Citation for published version (APA):
Aliasgari, M., Simeone, O., & Kliewer, J. (2020). Private and Secure Distributed Matrix Multiplication with Flexible Communication Load. IEEE Transactions on Information Forensics and Security, 15, 2722-2734. [8985291]. https://doi.org/10.1109/TIFS.2020.2972166
Private and Secure Distributed Matrix Multiplication with Flexible Communication Load

Maliche Aliasgari*, Osvaldo Simeone† and Jürg Kliewer*
* Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ, U.S.A.
† Department of Engineering, King’s College London, Department of Engineering, London, U.K.

Abstract—Large matrix multiplications are central to large-scale machine learning applications. These operations are often carried out on a distributed computing platform with a master server and multiple 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, yielding a trade-off between recovery threshold, i.e., the number of workers required to recover the matrix product, and communication load, i.e., the total amount of data to be downloaded from the workers. In this paper, in addition to exact recovery requirements, we impose security and privacy constraints on the data matrices, and study the recovery threshold as a function of the communication load. We first assume that both matrices contain private information and that workers can collude to eavesdrop on the content of these data matrices. For this problem, we introduce a novel class of secure codes, referred to as secure generalized PolyDot (SGPD) codes, that generalize state-of-the-art non-secure codes for matrix multiplication. SGPD codes allow a flexible trade-off between recovery threshold and communication load for a fixed maximum number of colluding workers while providing perfect secrecy for the two data matrices. We then study a connection between secure matrix multiplication and private information retrieval. We specifically assume that one of the data matrices is taken from a public set known to all the workers. In this setup, the identity of the matrix of interest should be kept private from the workers. For this model, we present a variant of generalized PolyDot codes that can guarantee both secrecy of one matrix and privacy for the identity of the other matrix for the case of no colluding servers.

Index Terms—Coded distributed computation, distributed learning, secret sharing, information theoretic security, private information retrieval.

I. INTRODUCTION

A. Motivation and Problem Definition

At the core of many signal processing and machine learning applications are tensor operations, most notably large matrix multiplications [2]. In the presence of practically sized data sets, such operations are typically carried out using distributed computing platforms with a master server and multiple workers that can operate in parallel over distinct parts of the data set. The master server plays the role of the parameter server, distributing data to the workers and periodically reconciling their internal state [3]. Workers are commercial off-the-shelf servers that are characterized by possible temporary failures and delays [4].

Straggling workers can affect the computation latency by orders of magnitude, e.g., [5], [6]. While current distributed computing platforms conventionally handle straggling servers by means of replication of computing tasks [7], recent work has shown that encoding the input data can help reduce the computation latency. More generally, coding is able to control the trade-off between computational delay and communication load between workers and master server [8]–[17]. Furthermore, stochastic coding can help keeping both input and output data secure from the workers, assuming that the latter are honest, i.e., carrying out the prescribed protocol, but curious [18]–[25]. This paper contributes to this line of work by investigating the trade-off between computational delay and communication load as a function of the privacy level.

As illustrated in Figs. 1 and 2, we focus on the basic problem of computing a matrix multiplication $C = AB$ in a distributed computing system of $P$ workers that can process each only a fraction $1/m$ and $1/n$ of matrices $A$ and $B$, respectively. In the first setup under study, illustrated in Fig. 1 both matrices $A$ and $B$ are to be kept private from the workers. Here, three performance criteria are of interest:

- the recovery threshold $P_R$, that is, the number of workers that need to complete their task before the master server can recover the product $C$;
- the communication load $C_L$ between workers and master server, i.e., the amount of information to be downloaded from the workers;
- the maximum number $P_C$ of colluding servers that ensures perfect secrecy for both data matrices $A$ and $B$.

In the second setup of interest shown in Fig. 2 only matrix $A$ is private, while matrix $B$ is selected from a public data set $B$. In this case, apart from the security constraint on $A$, we only impose a privacy constraint on the identity of the specific matrix $B \in B$ of interest. As a motivation for this second setup, consider a recommender system based on collaborative filtering [26]. In this case, recommendations are based on the product of two matrices, one describing the profile of a user, or a group of users, and one representing features of the items of interest, such as movies, music or TV shows. The users’ profile matrix can be modelled by the private matrix...
A, hence ensuring the privacy of users’ data; while the items’
data matrix for each category is represented by one of the
matrices in the public data set $B = \{B^{(k)}\}_{k=1}^2$. This latter
assumption captures the constraint that users may want to keep
the confidential types of items they are interested in. For this
problem, the criteria of interest are still $P_R$ and $P_C$, and we
simplify the problem by setting $P_C = 1$. This paper focuses
on the design of coding and computing techniques for both
problems.

B. Related Work

In order to put our contribution in perspective, we briefly
review prior related work. Consider first solutions that provide
no security guarantees, i.e., $P_C = 0$, for the problem in Fig. 1
As a direct extension of [8], a first approach is to use product
codes that apply separately the maximum distance separable
(MDS) codes to encode the two matrices [27]. The recovery
threshold of this scheme is improved by [9], which introduces
polynomial codes. The construction in [9] is proved to be
optimal under the assumption that minimal communication
is allowed between workers and master server. In [15], MatDot
codes are introduced, resulting in a lower recovery threshold at
the expense of a larger communication load. The construction
in [15] bridges the gap between polynomial and MatDot codes
and proves improved recovery thresholds of PolyDot codes, yielding a trade-off between
recovery threshold and communication load. An extension
of this scheme, termed Generalized PolyDot (GPD) codes
improves on the recovery threshold of PolyDot codes [14], which
is independently obtained also by the construction in [28]. In
[14], GPD codes are used to design a unified coded computing
strategy for the training of deep neural networks.

Much less work has been done in the literature for the
case in which security constraints are factored in, i.e., where
$P_C \neq 0$, for the problem of Fig. 1. In [19], Lagrange coding
is presented that achieves the minimum recovery threshold for
multilinear functions by generalizing MatDot codes. In [18],
coded schemes have been used to develop multi-party
computation techniques to calculate arbitrary polynomials of
massive matrices, preserving the security of the data matrices.
In [20], [21], [22] a reduction of the communication load is
obtained by extending polynomial codes. While these works
focus on either minimizing recovery threshold or commu-
nication load, the trade-off between these two fundamental
quantities has not been addressed in the open literature to
the best of our knowledge. A new class of secure distributed
matrix multiplication and its capacity is studied in [29].

In the second part of this work, we study a connection
between secure matrix multiplication and private information
retrieval (PIR), as illustrated in Fig. 2. The PIR problem was
introduced in [30] and has been widely studied in recent years,
e.g., in [31]–[40]. In [38] and [39] the PIR setup was inves-
tigated for the problem of distributed matrix multiplication
illustrated in Fig. 2 that imposes PIR guarantees for the index
of matrix $B$ within a public library. In [38], a coding strategy
is proposed that combines the PIR scheme for non-colluding
 servers (i.e., with $P_C = 1$) [30] with polynomial codes [9]. In
[39], the authors introduce a related approach for this problem,
and show that it outperforms the scheme proposed in [38] in
terms of upload and download cost. The code design in [39]
focuses on the minimization of the communication load, and
does not explore the trade-off between this metric and the
recovery threshold.

