Secure MatDot codes for secure distributed matrix multiplication

Hiram H. López
Department of Mathematics and Statistics
Cleveland State University
Cleveland, Ohio 44115
Email: h.lopezvaldez@csuohio.edu

Gretchen L. Matthews
Department of Mathematics
Virginia Tech
Blacksburg, VA 24061
Email: gmatthews@vt.edu

Daniel Valvo
Department of Mathematics
Virginia Tech
Blacksburg, VA 24061
Email: vdaniel1@vt.edu

***Abstract*—The recently introduced MatDot codes achieve the optimal recovery threshold for distributed matrix multiplication schemes. This paper presents secure MatDot codes, a family of codes that support secure distributed matrix multiplication via a careful selection of evaluation points. Because of the threshold optimality on MatDot codes, there is no scheme where the user can recover AB after any $S-1$ servers have finished where S is the recovery threshold. However, under certain conditions, the secure MatDot code has the property that the user can also recover the matrix multiplication using less than S selected servers. Thus, the secure MatDot code adds an alternative way to compute the matrix multiplication by identifying the fastest servers in advance. The discrete Fourier transform codes have been recently studied as distributed matrix multiplication schemes that provide security against the user. We show that the secure MatDot codes may also provide protection against the user. We offer scenarios where the discrete Fourier transform code cannot be applied, but the secure MatDot may be utilized.**

I. Introduction

The goal of a T-secure distributed matrix multiplication (SDMM) scheme is to transmit from a *source node* to a *user* the product of two matrices A and B, using N *servers* to release the heavy multiplication duty so that even if T servers collude, no information about A or B is revealed.

In this paper, we employ locally recoverable codes (also known as codes with locality) in the polynomial code approach recently introduced in [1]. The core idea is the following. Consider matrices A and B defined over a finite field $\mathbb{F}_q$. Assume that A and B have been partitioned into smaller matrices $A_1, \ldots, A_{L_1}$ and $B_1, \ldots, B_{L_2}$, respectively, such that the product AB depends of the products A_iB_j. Let $R_1, \ldots, R_T, S_1, \ldots, S_T$ be random matrices such that the size of every R_i (resp., S_i) is the same as A_i (resp., B_i). The source node defines polynomials $f(x)$ and $g(x)$ to encode the information from the matrices A_i, R_i, and B_i, S_i, respectively. The SDMM scheme transmits to the servers the values $f(\alpha_i)$ and $g(\alpha_i)$ for certain $\alpha_i \in \mathbb{F}_q$. The servers send to the user the product $f(\alpha_i)g(\alpha_i)$. The user recovers the polynomial $h(x) := f(x)g(x)$, or part of it, which contains the desired matrix multiplication AB. We focus on the inner product partitioning given by $A = [A_1 \cdots A_L]$, and $B = [B_1 \cdots B_L]$, such that $AB = A_1B_1 + \cdots + A_LB_L$, where the products A_iB_i have the same size.

In terms of security, several schemes using polynomial codes have been proposed in the literature. For instance, in [2], [1], the authors assume that the source node and the user are the same, so both have access to the matrices A_i's and B_i's. Of course, there is extensive work on coded matrix multiplication, including [3], [4], [5], [6], [7], [8], [9].

This paper defines the secure MatDot codes. We consider codes that are T-secure with $T > 0$; the source node differs from the user. We say in this case that the scheme provides *security against the user* to mean the user cannot get any information about the matrices A_i's, B_i's, A, or B from any collection of T servers. We also consider the case where the source node, the servers, and the user have access to the matrices. This may represent the user using their servers to multiply A and B. Here, $T = 0$, and the source node equals the user. In this context, the MatDot codes developed in [10] outperform Polynomial codes [1] and the Algorithm-Based Fault Tolerance algorithm [11]. Even more, MatDot codes achieve the *optimal recovery threshold*, which is the minimum number of successful (non-delayed, non-faulty) processing servers required for completing the computation. We obtain an improved way to compute AB in a MatDot code by selecting adequate evaluation points. This alternative way depends on the identification of the fastest servers in advance. We compare the secure MatDot codes with the discrete Fourier transform (DFT) code [12]. As we see in Section III, there are settings in which the proposed secure MatDot codes can be used, but the DFT scheme cannot be applied.

