Li, Jie; Hollanti, Camilla

Improved Private and Secure Distributed Matrix Multiplication

Published in:
2021 IEEE International Symposium on Information Theory, ISIT 2021 - Proceedings

DOI:
10.1109/ISIT45174.2021.9517825

Published: 12/07/2021

Please cite the original version:

This material is protected by copyright and other intellectual property rights, and duplication or sale of all or part of any of the repository collections is not permitted, except that material may be duplicated by you for your research use or educational purposes in electronic or print form. You must obtain permission for any other use. Electronic or print copies may not be offered, whether for sale or otherwise to anyone who is not an authorised user.
Improved Private and Secure Distributed Matrix Multiplication

Jie Li and Camilla Hollanti
Department of Mathematics and Systems Analysis, Aalto University
FI-00076 Aalto, Finland
E-mails: jie.li873@gmail.com, camilla.hollanti@aalto.fi

Abstract—Consider the problem of a user having a private matrix $A$ and $N$ non-colluding servers sharing a library of $L$ ($L > 1$) matrices $B^{(0)}, B^{(1)}, \ldots, B^{(L-1)}$, for which the user wishes to compute $AB^{(\theta)}$ for some $\theta \in [0, L)$ without revealing any information of the matrix $A$ to the servers, and keeping the index $\theta$ private to the servers. This problem is known as private and secure distributed matrix multiplication (PSDMM) and is supposed to have wide application potential. However, studies of PSDMM are still scarce in the literature. In this paper, we propose a new efficient private and secure distributed matrix multiplication coding scheme, which has a better performance than state-of-the-art schemes in that it achieves a smaller recovery threshold and download cost as well as providing a more flexible tradeoff between the upload and download costs.

I. INTRODUCTION

Matrix multiplication is one of the key operations in many science and engineering fields, such as machine learning and cloud computing. Carrying out the computation on distributed servers is desirable for improving efficiency and reducing the user’s computation load, as the user can divide the computation at hand into several sub-tasks to be carried out by the helper servers. By scaling out computations across many distributed servers, the computation latency can be affected by orders of magnitude due to the presence of stragglers, see e.g., [1]. Recent work has shown that coding technique can help reduce the computation latency [2]–[6].

As the computations are scaling out across many distributed servers, besides stragglers, security is also a concern as the servers might be curious about the matrix contents. A typical assumption is that any $X$ of them may collude to deduce information about the matrices possessed by the user [7]. This raises the problem of secure distributed matrix multiplication (SDMM), which has recently received a lot of attention from an information-theoretic perspective [7]–[14].

In a variant of this problem, the user privacy should also be taken into account. Consider the scenario that the user has a private matrix $A$ and there are $N$ non-colluding servers sharing a library of $L$ matrices $B^{(0)}, B^{(1)}, \ldots, B^{(L-1)}$, for which the user wishes to compute $AB^{(\theta)}$ for some $\theta \in [0, L)$ without revealing any information of the matrix $A$ to the servers, and the servers should not identify the index $\theta$ of the desired matrix $B^{(\theta)}$. This problem is termed private and secure distributed matrix multiplication (PSDMM), which is supposed to have wide application potential, including e-Health [15], smart systems [16], private infection tracking [17], AI recommendation service [18], etc. PSDMM together with the problem of SDMM is also quite relevant to the problem of private information retrieval (PIR) [19]–[33]. Indeed, a connection between a form of SDMM and a form of PIR was drawn in [8], where an upper bound of the capacity of SDMM is characterized by that of PIR. Later in [29], a scheme that was initially used to study the problem of PIR was also applied to another PSDMM model that differs from the one in this paper. Further in [34], the problem of SDMM is converted into a PIR problem. The PSDMM code in [35] is also based on PIR. Besides, PSDMM is also relevant to the problem of private computation [36]. Especially, when the private matrix $A$ is a row vector while the matrices $B^{(i)}$ stored across the servers are column vectors, the problem of PSDMM can be viewed as the problem of private computation.

Generally, three performance metrics are of particular interest for PSDMM and more generally for any matrix multiplication schemes:

- The upload cost: the amount of data transmitted from the user to the servers to assign the sub-tasks;
- The download cost: the amount of data to be downloaded from the servers;
- The recovery threshold $R_c$: the number of servers that need to complete their task before the user can recover the desired matrix product.