C. Main Contribution

In this paper we first present a novel class of secure computation codes, referred to as secure GPD (SGPD) codes,
for the setup in Fig. 1. SGPD codes generalize GPD codes to operate at a flexible communication load level. This yields
a new achievable trade-off between recovery threshold $P_R$
and communication load $C_L$ as a function of a prescribed
number of colluding workers $P_C$. In the process, we also
introduce a novel perspective on distributed computing codes
based on the signal processing concepts of convolution and $z$-
transform. SGPD codes were first introduced in the conference
version of this paper [1], which did not contain complete
proofs and provided only limited illustrations and examples.
Then, SGPD codes are modified to offer a solution, introduced
here for the first time, for the scenario in Fig. 2. This is
done through concatenation with the PIR code in [38], which
ensures both secrecy of the input matrix $A$ and privacy of the
identity for the desired matrix in the library $B$ if $P_C = 1$.
The resulting codes are referred to as private and secure
GPD (PSGPD) codes. They generalize the approach in [39],
ensuring a trade-off between (upload) communication load and recovery threshold. We finally illustrate the benefits of
the proposed codes, which offer a flexible trade-off between
communication load and recovery threshold, by analyzing the
overall completion time due to both computation and
communication.

D. Organization

The rest of the paper is organized as follows. In Section II
we present the system models for secure matrix multiplication
(Fig. 1 in Section II-C) and for private and secure matrix
multiplication (Fig. 2 in Section II-D), respectively. In Section
III we propose an intuitive interpretation of the GPD code
introduced in [15]. Using $z$-transforms, Section IV proposes
a novel extension of GPD codes by imposing a security
constraint on the data matrices and deriving the resulting trade-off
between recovery threshold $P_R$ and communication load
$C_L$. In this section, we also study overall completion latency
encompassing both computation and communication latencies
for SGPD codes. In Section V we address the setup in Fig. 2
again with respect to the trade-off between $P_R$ and $C_L$, and
to the overall completion latency. The paper is concluded in
Section VI.

II. PROBLEM STATEMENT

A. Notation

Throughout the paper, we denote a matrix with upper
boldface letters (e.g., $\mathbf{X}$), and lower boldface letters indicate
a vector or a sequence of matrices (e.g., $\mathbf{x}$). Furthermore, a math
calligraphic font refers to a set (e.g., $\mathcal{X}$). A set $\mathcal{F}$ represents
the Galois field with cardinality $|\mathcal{F}|$. We denote by $\mathbb{N}$ the set of
all non-zero positive integers, and for some $a,b \in \mathbb{N}$, $a \leq b,$
The workers are honest but curious. Accordingly, we impose the secrecy constraint that, even if up to \( P \) workers collude, the workers cannot obtain any information about both matrices \( A \) and \( B \) based on the data received from the master server.

To keep the data secure and to leverage possible computational redundancy at the workers (namely, if \( P/m > 1 \) and/or \( P/n > 1 \), the master server sends encoded versions of the input matrices to the workers due to the above mentioned communication and complexity constraints. Specifically, it produces the encoded matrices \( A_p = f_p(A, R) \), where \( R \) is a random matrix of dimension \( T' \times S' \), for some integers \( T' \) and \( S' \) to be defined below, via the function

\[
f_p : \mathbb{F}^{T \times S} \times \mathbb{F}^{T' \times S'} \rightarrow \mathbb{F}^{T/t \times S/s},
\]

for some integers \( t \) and \( s \) such that \( m = st \). The resulting \( TS/m \) entries in the output of function \( f_p \) are then sent to worker \( p \), with \( p \in [1, P] \). Likewise, the master server computes the encoded matrices \( B_p = g_p(B, R') \), where \( R' \) is a random matrix of dimension \( S' \times D' \), for some integers \( S' \) and \( D' \) to be defined below, using the function

\[
g_p : \mathbb{F}^{S \times D} \times \mathbb{F}^{S' \times D'} \rightarrow \mathbb{F}^{S/s \times D/d},
\]

for some integers \( s \) and \( d \) such that \( n = sd \). The resulting \( SD/n \) entries in \( B_p \) are then sent to worker \( p \). The random matrices \( R \) and \( R' \) consists of i.i.d. uniformly distributed entries from a field \( \mathbb{F} \). The security constraint imposes the condition

\[
I(A_p, B_p; A, B) = 0,
\]

for all subsets of \( P \subset [1, P] \) of \( P_C \) workers, where the random matrices \( R \) and \( R' \) serve as random keys in order to meet the security constraint (3) [31].

Each worker \( p \) computes the product \( C_p = A_p B_p \) of the encoded sub-matrices \( A_p \) and \( B_p \). The master server collects a subset of \( P_R \leq P \) outputs from the workers as defined by the subset \( \{C_p\}_{p \in P_R} \) with \( |P_R| = P_R \). It then applies a decoding function as

\[
h : \mathbb{F}^{T/t \times D/d} \times \cdots \times \mathbb{F}^{T/t \times D/d} \rightarrow \mathbb{F}^{T \times D}.
\]

Note that correct decoding translates into the condition

\[
H(AB | \{C_p\}_{p \in P_R}) = 0.
\]

A coding and decoding strategy that satisfies condition (3) and (5) is said to be feasible.
D. Private and Secure Matrix Multiplication

In this subsection, we discuss the private and secure matrix multiplication problem illustrated in Fig. 2. In this setup, the master server wishes to compute the product \( C^{(\kappa)} = A^{(\kappa)} B^{(\kappa)} \) of a confidential input matrix \( A^{(\kappa)} \) with a matrix \( B^{(\kappa)} \) from a set of public matrices \( \{B^{(1)}, \ldots, B^{(L)}\} \), while keeping the index \( \kappa \) of the matrix \( B^{(\kappa)} \) of interest private from the workers.

Similar to the secure model in Fig. 1, we consider a distributed computing system with a master server and \( P \) honest but curious workers. The master server contains a confidential data matrix \( A \) with dimension \( T \times S \). Each worker has access to the library \( B \), which consists of \( L \) distinct matrices \( \{B^{(1)}, \ldots, B^{(L)}\} \), each with dimension \( S \times D \). As above, all matrices contain data symbols chosen uniformly i.i.d. from a sufficient large finite field \( \mathbb{F} \), with \( |\mathbb{F}| > P \). The master server is interested in computing the matrix product \( C^{(\kappa)} = AB^{(\kappa)} \) of the data matrix \( A \) and of a matrix \( B^{(\kappa)} \) for some index \( \kappa \) \( \in [1, L] \). This should be done while keeping the data matrix \( A \) secret against the workers in the same sense as in the scenario of Fig. 1 while also ensuring that the index \( \kappa \) is kept secret from the workers.