This paper is organized as follows. This section concludes with preliminaries. Section II centers on the new code family. Examples are found in Section III, followed by a conclusion in Section IV

Preliminaries. We now define terms and helpful notation. Let $\mathbb{F}_q$ be the finite field with q elements. The set of matrices with entries in $\mathbb{F}_q$ of size $a \times b$ is denoted by $\mathbb{F}_q^{a\times b}$. A *symbol* is an element in $\mathbb{F}_q$. The following are important definitions for an SDMM scheme. The *Upload Cost* is the total number of

The work of Hiram H. López was supported in part by the AMS–Simons Travel Grant. The work of Gretchen L. Matthews was supported in part by NSF under Grant DMS-1855136 and in part by the Commonwealth Cyber Initiative. (Corresponding author: Hiram H. López.)

symbols that the scheme requires to be uploaded to all of the servers. The *Download Cost* is the total number of symbols that the scheme requires to be transmitted from all the servers to the user to calculate AB. The *Download Rate* is the ratio of the number of symbols contained in the result AB to the number of symbols the scheme requires to be downloaded, i.e $\frac{ac}{\text{Download Cost}}$. The *Total Cost* is the total number of symbols the scheme requires to be uploaded or downloaded in the entire process of calculating AB, i.e., the upload cost plus the download cost. The *Total Rate* $\mathcal{R}$ is the ratio of the number of symbols of information contained in the result AB to the total cost, i.e., $\mathcal{R} = \frac{ac}{\text{Total Cost}}$.

The MatDot and secure MatDot codes rely on Reed-Solomon codes. The Reed-Solomon code over $\mathbb{F}_q$ with evaluation set $\mathcal{S} = \{a_1, \ldots, a_n\} \subseteq \mathbb{F}_q$ and degree $k \in \mathbb{Z}^+$ is

$$\text{RS}(\mathcal{S}, k) := \{(f(a_1), \ldots, f(a_n)) \mid f \in \mathbb{F}_q[x]_{<k}\}$$

where $\mathbb{F}_q[x]_{<k} := \{f \in \mathbb{F}_q[x] : \deg(f) < k\}$. Write $ev(f) := (f(a_1), \ldots, f(a_n))$. If $\mathcal{S} = \mathbb{F}_q$, the code is denoted $\text{RS}(k)$.

Recall the dual of a code $\mathcal{C} \subseteq \mathbb{F}_q^n$ is

$$\mathcal{C}^\perp := \{\boldsymbol{b} \in \mathbb{F}_q^n \mid \boldsymbol{a} \cdot \boldsymbol{b} = 0, \text{ for all } \boldsymbol{a} \in \mathcal{C}\} \subseteq \mathbb{F}_q^n.$$

Note that $\text{RS}(k)^\perp = \text{RS}(n-k)$. It is convenient to write $[n] := \{1, \ldots, n\}$.

II. Secure MatDot codes

We define a T-secure SDMM scheme referred to as the *secure MatDot code*, inspired by the MatDot codes developed in [10], and the locally recoverable codes considered by Tamo and Barg [13]; see also [14]. For the rest of the section, assume that $q \geq 3L + 2T - 1$. We also consider that A and B are matrices with inner product partitionings $A = [A_1 \cdots A_L] \in \mathbb{F}_q^{a \times b}$ and $B^\intercal = [B_1^\intercal \cdots B_L^\intercal] \in \mathbb{F}_q^{c \times b}$ so that $AB = A_1B_1 + \cdots + A_LB_L$. Enumerate the elements of the field $\mathbb{F}_q = \{a_1, \ldots, a_n\}$, so $n := q$.