Previous schemes aim at minimizing the above three metrics. Usually, the amount of data to be downloaded from an individual server is fixed after the user assigns the sub-tasks to the servers. Thus in most cases, including all the schemes involved in this paper, minimizing the download cost is equivalent to minimizing the recovery threshold $R_c$.

Up to now, there are not many constructions for PSDM codes. The PSDMM code in [35] based on PIR can provide a flexible tradeoff between the upload cost and download cost, but requires a high sub-packetization degree and cannot mitigate stragglers. In addition, the minimum value of the upload cost of PSDMM has also been derived in [35]. However, achieving the minimum upload cost with this scheme requires a huge download cost. In [18], PSDMM codes based on the polynomial codes in [3] were presented. In [12], Aliasgari et al. proposed a new PSDMM code based on the entangled
where $L>\in$. Also assume that the private matrix $A$ is explicit and has, in certain aspects, better performance than previous ones. More specifically,

- The new PSDMM code outperforms the ones in [12] and [18] in that it has a smaller recovery threshold as well as download cost under the same upload cost, and it provides a more flexible tradeoff between the upload and download costs than the one in [18].
- The new PSDMM code also has a smaller recovery threshold and download cost when compared with the ones in [13] and [35] for some regions. In addition, it can tolerate stragglers and does not require sub-packetization, which superior to the one in [35].

The detailed comparisons will be given in Section IV.

The rest of this paper is organized as follows. Section II introduces the problem settings. Section III presents a motivating example and the new PSDMM code construction. Section IV compares the new PSDMM code to previous ones. Finally, Section V draws the concluding remarks.

II. PROBLEM SETTINGS

Let $\{i,j\}$ denote the set $\{i,i+1,\cdots,j-1\}$ for any two integers $i<j$. Throughout this paper, we assume that the $N$ servers share the library of $L$ matrices $B(i)$, $i \in [0, L)$ in a replicated form, i.e., every server stores all the $L$ matrices. We also assume that the private matrix $A$ possessed by the user is a $t \times s$ matrix over a finite field $F_q$, and the matrices $B(i)$, $i \in [0, L)$ stored in each server are $s \times r$ matrices over $F_q$, where $L>1$ and $t,s,$ and $r$ are positive integers.

Typically, a PSDMM scheme contains three phases:

- Encoding phase: The user encodes $A$ to obtain $\tilde{A}_i$ for $i \in [0, N)$.
- Query, communication and computation phase: The user sends $\tilde{A}_i$ and a query $q_i(\theta)$ (which is usually independent of the matrix $\tilde{A}_i$) to server $i$, who then first encodes the library into $\tilde{B}_i$ based on the received query, and then computes $\tilde{A}_i \tilde{B}_i$ and returns the result to the user, where $i \in [0, N)$.
- Decoding phase: From the results returned by the fastest $R_c$ servers, the user can then decode the desired matrix product $AB(\theta)$.

We say that an encoding scheme for PSDMM is private if the index $\theta$ is private to any individual server, i.e.,

$$I(\theta; q_i(\theta), \tilde{A}_i, B(i), \cdots, B(L-1)) = 0,$$

for any $i \in [0, N)$ and $\theta \in [0, L)$, where $I(Y; Z)$ denotes the mutual information between $Y$ and $Z$. A scheme is called secure if no information is leaked about the matrix $A$ to any server, i.e.,

$$I(\tilde{A}_i; A) = 0,$$

for any $i \in [0, N)$.

In addition, we wish to minimize the recovery threshold $R_c$ as well as the communication cost, which is comprised of upload cost and download cost. The upload cost is defined as $\sum_{i=0}^{N-1} H(\tilde{A}_i)$. The download cost is defined as the number of elements in $F_q$ that the user downloaded from the fastest $R_c$ servers $j_0, \cdots, j_{R_c-1}$, maximized over $\{j_0, \cdots, j_{R_c-1}\} \subseteq [0, N)$, i.e.,

$$\max_{\{j_0, \cdots, j_{R_c-1}\} \subseteq [0, N)} \sum_{i=j_0}^{j_{R_c-1}} H(\tilde{A}_i \tilde{B}_i),$$

where $H(Y)$ denotes the entropy of $Y$.