To do so, as in the PIR problem [33], [34], the master server generates \( P \) query vectors \( q^{(\kappa)}_1, \ldots, q^{(\kappa)}_P \in \mathbb{F}^L \), for some index \( \kappa \) \( \geq 1 \) as a function of the desired index \( \kappa \) and sends each worker \( p \in [1, P] \), the query vector \( q^{(\kappa)}_p \). We assume that the workers do not collude, i.e., we set \( P_C = 1 \). Extensions to any \( P_C > 1 \) are possible and are left for future work. We note that, when the input matrix \( A \) is an identity matrix, the setup reduces to a PIR problem.

To keep the data matrix \( A \) secure against workers, the master server sends each worker \( p \in [1, P] \) an encoded version \( A^{(\kappa)}_p = f_p(\kappa, A, R) \in \mathbb{F}^{T_L \times S} \), which is a function of index \( \kappa \), and through it, of the query \( q^{(\kappa)}_p \), of the data matrix \( A \) and of a random matrix \( R \), for some integers \( t \) and \( s \) such that \( m = ts \).

Upon receiving \( (q^{(\kappa)}_p, A^{(\kappa)}_p) \), each worker \( p \) uses the query \( q^{(\kappa)}_p \) to derive an \( S \times S \times D \times d \) matrix \( A^{(\kappa)}_p = g_p(q^{(\kappa)}_p, B) \in \mathbb{F}^{S \times S \times D \times d} \) from the library \( B \) by using an encoding function

\[
g_p : \mathbb{F}^L \times \mathbb{F}^{S \times S \times D \times d} \rightarrow \mathbb{F}^{S \times S \times D \times d}, \tag{7}
\]

for some integers \( s \) and \( d \) such that \( n = sd \). We emphasize that, unlike the setup considered in Fig. 1, the content of the desired matrix \( B^{(\kappa)} \) is not secure against workers, since the library \( B \) is public. Each worker \( p \) then computes the product \( C^{(\kappa)}_p = A^{(\kappa)}_p B^{(\kappa)} \) and sends it to the master server. The master server collects a subset \( \{C^{(\kappa)}_p\}_{p \in P_R} \) of \( P_R \leq P \) outputs from the workers with \( |P_R| = P_R \). It then applies a decoding function \( h(\{C^{(\kappa)}_p\}_{p \in P_R}) \), as in [4], in order to retrieve the desired product \( C^{(\kappa)} = AB^{(\kappa)} \).

To guarantee the secrecy of input matrix \( A \), in a manner similar to [5], we have the constraint

\[
I(A^{(\kappa)}_p, B^{(\kappa)}_p, q^{(\kappa)}_p; A) = 0, \tag{8}
\]

for all \( p \in [1, P] \). Following the PIR formulation on [38], in order to ensure the privacy of index \( \kappa \), for some value of \( \kappa \) the information available at each worker should be statistically indistinguishable from that available for any other value \( \kappa' \neq \kappa \). Mathematically, for all \( \kappa, \kappa' \in [1, L] \) with \( \kappa' \neq \kappa \) and for all workers \( p \in [1, P] \), we have the condition

\[
(q^{(\kappa)}_p, A^{(\kappa)}_p, C^{(\kappa)}_p, B) \sim (q^{(\kappa')}_p, A^{(\kappa')}_p, C^{(\kappa')}_p, B), \tag{9}
\]

that is, the joint distribution of variables \( (q^{(\kappa)}_p, A^{(\kappa)}_p, C^{(\kappa)}_p, B) \) should be the same for any pair of values \( \kappa, \kappa' \).
of index values $\kappa' \neq \kappa$. Finally, the correct decoding requirement is defined as in (5), that is

$$H(AB^{(\kappa)}(C_P^{(\kappa)})_{P \in \mathcal{P}_R}) = 0.$$  

(10)

A coding and decoding strategy that satisfies conditions (8), (9), and (10) is said to be feasible. For given parameters $m$ and $n$ the performance is measured by the pair $(P_R, C_L)$, with $P_C = 0$, where $C_L$ is the communication load defined in (6).

III. BACKGROUND: GENERALIZED POLYDOT CODE

WITHOUT SECURITY CONSTRAINT

In this section, we consider the system model shown in Fig. 1 and review the GPD construction first proposed in [15] and later improved in [14], [28] for the special case of no secrecy constraints, i.e., $P_C = 0$. In the process, we propose a novel intuitive interpretation of GPD encoding and decoding based on the distributed computation of samples from convolutions via z-transforms.

We start by recalling that the GPD coding scheme achieves the best currently known trade-off between recovery threshold $P_R$ and communication load $C_L$ for $P_C = 0$, i.e., under no security constraint. The entangled polynomial codes of [28] have the same properties in terms of $(P_R, P_C)$. The GPD codes for $P_C = 0$ also achieve the optimal recovery threshold among all linear coding strategies in the cases of $t = 1$ or $d = 1$, also they minimize the recovery threshold for the minimum communication load $C_{L,\text{min}}$ [9], [28].

The GPD code splits the data matrices $A$ and $B$ both horizontally and vertically as

$$A = \begin{bmatrix}
A_{1,1} & \cdots & A_{1,s} \\
:\ & \ddots & : \\
A_{t,1} & \cdots & A_{t,s}
\end{bmatrix}, \quad B = \begin{bmatrix}
B_{1,1} & \cdots & B_{1,d} \\
:\vdots & \ddots & \vdots \\
B_{s,1} & \cdots & B_{s,d}
\end{bmatrix}.$$  

(11)

The parameters $s$, $t$, and $d$ can be set arbitrarily under the constraints $m = ts$ and $n = sd$. Note that polynomial codes set $s = 1$, while MatDot codes have $t = d = 1$ [15]. All submatrices $A_{i,j}$ and $B_{k,l}$ have dimensions $T/t \times S/s$ and $S/s \times D/d$, respectively. The GPD code computes each block $(i, j)$ of the product $C = AB$, namely $C_{i,j} = \sum_{k=1}^{s} A_{i,k} B_{k,j}$, for $i \in [1, t]$ and $j \in [1, d]$, in a distributed fashion. This is done by means of polynomial encoding and decoding interpolation. As we review next, the computation of block $C_{i,j}$ can be interpreted as the evaluation of the middle sample of the convolution $c_{i,j} = a_{i} * b_{j}$ between the block sequences $a_i = \{A_{i,1}, \ldots, A_{i,s}\}$ and $b_j = \{B_{s,1}, \ldots, B_{s,d}\}$. In fact, the $s$th sample of the block sequence $c_{i,j}$ equals $C_{i,j}$, i.e., $[c_{i,j}]_s = C_{i,j}$. The computation is carried out distributively in the frequency domain by using z-transforms with different workers being assigned distinct samples in the frequency domain.