Theorem 2.1. *Take $F_1 := \{a_1, \ldots, a_L\} \subseteq \mathbb{F}_q$ and a polynomial $H \in \mathbb{F}_q[x]_{<n-2L-2T+1}$ such that $H(a) = \alpha \in \mathbb{F}_q \setminus \{0\}$ for all $a \in F_1$. Define $U := \{a_i \in \mathbb{F}_q : H(a_i) \neq 0\} \setminus F_1$. Consider there are $N = n - L$ servers, which are indexed by the field elements $a_{L+1}, \ldots, a_n$. Then there is a T-secure SDMM scheme, called the secure MatDot code, which determines the product $AB \in \mathbb{F}_q^{a \times b}$ by downloading the information from either the servers indexed by U, or any $2L + 2T - 1$ servers.*

Proof. Consider polynomials $f, g \in \mathbb{F}_q[x]$ such that $f(a_i) = A_i$ and $g(a_i) = B_i$, for $i \in [L]$; and $f(a_i) = R_{i-L}$ and $g(a_i) = S_{i-L}$, for $i \in [T + L] \setminus [L]$, where $R_i \in \mathbb{F}_{q^t}^{a \times b}$ and $S_i \in \mathbb{F}_{q^t}^{b \times c}$ are matrices chosen independently and at random and $\deg f = \deg g = L + T - 1$. Set $h(x) := f(x)g(x)$. As $\deg(h) = 2L + 2T - 2$, $ev(h) \in \text{RS}(2L + 2T - 1)$. For all $i \in [L]$, $h(a_i) = f(a_i)g(a_i) = A_iB_i$. Hence,

$$h(a_1) + \cdots + h(a_L) = A_1B_1 + \cdots + A_LB_L = AB.$$

Therefore, AB may be determined by finding the sum $\sum_{i=1}^{L} h(a_i)$. For $i \in [n] \setminus [L]$, upload $f(a_i)$ and $g(a_i)$ to server a_i. Notice that for any $i \in [L]$, if any server had access to $f(a_i)$ or $g(a_i)$ it would have access to some information about A or B, namely, A_i or B_i. By [15, Lemma 2], the information in the N servers is T-secure, meaning no T servers can collude to obtain any information about A or B.

Next, to determine the sum $\sum_{i=1}^{L} h(a_i)$, consider the polynomial $H(x) \in \mathbb{F}_q[x]_{<n-(2L+2T-1)}$. As $RS(n - 2L - 2T + 1) = RS(2L + 2T - 1)^\perp$, we obtain

$$0 = \sum_{i=1}^{n} h(a_i)H(a_i) = \sum_{i=1}^{L} h(a_i)H(a_i) + \sum_{i=L+1}^{n} h(a_i)H(a_i).$$

Thus, by isolating the sum of interest,

$$\sum_{i=1}^{L} h(a_i)H(a_i) = -\sum_{i=L+1}^{n} h(a_i)H(a_i).$$

Recall that $H(a_i) = \alpha \neq 0$ for all $a_i \in F_1$. Therefore,

$$AB = \sum_{i=1}^{L} h(a_i) = \frac{-1}{\alpha} \sum_{i=L+1}^{n} h(a_i)H(a_i).$$

We obtain a T-secure SDMM scheme that finds AB using $|U|$ of the $N = n - L$ servers.

Alternatively, as $\deg(h) = 2L + 2T - 2$, the values of $h(x)$ at any $2L + 2T - 1$ elements of the field determines the polynomial $h(x)$. Thus, the transmission from any $2L+2T-1$ servers to the user finds $h(x)$, and the sum $\sum_{i=1}^{L} h(a_i)$. ☐

It is important to highlight differences and similitudes of Theorem 2.1 with related schemes. The following result justifies the name secure MatDot codes.

Corollary 2.2. *The threshold of the T-secure MatDot codes is $2L + 2T - 1$. In particular, if $T = 0$, we obtain the MatDot codes, whose threshold is $2L - 1$. In this case, the secure MatDot codes will determine AB at least as fast as the MatDot codes. In particular, the secure MatDot codes are faster if the servers indexed by U finish before than any $2L - 1$ servers.*

Proof. By Theorem 2.1, the T-secure MatDot codes recover AB when any $2L + 2T - 1$ servers have finished. Thus, the threshold is $2L + 2T - 1$. If $T = 0$, we see that the T-secure MatDot codes are the same as the MatDot codes by construction.