III. PRIVATE AND SECURE DISTRIBUTED MATRIX MULTIPLICATION

In this section, we propose a new PSDMM code. The construction is largely similar to the one in [12], but the encoding phase of the new PSDMM scheme is more efficient. This is enabled by considering the problem of designing efficient encoding polynomials in the encoding phase and the server computation phase, in the view of designing a degree table, which was previously introduced for the problem of SDMM [7]. A better degree table is found, yielding better encoding. This leads to a smaller recovery threshold and download cost. Before presenting the general construction, we first give a motivating example.

A. A motivating example

Assume that $L = 2$, $t,s, r$ are even, and the user wishes to compute $AB(0)$. Divide the matrices $A$ and $B(i)$ into block matrices

$$A = \begin{pmatrix} A_{0,0} & A_{0,1} \\ A_{1,0} & A_{1,1} \end{pmatrix}, \quad B(i) = \begin{pmatrix} B_{0,0}^{(i)} & B_{0,1}^{(i)} \\ B_{1,0}^{(i)} & B_{1,1}^{(i)} \end{pmatrix},$$

where $A_{k,j} \in F_q^{t \times s}$ and $B_{j,k}^{(i)} \in F_q^{s \times r}$. Then

$$AB(0) = \begin{pmatrix} C_{0,0}^{(0)} & C_{0,1}^{(0)} \\ C_{1,0}^{(0)} & C_{1,1}^{(0)} \end{pmatrix},$$

where

$$\begin{cases} C_{0,0}^{(0)} = A_{0,0}B_{0,0}^{(0)} + A_{0,1}B_{1,0}^{(0)}, \\ C_{0,1}^{(0)} = A_{0,0}B_{0,1}^{(0)} + A_{0,1}B_{1,1}^{(0)}, \\ C_{1,0}^{(0)} = A_{1,0}B_{0,0}^{(0)} + A_{1,1}B_{1,0}^{(0)}, \\ C_{1,1}^{(0)} = A_{1,0}B_{0,1}^{(0)} + A_{1,1}B_{1,1}^{(0)}. \end{cases}$$

(1)
Let \( Z \) be a random matrix over \( \mathbb{F}_q^{\frac{1}{2} \times \frac{1}{2}} \). Then, define a polynomial

\[
 f(x) = A_{0,0}x^{\alpha_{0,0}} + A_{0,1}x^{\alpha_{0,1}} + A_{1,0}x^{\alpha_{1,0}} + A_{1,1}x^{\alpha_{1,1}} + Zx^{\gamma},
\]

where \( \alpha_{k,j}, \gamma \) are some integers to be specified later.

Let \( a_0, a_1, \ldots, a_{N-1} \) be \( N + 1 \) pairwise distinct elements randomly chosen from \( \mathbb{F}_q \). For every \( i \in [0, N) \), the user first evaluates \( f(x) \) at \( a_{0,i} \), then sends \( f(a_{0,i}) \) and the query

\[
 q_i^{(0)} = (a_{0,i}, a_1)
\]

to server \( i \). Upon receiving the query \( q_i^{(0)} \), server \( i \) encodes the library into a matrix as \( g(a_{0,i}) \) where \( g(x) \) is defined as

\[
 g(x) = \sum_{j=0}^{1} \sum_{k=0}^{1} B_{j,k} \gamma^{\beta_{j,k} + a_1}. \]

After encoding the library, server \( i \) computes \( f(a_{1})g(a_{1}) \) and then returns the result to the user.

Let \( h(x) = f(x)g(x) \), where the exact expression is given in (5) in the next page. Then the results returned to the user by the servers are exactly the evaluations of \( h(x) \) at some evaluation points.

The user wishes to obtain the data in (1) (related to the useful terms in \( h(x) \) from any \( R_e \) out of the \( N \) evaluations of \( h(x) \), which can be fulfilled if

(i) For \( k \in [0,2) \) and \( k' \in [0,2) \),

\[
 \alpha_{k,0} + \beta_{0,k'} = \alpha_{k,1} + \beta_{1,k'}.
\]

(ii) For \( U = \{ \alpha_{k,0} + \beta_{0,k'} | 0 \leq k < 2, 0 \leq k' < 2 \} \) and \( I = \{ \alpha_{k,j} + \beta_{j,k'} | 0 \leq k, k' < 2, 0 \leq j \neq j' < 2 \} \) where

\[
 |U| = 4 \text{ and } U \cap I = \emptyset.
\]

(iii) \( R_e = \deg(h(x)) + 1 \).