To elaborate, define the block sequence $a$ obtained by concatenating the block sequences $a_i$ as $a = \{a_1, a_2, \ldots, a_t\}$. Pictorially, a sequence $a$ is obtained from the matrix $A$ by reading the blocks in the left-to-right top-to-bottom order, as seen in Fig. 3. We also introduce the longer time block sequence $b$ as

$$b = \{b_1, 0, b_2, 0, \ldots, b_d\},$$  

(12)

with 0 being a block sequence of $s(t^* - 1)$ all-zero block matrices with dimensions $S/s \times D/d$. The sequence $b$ can be obtained from the matrix $B$ by following the bottom-to-top left-to-right order shown in Fig. 3 and by adding the all-zero block sequences between any two columns of the matrix $B$.

In the frequency domain, the z-transforms of sequences $a$ and $b$ are obtained as

$$F_a(z) = \sum_{r=0}^{ts-1} [a]_{r+1} z^r = \sum_{i=1}^{t} \sum_{j=1}^{s} A_{i,j} z^{s(i-1)+j-1},$$  

(13)

$$F_b(z) = \sum_{r=0}^{s-1+t^*(d-1)} [b]_{r+1} z^r = \sum_{k=1}^{d} \sum_{l=1}^{d} B_{k,l} z^{s(k-1)+t(l-1)},$$  

(14)

respectively. The master server evaluates the polynomials $F_a(z)$ and $F_b(z)$ in $P$ non-zero distinct points $z_1, \ldots, z_P \in \mathbb{F}$ and sends the corresponding linearly encoded matrices $A_p = F_a(z_p)$ and $B_p = F_b(z_p)$ to server $p$. The encoding functions are hence given by the polynomial evaluations (13) and (14), for $z_1, \ldots, z_P$. Server $p$ computes the multiplication $F_a(z_p)F_b(z_p)$ and sends it to the master server. The master...
server computes the inverse z-transform for the received products \( \{A_pB_p\}_{p \in P_R} = \{F_a(z_p)F_b(z_p)\}_{p \in P_R} \), obtaining the convolution \( a \ast b \).

From the convolution \( a \ast b \) we can see that the master server is able to compute all the desired blocks \( C_{i,j} \) by reading the middle samples of the convolutions \( c_{i,j} = a \ast b \) from samples of the sequence \( c = a \ast b \) in the order \( [c]_{s-1} = C_{1,1}, [c]_{s-1} = C_{2,1}, \ldots, [c]_{s-1} = C_{i,1}, [c]_{s-1+t^*s} = C_{1,2}, \ldots, [c]_{s-1+t^*s} = C_{t,2}, \ldots \). Note that, in particular, the zero block subsequences added to sequence \( b \) ensure that no interference from the other convolutions, \( c_{i',j'} \), affects the middle (\( s \))th sample of a convolution \( c_{i,j} \) with \( i' \neq i \) and \( j' \neq j \).

To carry out the inverse transform, the master server needs to collect as many values \( F_a(z_p)F_b(z_p) \) as there are samples of the sequence \( a \ast b \), yielding the recovery threshold
\[
P_R = tsd + s - 1.
\]
Equivalently, in terms of the underlying polynomial interpretation, the master server needs to collect a number of evaluations of the polynomial \( F_a(z)F_b(z) \) equal to the degree of \( F_a(z)F_b(z) \) plus one. This computation is of complexity order \( O(TD_P R \log(P_R))^2 \) [13]. Furthermore, the communication load is given as
\[
C_L = P_R \frac{TD}{td},
\]
where \( TD/(td) \) is the size of each matrix \( F_a(z)F_b(z) \).

IV. SECURE POLYDOT CODE

In this section, we propose a novel extension of the GPD code that is able to ensure the secrecy constraint for any \( C_P < P \). We also derive the corresponding achievable set of triples \( (P_C, P_R, C_L) \). As we will discuss, the projection of this set onto the plane defined by the condition \( P_C = 0 \) includes the set of pairs \( (P_R, C_L) \) in [15] and [16] obtained by the GPD code [14]. The proposed secure GPD (SGPD) code augments matrices \( A \) and \( B \) by adding \( P_C \) random block matrices to the input matrices \( A \) and \( B \), in a manner similar to prior works [18]–[21], [23], yielding augmented matrices \( A^* \) and \( B^* \). As we will see, a direct application of the GPD codes to these matrices is suboptimal.

In contrast, we propose a novel way to construct sequences \( a^* \) and \( b^* \) from matrices \( A^* \) and \( B^* \) that enables the definition of a more efficient code by means of the z-transform approach discussed in the previous section. To this end, we follow the design criterion of decreasing the recovery threshold \( P_R \) for a given communication load \( C_L \). Based on the discussion in the previous section, this goal can be realized by decreasing the length of the sequence \( c^* = a^* \ast b^* \), which in turn can be ensured by reducing the length of the sequence \( b^* \) for a given length of the sequence \( a^* \). We accomplish this objective by (i) adaptively appending rows or columns with random elements to matrix \( A \), and, correspondingly columns or rows to \( B \), which can reduce the recovery threshold; and (ii) modifying the zero padding procedure (see Fig. 3) for the construction of sequence \( b^* \). In order to account for point (i), we consider separately the two cases \( s < t \) and \( s \geq t \).

A. Secure Generalized PolyDot Code: The \( s < t \) Case

As illustrated in Fig. 4 when \( s < t \), we augment the input matrices \( A \) and \( B \) by adding
\[
\Delta_{P_C} \triangleq \begin{bmatrix} P_C \\ s \end{bmatrix},
\]
random row and column blocks to matrices \( A \) and \( B \), respectively. Accordingly, the \( t^* \times s \) augmented block matrix \( A^* \) with \( t^* = t + \Delta_{P_C} \) is obtained as
\[
A^* = \begin{bmatrix} A_{1,1} & \cdots & A_{1,s} \\ \vdots & \ddots & \vdots \\ A_{t,1} & \cdots & A_{t,s} \\ R_{1,1} & \cdots & R_{1,s} \\ \vdots & \ddots & \vdots \\ R_{s,1} & \cdots & R_{s,1} + \Delta_{P_C} \\ \vdots & \ddots & \vdots \\ R_{s,t,1} & \cdots & R_{s,t,1} + \Delta_{P_C} \end{bmatrix},
\]
while the \( s \times d^* \) augmented matrix \( B^* = [B \ R'] \) with \( d^* = d + \Delta_{P_C} \) is obtained as
\[
B^* = \begin{bmatrix} B_{1,1} & \cdots & B_{1,d} & R_{s,1} & \cdots & R_{s,1} + \Delta_{P_C} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ B_{s,1} & \cdots & B_{s,d} & R_{1,1} & \cdots & R_{1,1} + \Delta_{P_C} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots & \ddots & \ddots & \ddots \end{bmatrix}.
\]
In (18) and (19), if $s$ divides $P_C$, all block matrices $R_{i,j} \in \mathbb{F}_c^{\frac{P}{s} \times \frac{R}{s}}$ and $R_{i,j}^* \in \mathbb{F}_c^{\frac{P}{s} \times \frac{R}{s}}$ are generated with i.i.d. uniform random elements in $\mathbb{F}$. Otherwise, if $\Delta_{P_C}/P_C > s > 0$, the last $s \Delta_{P_C} - P_C$ matrices in (18), with right-to-left ordering in the last row of $R_{i,j}$, and in (19) with top-to-bottom ordering in the last column of $R_{i,j}^*$, are all-zero block matrices.

