert, T.; Romera-Paredes, B.; Barekatain, M.; Novikov, A.; Ruiz, F.J.; Schrittwieser, J.; Swirszcz, G.; et al. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature* **2022**, *610*, 47–53. [CrossRef] [PubMed]

31. Elia, M.; Leone, M. On the inherent space complexity of fast parallel multipliers for GF(2/sup m/). *IEEE Trans. Comput.* **2002**, *51*, 346–351. [CrossRef]
32. Elia, M.; Rosenthal, J.; Schipani, D. Polynomial evaluation over finite fields: New algorithms and complexity bounds. *Appl. Algebra Eng. Commun. Comput.* **2012**, *23*, 129–141. [CrossRef]