The task can be finished because

- (i) guarantees that each \( C_{k,k}^{(0)} \) appears in \( h(x) \),
- (ii) guarantees that \( C_{k,k'}^{(0)} \), \( k, k' \in [0,2) \) are coefficients of different terms of \( h(x) \) with different degrees, which are different from the degrees of the interference terms of \( h(x) \), i.e., each \( C_{k,k}^{(0)} \) is the coefficient of a unique term in \( h(x) \),
- (iii) guarantees the decodability from Lagrange interpolation.

By (2) and (ii), we can get \( \gamma \neq \alpha_{k,j} \) for \( k, j = 0, 1 \), thus one easily obtains \( I \{ \{ f(a_{i}) \}; A \} = 0 \) for any \( i \in [0, N) \) and \( a_i \in \mathbb{F}_Q \), hence security is satisfied. While the privacy condition is met by the definition of the query vector in Eq. (3) for the desired index, the detailed proof is similar to [18].

We provide a concrete exponent assignment for this example in Table I (named degree table\(^1\) in [7]). From the given assignment, we see that \( \deg(h(x)) = 13 \) and thus \( R_e = 14 \).

### B. General construction

In the following, we propose a general construction for PSDMM. Let \( m, p, \) and \( n \) be positive integers such that \( m | t \), \( p | s \), \( n | r \), then we divide the matrices \( A \) and \( B^{(i)} (i \in [0, L]) \) into block matrices as

\[
 A = (A_{k,j})_{0 \leq k < m, 0 \leq j < p}, \quad B^{(i)} = \left( B^{(i)}_{j,k'} \right)_{0 \leq j' < p, 0 \leq k' < n},
\]

where \( A_{k,j} \in \mathbb{F}_q^{\frac{1}{2} \times \frac{1}{2}} \) and \( B^{(i)}_{j,k'} \in \mathbb{F}_q^{\frac{1}{2} \times \frac{1}{2}} \).

Thus,

\[
 AB^{(i)} = C^{(i)} = \left( C^{(i)}_{k,k'} \right)_{0 \leq k < m, 0 \leq k' < n}
\]

where

\[
 C^{(i)}_{k,k'} = \sum_{j=0}^{p-1} A_{k,j} B^{(i)}_{j,k'}, \quad k \in [0, m), \quad k' \in [0, n).
\]

Let \( Z \) be a random matrix over \( \mathbb{F}_q^{\frac{1}{2} \times \frac{1}{2}} \), and define a polynomial

\[
 f(x) = \sum_{k=0}^{m-1} \sum_{j=0}^{p-1} A_{k,j} x^{\alpha_{k,j}} + Zx^{\gamma},
\]

where \( \alpha_{k,j}, \gamma \) are some integers to be specified later.

Suppose the user wishes to compute \( AB^{(i)} \) for some \( \theta \in [0, L) \), then let \( a_0, \ldots, a_{\theta-1}, a_{\theta+1}, \ldots, a_{L-1} \) and \( a_0, a_1, \ldots, a_{\theta}, N-1 \) be \( N + L - 1 \) pairwise distinct elements randomly chosen from \( \mathbb{F}_q \). For every \( i \in [0, N) \), the user first evaluates \( f(x) \) at \( a_{\theta,i} \), then sends \( f(a_{\theta,i}) \) and the query

\[
 q_i^{(0)} = (a_0, \ldots, a_{\theta-1}, a_{\theta+1}, \ldots, a_{L-1})
\]

to server \( i \).

Let \( g_i(x) = \sum_{k=0}^{p-1} \sum_{j=0}^{n-1} B^{(i)}_{j,k} x^{\beta_{j,k}}, \) where \( \beta_{j,k}, j \in [0, p), k \in [0, n) \) are some integers to be specified later. Let \( q_i^{(0)}(t) \) be the \( t \)-th element in the vector \( q_i^{(0)} \). Then, upon receiving the

\(^{1}\)In [7], the variables in the first column and the first row of the degree table denote the exponents of the encoding polynomial for the left side matrix \( A \) and the right side matrix \( B \). Although there are \( L \) right side matrices in the PSDMM problem, the exponents of the encoding polynomials for the \( L \) matrices are the same in our scheme according to (4), thus the PSDMM problem can also be studied in the view of optimizing the degree table.
Theorem 1