As illustrated in Fig. 4 in the SGPD scheme, the block sequence $a^*$ is defined in the same way as in the conventional GPD, yielding

$$a^* = \{a_1, \ldots, a_r, r_1, \ldots, r_{\Delta_{P_C}}\},$$

where $r_i$ is the $i$th row of the block matrix $R_i$, $i \in [1, \Delta_{P_C}]$. We also define the time block sequence $b^* = \{b, r^j\}$ as

$$b^* = \{b_1, 0, b_2, 0, \ldots, b_d, 0, r_1', r_2', \ldots, r_{\Delta_{P_C}}\},$$

where 0 is block sequences of $s(t^* - 1)$ all-zero block matrices, respectively, with dimensions $S/s \times D/d$, while $r_1'$ is the $j$th column of the random matrix $R'$. The key novel idea of this construction is that no zero matrices are introduced between the columns of matrix $R'$. As shown in Theorem 1 below, this construction allows the master server to recover all the desired submatrices $C_{i,j}$ for $i \in [1, t]$ and $j \in [1, d]$ from the middle samples of the convolutions $c_{i,j} = a_i \ast b_j$ (see Fig. 5 for an illustration).

**Theorem 1.** For a given security level $P_C < P$, the proposed SGPD code achieves the recovery threshold $P_R$

$$P_R = \begin{cases} tsd + s - 1, & \text{if } P_C = 0, \\ ts(d + 1) + s - \Delta_{P_C} - 1, & \text{if } P_C \geq 1 \text{ and } \Delta_{P_C} = P_C, \\ ts(d + 1) + s - \Delta_{P_C} + 2P_C - 1, & \text{if } P_C \geq 1 \text{ and } \Delta_{P_C} > P_C. \end{cases}$$

and the communication load (16), where $t^* = t + \Delta_{P_C}$ and $d^* = d + \Delta_{P_C}$ for any integer values $t, s$, and $d$ such that $s < t$, $m = ts$, and $n = sd$.

**Proof.** The $z$-transform of sequences $a^*$ and $b^*$ are given respectively as

$$F_{a^*}(z) = \sum_{i=1}^{t} \sum_{j=1}^{s} A_{i,j}^* z^{s(i-1)+(j-1)} \pm \sum_{i=1}^{t} \sum_{j=1}^{s} A_{i,j}^* z^{s(i-1)+(j-1)},$$

and

$$F_{b^*}(z) = \sum_{k=1}^{d} \sum_{l=1}^{d^*} B_{k,l}^* z^{s-k+t^*s(l-1)} \pm \sum_{k=1}^{d} \sum_{l=1}^{d^*} B_{k,l}^* z^{s-k+t^*s(l-1)}.$$
needs to collect a number of values of the product $F_a(z)F_b(z)$ equal to the length of the sequence $c^*$, which can be computed as the degree \(\deg(F_a(z)F_b(z))\) + 1, where \(\deg(F_a(z)F_b(z))\) is
\[
\begin{aligned}
& \begin{cases}
  t^*s(d + 1) + s\Delta_{P_C} - 1, & \text{if } \Delta_{P_C} = \frac{P_C}{s}, \\
  dst^* - s\Delta_{P_C} + 2P_C + t - 2, & \text{if } \Delta_{P_C} > \frac{P_C}{s}.
\end{cases}
\end{aligned}
\]
(25)

For \(P_C \geq 1\) this implies the recovery threshold \(P_R\) in (22). The communication load \(C_I\) in (20) follows from the fact that there are \(TD/(td)\) entries in \(F_a(z)F_b(z)\) for all \(p \in [1, P_R]\).

The security constraint (3) can be proved in a manner similar to (20) by the following steps:

\[
I(A, B; A_P, B_P) = H(A_P, B_P) - H(A_P, B_P | A, B)
\]
(a) \(= H(A_P, B_P) - H(A_P, B_P | A, B) + H(A_P, B_P | A, B) - H(\{A_P, B_P, A_R, R_{PC}^1, \ldots, R_{PC}^t\} | A, B)\)
(b) \(= H(A_P, B_P) - H(\{R_{PC}^1, \ldots, R_{PC}^t\} | A, B)\)
(c) \(\leq H(A_P) + H(B_P) - \sum_{p=1}^{P_C} H(\{R_p\}) - \sum_{p=1}^{P_C} H(\{R_p^t\})\)
(d) \(\leq H(A_P) + H(B_P) - P_C \frac{TS}{m} \log |F| - P_C \frac{SD}{n} \log |F|\)
(e) \(\leq \sum_{p=1}^{P_C} H(A_p) + \sum_{p=1}^{P_C} H(B_p) - P_C \frac{TS}{m} \log |F| - P_C \frac{SD}{n} \log |F|\)
(f) \(\leq P_C \frac{TS}{m} \log |F| - P_C \frac{SD}{n} \log |F| - P_C \frac{TS}{m} \log |F| - P_C \frac{SD}{n} \log |F|\)
\]
= 0,
(26)

where (a) follows from the definition of encoding functions, since \(A_P\) is a deterministic function of \(A\) and \(R_P\), and \(B_P\) is a deterministic function of \(B\) and \(R_P^t\), respectively, for all \(p \in [1, P_C]\); (b) follows from (23) and (24), since from \(P_R\) polynomial evaluations \(A_P\) and \(B_P\) in (23) and (24) we can recover \(2P_C\) unknowns when the coefficients \(A_{ij}\) and \(B_{k,t}\) are known, given that we have \(P_R \geq 2P_C\); (c) and (d) follows since \(R_p\) and \(R_p^t\) are independent uniformly distributed entries; (e) follows by upper bounding the joint entropy using the sum of individual entropies; and (f) follows from an argument similar to (d). Hence, the proposed scheme is information-theoretically secure.

Remark 1. When \(P_C \geq 1\) a direct application of the GPD construction in Fig. 5 would yield the larger recovery threshold

\[
P_R = \begin{cases}
t^*sd^* + s - 1, & \text{if } \Delta_{P_C} = \frac{P_C}{s}, \\
dst^* + s - 1 - 2(s\Delta_{P_C} - P_C), & \text{if } \Delta_{P_C} > \frac{P_C}{s}.
\end{cases}
\]
(27)

B. Secure Generalized PolyDot Code: The \(s \geq t\) Case

As illustrated in Fig. 6 when \(s \geq t\), we instead augment input matrices \(A\) and \(B\) by adding

\[
\Delta_{P_C} \leq \frac{P_C}{\min\{t, d\}}
\]
(28)

column and row blocks to matrices \(A\) and \(B\). This can be seen to yield a smaller recovery threshold. Accordingly, the \(t \times s^*\) augmented block matrix \(A^* = [A \ R]\) with \(s^* = s + \Delta_{P_C}\) is obtained as

\[
A^* = \begin{bmatrix}
A_{1,1} & \ldots & A_{1,s} & R_{1,1} & \ldots & R_{1,\Delta_{P_C}} \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
A_{t,1} & \ldots & A_{t,s} & R_{t,1} & \ldots & R_{t,\Delta_{P_C}}
\end{bmatrix}
\]
(29)
while the \( s^* \times d \) augmented block matrix \( B^* \) is defined as

\[
B^* = \begin{bmatrix}
R' & \cdots & R'_{s-1, d} \\
R_{1,1} & \cdots & R_{1,d} \\
\vdots & \ddots & \vdots \\
B_{s,1} & \cdots & B_{s,d}
\end{bmatrix}
\]

(30)

As for (29) and (30), if \( \Delta P_C' = P_C/t \min \{t, d \} > 0 \), the last \( s\Delta P_C - P_C' \) block matrices in (29) with bottom-to-top right-to-left ordering in \( R \), and in (30) with right-to-left top-to-bottom ordering in \( R' \), are all-zero block matrices. The construction of sequences \( a^* \) and \( b^* \) is analogous to the GPD in the non-secure case. In particular, as seen in Fig. 6, the time block sequence \( a^* \)

\[
a^* = \{a_1, r_1, a_2, r_2, \ldots, a_t, r_t\},
\]

(31)

whereas the block sequence \( b^* \) is defined as

\[
b^* = \{b_1, 0, b_2, \ldots, 0, b_d, 0, r_1'_{s-1}, \ldots, r_1'\}.
\]

(32)

Here, 0 and \( 0 \) are a block sequence of \( t \) and \( t - 1 \) all-zero block matrices with dimensions \( S/s \times D/d \), respectively, while \( r_1' \) is the \( t \)th row of the random matrix \( R' \).

**Theorem 2.** For a given security level \( P_C < P \), the proposed SGPD code achieves the recovery threshold

\[
P_R = t(s^* d - \Delta P_C) + ts + 2P_C - 1
\]

(33)

and the communication load (16), where \( s^* \) is defined as \( s^* = s + \Delta P_C \), for any integer values \( t, s \), and \( d \) such that \( s \geq t, m = ts \), and \( n = sd \).

**Proof.** We define the \( z \)-transform of sequences \( a^* \) and \( b^* \) respectively as

\[
F_{a^*}(z) = \sum_{i=1}^{t} \sum_{j=1}^{s^*} A_{i,j} z^{i-1+t(j-1)} + \sum_{i=1}^{t} \sum_{j=1}^{s^*} A_{i,j} z^{i-1+t(j-1)},
\]

(34)

\[
F_{b^*}(z) = \sum_{k=1+\Delta P_C}^{s^* d} B_{k,l} z^{(s^* - k)t + ts^*(l-1)} + \Delta P_C - d \sum_{k=1}^{\Delta P_C} \sum_{l=1}^{d} B_{k,l} z^{t(s^* d - \Delta P_C) + d(\Delta P_C - k) + l - 1}.
\]

(35)

The \((i, l)\) block \( C_{i,l} = \sum_{r=0}^{s^* d} A_{i,r} B_{r,l} \), for all \( i \in [1, t] \) and \( l \in [1, d] \), of matrix \( C = AB \) can be seen equal to the \((i - 1 + t(s^* l - 1))\)th sample of the convolution \( c^* = a^* * b^* \). The rest of the proof follows in a manner akin to Theorem 1.

**Remark 2.** The computational complexity of SGPD codes for both workers and master server can be summarized as follows. Each worker is assigned to compute the multiplication \( C_p = A_p B_p \), requiring \( TSD/(tsd) \) multiplications. For the master server, encoding matrices \( A_p \) and \( B_p \) at each worker amounts to evaluating \( z \)-transforms \( F_{a^*}(z) \) and \( F_{b^*}(z) \) at a random point \( z_p \). This requires multiplying \( z_p \) by \((ts + P_C)\) and \((sd + P_C)\) submatrices, each of dimension \( D/t \times D/s \) and \( D/s \times D/d \), respectively. This requires \( P_C / (ts + P_C) + (sd + P_C) \) multiplications. Overall, the master server requires to carry out \( P_C/ (ts + P_C) + (sd + P_C) \) multiplications. For decoding, the master server interpolates a polynomial degree \( P_R - 1 \) for each element in \( C \). Using a polynomial interpolation algorithm, the decoding complexity amounts to \((P_R - 1)(\log(P_R - 1))^2TD/(td) \) multiplications (42).

**Example 1.** We now provide some numerical results of the proposed SGPD scheme. We set \( P = 3000 \) workers and parameters \( m = n = 36 \). The trade-off between communication load \( C_L \) and recovery threshold \( P_R \) for both non-secure conventional GPD codes \( (P_C = 0) \) and proposed SGPD code with colluding workers \( P_C = 11 \) and \( P_C = 29 \) is illustrated in Fig. 7. The figure quantifies the loss in terms of achievable pairs \((P_R, C_L)\) that is caused by the security constraint.

**C. Trading Off Computation and Communication Latencies**

In this subsection, we elaborate on the importance of enabling a flexible trade-off between communication load and recovery threshold by analyzing the overall completion time for the matrix multiplication task at hand. The completion delay is the sum of latencies due to computation and communication.

To this end, following a well-established model (43), (11), we assume that computation at each worker \( p \) requires a random time \( T_{p}^{\text{comp}} \), measured in some specified unit of time, that is modeled as a shifted exponential distribution with cumulative distribution function (cdf)

\[
F_{\text{comp}}(T_{p}^{\text{comp}}) = 1 - \exp \left( -\frac{t_{sd}}{T_{p}^{\text{comp}} - T_{\text{min}}^{\text{comp}}} \right),
\]

(36)
for $T \geq T^{\text{comp}}_{\min}$ and $F^{\text{comp}}(T) = 0$ otherwise. According to \cite{36}, the parameter $T^{\text{comp}}_{\min}$ represents the minimum computing time, and $1/\mu$ represents the average excess computing time, with respect to $T^{\text{comp}}_{\min}$, per multiplication (recall Remark 2).

Assuming independent computing times, for a given recovery threshold $P_R$, the computation time $T^{\text{comp}}$ is hence given as the $P_R$th-order statistic, i.e., the $P_R$th smallest variable, among the i.i.d. variables $(T_1^{\text{comp}}, \ldots, T_P^{\text{comp}})$. Its expectation is given by \cite{44}

$$E[T^{\text{comp}}] = \frac{tsd}{\mu TSD} \sum_{i=1}^{P_R} \frac{1}{P - P_R + i} = \frac{tsd}{\mu TSD} (H_P - H_{P-P_R}),$$

where $H_P$ is the generalized harmonic number defined as $H_P = \sum_{i=1}^{P} 1/i$.