For convenience, let

\[
\text{query } q_i^{(\theta)} \text{, server } i \text{ first encodes the library into the following matrix}
\]

\[
L-1 \sum_{t=0} \sum_{j=0}^{p-1} \sum_{k=0}^{n-1} B_j^{(\theta)} a_{\theta,j} k + \sum_{t, t \neq \theta} \sum_{j=0}^{p-1} \sum_{k=0}^{n-1} B_j^{(\theta)} a_{\theta,j} k.
\]

For convenience, let

\[
g(x) = g(a_{\theta,i}) = \sum_{t=0}^{L-1} g_i^{(\theta)}[t].
\]

Now it is obvious that \( g(a_{\theta,i}) = \sum_{t=0}^{L-1} g_i^{(\theta)}[t]. \)

After encoding the library, server \( i \) computes \( f(a_{\theta,i})g(a_{\theta,i}) \) and then sends the result back to the user. Let \( h(x) = f(x)g(x) \), i.e.,

\[
h(x) = \sum_{k=0}^{m-1} \sum_{j=0}^{p-1} \sum_{k'=0}^{n-1} A_{k,j} B_j^{(\theta)} x^{a_{\theta,j} k + \beta_{j,k'}, k'} + \sum_{j'=0}^{p-1} \sum_{k'=0}^{n-1} Z B_j^{(\theta)} x^{\gamma + \beta_{j', k'}}.
\]

interface terms

\[
\left( A_{0,0} x^{a_{0,0}} + A_{0,1} x^{a_{0,1}} + A_{1,1} x^{a_{1,1}} + Z x^\gamma \right) \left( B_0^{(1)} a_{1,0} + B_1^{(1)} a_{1,1} + B_1^{(1)} a_{1,1} \right)
\]

interface terms

\[
(5)
\]

### Table II

<table>
<thead>
<tr>
<th>Chang–Tan’s code</th>
<th>Upload cost</th>
<th>Download cost</th>
<th>Recovery threshold ( R_c )</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \frac{R_c}{m} )</td>
<td>( tr \frac{m+1}{m} )</td>
<td>( m+1 )</td>
<td>( n+1 )</td>
<td>[33]</td>
</tr>
<tr>
<td>Kim–Lee’s code</td>
<td>( \frac{R_c}{m} )</td>
<td>( m+1 )</td>
<td>( n+1 )</td>
<td>[18]</td>
</tr>
<tr>
<td>Aliassari et al.’s PSQPD code</td>
<td>( \frac{R_c}{mp} )</td>
<td>( pmn+pm+p+2 )</td>
<td>( 1 )</td>
<td>[12]</td>
</tr>
<tr>
<td>Yu et al.’s PSQPD code</td>
<td>( \frac{R_c}{mp} )</td>
<td>( 2R(m, p, n) + 1 )</td>
<td>( 1 )</td>
<td>[13]</td>
</tr>
<tr>
<td>The new PSQPD code</td>
<td>( \frac{R_c}{mn} )</td>
<td>( pmn+pm+n )</td>
<td>( 1 )</td>
<td>Theorem 1</td>
</tr>
</tbody>
</table>

A comparison of key parameters between the proposed PSQPD code and some previous codes. \( R(p, m, n) \) denotes the minimum number of element-wise multiplications required to complete the multiplication of an \( m \times p \) matrix and a \( p \times n \) matrix [6].
matrix $Z$ according to (6). The detailed and rigorous proof is similar to [18], thus we omit it here.

It is obvious that the upload cost is $\sum_{i=0}^{L-1} H(f(a_i)) = N \frac{tr}{mp}$ and the download cost is $R \frac{tr}{mn}$. This finishes the proof. \hfill \blacksquare

IV. COMPARISON

Next, we provide comparisons of some key parameters of the proposed PSDMM code and some previous codes. Table II illustrates the first comparison. To this end, note that Chang–Tandon’s code [35] cannot mitigate stragglers and the number of servers can be an arbitrary integer larger than $m$ but should be prefixed. Table III gives a (fairer) comparison of the normalized parameters of the proposed PSDMM code and Chang–Tandon’s code, where the normalized upload cost is the ratio of upload cost to $H(A)$ and the normalized download cost is the ratio of the download cost to $H(AB^{(L)})$. Figure 1 visualizes the comparison in Table III by assuming $N = R_c$ in the new PSDMM code.