Suppose now that the workers communicate with the master server are in a link with an overall download rate $R^{\text{comm}}$ (symbols per unit time). The communication latency is hence given as

$$T^{\text{comm}} = P_R \frac{TD}{tsdR^{\text{comm}}},$$

since the workers need to return $P_R TD/(td)$ symbols to the master server. Overall, the average completion time is given as

$$E[T] = T^{\text{comp}}_{\min} + \frac{tsd}{\mu TSD} (H_P - H_{P-P_R}) + P_R \frac{TD}{tsdR^{\text{comm}}}. \quad (39)$$

**Example 2.** Let consider $P = 3000$ workers and parameters $m = n = 36$. We assume that $P_C = 29$, $T = S = D = 1008$, $\mu = 0.5 \times 10^{-4}$, and $T^{\text{comp}}_{\min} = 1$. We compare the performance of the following SGPD codes: (i) $t = d = 36$ and $s = 1$ (secure Polynomial code); (ii) $t = s = d = 6$; (iii) $t = d = 1$ and $s = 36$ (secure MatDot code). The values of $C_L$ and $P_R$ for these codes are shown in Fig. [2]. The average completion time $E[T]$ is plotted versus the communication rate $R^{\text{comm}}$ in Fig. [8]. The figure shows that the optimal choice of the latency-minimizing SGPD code along the curve in Fig. [7] depends on the system’s operating point: For small communication rates, it is preferable to reduce the communication load $C_L$, and hence secure Polynomial codes are the best choice; while for large communication rate, it is optimal to choose codes with an increasingly large value of the communication load $C_L$.

V. SECURE AND PRIVATE GENERALIZED POLYDOT CODE

In this section, we study the setup shown in Fig. [2] We propose a variant of the private and secure GPD code introduced in \cite{38} that we refer to as private and secure GPD (PSGPD) code. Note that in \cite{38} a private coded matrix multiplication scheme is proposed only for Polynomial codes with $s = 1$ in \cite{11}. We derive the corresponding achievable set of pairs $(P_R, C_L)$ as defined in Section II under the condition $P_C = 1$, i.e., the workers do not collude.

**Theorem 3.** For a given security level $P_C = 1$, there is an achievable PSGPD codes with the recovery threshold

$$P_R = \begin{cases} s(t+1)d, & \text{if } s < t, \\ ts(d+1) - t + 1, & \text{if } s \geq t, \end{cases} \quad (40)$$

and the communication load \cite{16}, for any integer values $t$, $s$, and $d$ such that $m = ts$, and $n = sd$.

**Proof.** The proof is presented in Appendix A. \hfill $\square$

**Remark 3.** The computational complexity of PSGPD codes for both workers and master server is summarized as follows. In PSGPD codes, each worker has two duties, namely encoding the library $B$ and computing the multiplication $C_p^{(s)} = A_p^{(s)} B_p^{(s)}$. Encoding the library, i.e., computing the matrix $B_p^{(s)}$ in \cite{44}, requires to evaluate $F_{2^{m-1}}(z), r \in [1, L]$ at query vector $q_v^{(r)}$. Hence, the former task requires LSD multiplications, while the latter entails $TSD/(ts)d$ multiplications. In total, each worker carries out LSD $+ TSD/(ts)d$ multiplications. The master server encodes matrix $A_k^{(s)}$ with $(1+ts)TSD/(ts)d$ multiplications. In total, for all $P$ workers, the master server needs $P(1+ts)TSD/(ts)d$ multiplications. The computation complexity of the decoding complexity of the master server is the same as for SGD codes, namely $\mathcal{O}((P_R - 1)(\log(P_R - 1))^2TSD/(td))$.

**Example 3.** Let us consider $P = 3000$ workers and parameters $m = n = 36$. We assume that $P_C = 1$ in order to compare the performance of proposed SGD and PSGPD codes. Note that both recovery threshold and communication load of the PSGPD code do not depend on the number of public matrices $|B| = L$ in the library. The trade-off between communication load $C_L$ and recovery threshold $P_R$ is illustrated in Fig. [8] for both codes. The figure shows that, for a fixed value of $P_R$, the resulting achievable value of the communication load $C_L$ is smaller for PSGPD than for SGD codes. This suggests that the privacy requirement on the index $\kappa$ imposed by PSGPD is less demanding than the security constraint on matrix $B$ under which SGD codes operate.
Remark 4. As for SGPD codes, the overall average completion time of PSGPD codes can be derived following the same steps as described in Section IV-C.

VI. CONCLUDING REMARKS

In this work, we have considered the problem of secure and private distributed matrix multiplication on \( C = AB \) in terms of design of computational codes for two settings. In the first setting, the two matrices \( A \) and \( B \) contain confidential data and must be kept secure from the workers; and in the second setting, matrix \( A \) is confidential, while matrix \( B \) is selected in a private manner from a library of public matrices. For both problems, this work presents the best currently known trade-off between communication load and recovery threshold. This is done by presenting two code constructions that generalize the state-of-the-art GPD codes \([13]–[15]\), in combination with PIR based codes \([38]\).

Among important items for future research, we mention the extension of the proposed PSGPD construction to \( P_C > 1 \). Here, we note that one can design an achievable PSGPD scheme for any arbitrary privacy level by trivially concatenating a robust PIR scheme for arbitrary colluding workers and private databases \([33]\) and the proposed SGPD code. However, this approach would require multiplying the data matrix \( A \) with all \( L \) public matrices in the set \( B = \{B^{(r)}\}_{r=1}^{L} \) for each worker \( p \in [1, P] \), implying a significantly increased computation load. Future work will focus on PSGPD schemes for any number of colluding workers that provides a smaller computational complexity at the workers. Finally, the establishment of a converse bound and the consideration of non-perfect communication channels between workers and master server \([45]\) are open problems.

APPENDIX A

PROOF OF THEOREM 3

We start by discussing the \( s < t \) case, as done in Section IV. The polynomial encoding function for the input matrix \( A \), is obtained as defined in \((23)\) for \( P_C = 1 \), that is

\[
F_A(z) = \sum_{i=1}^{t} \sum_{j=1}^{s} A_{i,j} z^{s(i-1)+(j-1)} + R z^{st},
\]

where we recall that \( R \) is an \( T/t \times S/s \) random matrix with i.i.d. uniform random elements in \( \mathbb{F} \). The encoded matrices are given as \( A_p^{(\kappa)} = F_A(z_{\kappa,p}) \) for values \( z_{\kappa,p} \) to be discussed below. For the desired index \( \kappa \), the master server also computes the query vector \( q_p^{(\kappa)} \) for all \( p \in [1, P] \). This is obtained as

\[
q_p^{(\kappa)} = [z_1, \ldots, z_{\kappa-1}, z_{\kappa,p}, z_{\kappa+1}, \ldots, z_L],
\]

where all points \( \{z_i\}_{i \neq \kappa} \) are selected uniformly i.i.d. from \( \mathbb{F} \) but are identical for all \( p \). The points \( \{z_{\kappa,p}\}_{p=1}^{P} \) are selected i.i.d. as distinct elements from \( \mathbb{F} \) (recall that we have \( |\mathbb{F}| > P \)). We note that, as in the PIR scheme \([38]\), the query vector \( q_p^{(\kappa)} \) does not leak any information on index \( \kappa \) in the sense defined by condition 9. The master server evaluates \( F_A(z) \) in \((41)\) at the distinct random point \( z_{\kappa,p} \) to produce the encoded matrices \( A_p^{(\kappa)} = F_A(z_{\kappa,p}) \), and then sends \( A_p^{(\kappa)} \) along with the query vector \( q_p^{(\kappa)} \) to worker \( p \in [1, P] \).