### Table III

<table>
<thead>
<tr>
<th>References</th>
<th>Normalized upload cost $R_c/m$</th>
<th>Normalized download cost $R_c/mn$</th>
<th>Recovery threshold $R_c$</th>
<th>$p$, $m$, $n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chang–Tandon’s code</td>
<td>$R_c/m$</td>
<td>$1 + \frac{m+1}{R_c} + \cdots + \frac{m+1}{R_c}L^{-1}$</td>
<td>$\geq m+1$</td>
<td>[35]</td>
</tr>
<tr>
<td>The new PSDMM code</td>
<td>$N/mpi$</td>
<td>$pmn + pm + n$</td>
<td>Theorem 1</td>
<td>[13]</td>
</tr>
</tbody>
</table>

IV. COMPARISON

Next, we provide comparisons of some key parameters of the proposed PSDMM code and some previous codes. Table II illustrates the first comparison. To this end, note that Chang–Tandon’s code [35] cannot mitigate stragglers and the number of servers can be an arbitrary integer larger than $m$ but should be prefixed. Table III gives a (fairer) comparison of the normalized parameters of the proposed PSDMM code and Chang–Tandon’s code, where the normalized upload cost is the ratio of upload cost to $H(A)$ and the normalized download cost is the ratio of the download cost to $H(AB^{(L)})$. Figure 1 visualizes the comparison in Table III by assuming $N = R_c$ in the new PSDMM code.

### Table IV

<table>
<thead>
<tr>
<th>$p$</th>
<th>$m$</th>
<th>$n$</th>
<th>Upper bound constructions for $R(p, m, n)$</th>
<th>$R_{BC}$ [13]</th>
<th>$R_{New}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>7 [39]</td>
<td>15</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>11 [38]</td>
<td>23</td>
<td>19</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>15 [40]</td>
<td>31</td>
<td>27</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>19 [41]</td>
<td>47</td>
<td>39</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>19 [42]</td>
<td>199</td>
<td>155</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>6</td>
<td>160 [43]</td>
<td>321</td>
<td>258</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>7</td>
<td>250 [44]</td>
<td>501</td>
<td>399</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>8</td>
<td>343 [39]</td>
<td>687</td>
<td>584</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>9</td>
<td>520 [44]</td>
<td>1041</td>
<td>819</td>
</tr>
</tbody>
</table>

From Tables II and III, the numerical comparison in Table IV, and Figure 1, we can see that the new PSDMM code has some advantages over the previous constructions:

- The new PSDMM code is more general than Kim-Lee’s code in [18]. Namely, it can provide a more flexible tradeoff between the upload cost and the download cost. In particular, when $p = 1$, the recovery threshold of the new code is one smaller than that of Kim-Lee’s code in [18] under the same upload cost.
- The recovery threshold and the download cost of the new PSDMM code are smaller than those of Aliasgari et al.’s PSGPD code [12] under the same upload cost\(^2\) and the PSDMM code in [13] for some small parameters.
- The download cost of Chang–Tandon’s PSDMM code in [35] depends on $L$ and is a monotonically increasing function w.r.t. $L$ (see the trends in Figure 1). Besides, Chang–Tandon’s PSDMM code cannot mitigate stragglers and requires a large sub-packetization degree. In contrast, the new PSDMM code does not have such defects. Furthermore, Figure 1 shows that the new PSDMM code has a smaller normalized download cost for some given normalized upload cost compared with Chang–Tandon’s PSDMM code.

V. CONCLUSION

We considered the problem of PSDMM and proposed a scheme based on [12] and the entangled polynomial code in [13]. The performance of the scheme was characterized via a degree table. Alongside, an efficient degree table was given, leading to a new PSDMM scheme that can provide a flexible tradeoff between the upload and download costs, and which has better performance than some previous codes in terms of the recovery threshold and download cost. An interesting future direction is to consider the case where the servers share a library of $L$ matrices in an encoded form, a natural starting point being maximum distance separable (MDS) coded data.

\(^2\)In [12], there was a small mistake in the recovery threshold of the PSGPD code. We give the correct value of the recovery threshold in Table II.