Each worker \( p \), after receiving the query vectors \( q_p^{(\kappa)} \), encodes the library \( B \) into a matrix \( B_p^{(\kappa)} \) as follows. Define the polynomial encoding function for each matrix \( B^{(r)} \), \( r \in [1, L] \), in the library \( B \) as in \((24)\) for \( P_C = 0 \), i.e.,

\[
F_{B^{(r)}}(z) = \sum_{k=1}^{s} \sum_{l=1}^{d} B_{k,l}^{(r)} z^{s-k+(l-1)(s+1)}.
\]

Each worker \( p \) computes the encoded matrices as

\[
B_p^{(\kappa)} = F_{B^{(r)}}([q_p^{(\kappa)}]_r)
\]

\[
= F_{B^{(r)}}(z_{\kappa,p}) + \sum_{r \in [1, L] \setminus \kappa} F_{B^{(r)}}(z_{r}),
\]

where \([q_p^{(\kappa)}]_r\) denotes the \( r \)th element of the query vector \( q_p^{(\kappa)} \).

After encoding the library, each worker \( p \) computes the matrix product \( C_p^{(\kappa)} = A_p^{(\kappa)} B_p^{(\kappa)} \) and then sends \( C_p^{(\kappa)} \) back to the master server. We note that both polynomials \( F_A(z) \) and \( F_{B^{(\kappa)}}(z) \), assigned to the input matrix \( A \) and the desired matrix \( B^{(\kappa)} \), are evaluated at the same random points \( z_{\kappa,1}, \ldots, z_{\kappa,P} \) for workers \( 1, \ldots, P \), respectively. Since each undesired matrix is evaluated at an identical random point for all workers the second term in \((44)\), i.e., \( \sum_{r \in [1, L] \setminus \kappa} F_{B^{(r)}}(z_{r}) \), can be considered as a constant term.

To reconstruct all blocks \( C_{i,k}^{(\kappa)} \) of the product matrix \( C^{(\kappa)} = A^{(\kappa)} B^{(\kappa)} \), the master server carries out polynomial interpolation, upon receiving a number of multiplication results equal to at least \( \deg(F_A(z)G_{B^{(\kappa)}}(z)) + 1 \), which is \( s(t+1)d \), for the case \( s < t \).

Similarly, for the \( s \geq t \) case, the polynomial encoding function for the input matrix \( A \) as in \((34)\) for \( P_C = 1 \), that is,

\[
F_A(z) = \sum_{i=1}^{t} \sum_{j=1}^{s} A_{i,j} z^{s(i-1)+(j-1)} + R z^{st},
\]
The encoded matrices $A_p$ and $B_p$ are defined as above, and so are the query vectors $q_p$ for all $p \in [1, P]$.

The security of the data matrix $A$ against non-colluding workers is guaranteed by appending the random matrix $R$ to the input matrix $A$ in (41) in the same way as described in Section [LV]. The details for both cases $s < t$ and $s \geq t$ are given in the proofs of Theorems 1 and 2, respectively, for the case of $P_C = 1$. The privacy condition of (9) follows by definition of the query vectors (62) for the desired index $\kappa \in [1, L]$, as proved in [38]. Finally, the recovery threshold and the communication load follow in a manner analogous to Theorems 1 and 2.

REFERENCES

[1] M. Aliasgari, O. Simeone, and J. Kliewer, “Distributed and private coded matrix computation with flexible communication load,” in Proc. IEEE Intern. Symp. Inform. Theory (ISIT), Jul. 2019, pp. 1092–1096.

[2] M. Janzamin, H. Sedghi, and A. Anandkumar, “Beating the perils of non-convexity: Guaranteed training of neural networks using tensor methods,” arXiv preprint, arXiv:1506.08473, 2015.

[3] Y. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su, “Scaling distributed machine learning with the parameter server,” in Proc. of the 11th USENIX Conference on Operating Systems Design and Implementation, OSDI, vol. 14, Oct. 2014, pp. 583–598.

[4] J. Dean and L. A. Barroso, “The tail at scale,” Communications of the ACM, vol. 56, no. 2, pp. 74–80, Feb. 2013.

[5] G. Joshi, E. Soljanin, and G. Wornell, “Efficient redundancy techniques for latency reduction in cloud systems,” ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS), vol. 2, no. 2, pp. 12:1–12:30, Apr. 2017.

[6] D. Wang, G. Joshi, and G. Wornell, “Using straggler replication to reduce latency in large-scale parallel computing,” ACM SIGMETRICS Performance Evaluation Review, vol. 43, no. 3, pp. 7–11, Dec. 2015.

[7] K.-H. Huang and J. A. Abraham, “Algorithm-based fault tolerance for matrix operations,” IEEE Trans. on Computers, vol. 100, no. 6, pp. 518–528, Jun. 1984.

[8] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, “Speeding up distributed machine learning using codes,” IEEE Trans. on Inform. Theory, vol. 64, no. 3, pp. 1514–1529, Aug. 2017.

[9] Q. Yu, M. Maddah-Ali, and S. Avestimehr, “Polynomial codes: an optimal design for high-dimensional coded matrix multiplication,” in Proc. Advances in Neural Inform. Processing Systems, Dec. 2017, pp. 4403–4413.

[10] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, “A fundamental tradeoff between computation and communication in distributed computing,” IEEE Trans. on Information Theory, vol. 64, no. 1, pp. 109–128, Sep. 2017.

[11] M. Aliasgari, J. Kliewer, and O. Simeone, “Coded computation against processing delays for virtualized cloud-based channel decoding,” IEEE Trans. on Commun., vol. 67, no. 1, pp. 28–38, Jan. 2019.

[12] ———, “Coded computation against straggling decoders for network function virtualization,” in Proc. IEEE Intern. Symp. Inform. Theory (ISIT), Jun. 2018, pp. 711–715.

[13] S. Dutta, M. Fahim, F. Haddadpour, H. Jeong, V. Cadambe, and P. Grover, “On the optimal recovery threshold of coded matrix multiplication,” arXiv preprint, arXiv:1801.10292, 2018.

[14] S. Dutta, Z. Bui, H. Jeong, T. M. Low, and P. Grover, “A unified coded deep neural network training strategy based on generalized polydot codes,” arXiv preprint, arXiv:1811.0751, 2018.

[15] M. Fahim, H. Jeong, F. Haddadpour, S. Dutta, V. Cadambe, and P. Grover, “On the optimal recovery threshold of coded matrix multiplication,” in Proc. 53th Allerton Conf. Commun., Control, Comput., IL, USA, Oct. 2017, pp. 1264–1270.