The MatDot codes recover AB when any $2L - 1$ servers finish by construction. By the proof of Theorem 2.1, we see that the secure MatDot code retrieves AB when either the servers indexed by U, or any $2L - 1$ servers finish. ☐

Remark 2.3. Each server may take a different time to calculate and transmit. Thus, just because the secure MatDot codes contact fewer servers when $|U| < 2L - 1$ does not imply the scheme will calculate the product AB faster. The MatDot codes are faster if the servers indexed by U finish before any $2L-1$ servers. Thus, identifying the $|U|$ fastest servers beforehand would guarantee that the secure MatDot codes are faster than the MatDot codes.

Corollaries 2.6 and 2.7 give instances where the secure MatDot codes are potentially faster than the MatDot codes. We prove that if L divides q or divides $q-1$, then there are secure MatDot codes that determine the sum AB potentially faster than any MatDot code with the same setup.

We can now calculate the upload, download, and total costs of the secure MatDot codes.

Lemma 2.4. *Given the setup in Theorem 2.1, the T-secure MatDot codes have the following costs:*

$$\textit{Upload } = \left(\frac{n}{L}-1\right)(ab+bc),$$
$$\textit{Download } = \begin{cases} \frac{|U|ac}{L} & \textit{in case } \text{(i)} \\ \frac{(2L+2T-1)ac}{L} & \textit{in case } \text{(ii)}, \end{cases}$$

and Total Cost=

$$\left(\frac{n}{L}-1\right)(ab+bc)+\begin{cases} \frac{|U|ac}{L} & \textit{in case } \text{(i)} \\ \frac{(2L+2T-1)ac}{L} & \textit{in case } \text{(ii)}. \end{cases}$$

where (i) *happens when the servers indexed by U finish before any $2L+2T-1$ servers, and* (ii) *otherwise.*

Proof. First, we will determine the upload cost. Recall, that the secure MatDot codes require the source node to transmit $f(a_i)$ and $g(a_i)$ to server a_i for all $i \in \{L+1,\dots,n\}$. Hence $(n-L)(\frac{ab}{L}+\frac{bc}{L})$ elements of $\mathbb{F}_q$ must be transmitted in the upload phase. Therefore,

$$\text{Upload Cost} = (n-L)\left(\frac{ab}{L}+\frac{bc}{L}\right) = \left(\frac{n}{L}-1\right)(ab+bc).$$

Next, we consider the download phase. In the secure MatDot code, $h(a_i)=f(a_i)g(a_i)$ must be transmitted to the user from each $a_i \in U = \{a_i \in \mathbb{F}_q : H(a_i)\neq 0\} \setminus F_1$, or from any $2L+2T-1$ other servers. Thus,

$$\text{Download } = \begin{cases} \frac{|U|ac}{L} & \text{in case (i)} \\ \frac{(2L+2T-1)ac}{L} & \text{in case (ii)}. \end{cases}$$

The Total Cost is a consequence of the previous cost calculations. □

A. Security against the user

Observe that, in general, the MatDot codes and the secure MatDot codes provide no security against the user. This happens because the source node uploads the information to the $N=n-L$ servers. Thus, the user will have enough information to recover the polynomial $h(x)=f(x)g(x)$ and the products A_iB_i, obtaining thus partial information about the matrices A and B. As we prove now, the secure MatDot codes may provide security against the user in some instances.

Theorem 2.5. *Assume that $|U|<2T+2L-1$ in Theorem 2.1. If the source uses only the servers indexed by U, rather than all the $n-L$ servers, the T-secure MatDot codes provide security against the user.*

Proof. By the proof of Theorem 2.1, the user will be able to recover $AB=\sum_{i=1}^{L} h(a_i)$ using only the servers indexed by U. As $|U|<2T+2L-1$, the user will not be able to recover $h(x)=f(x)g(x)$. Thus, the user has access only to AB, rather than the components A_iB_i. □

Recall N is the number of servers. Yu et al. use the N-th roots of the unity of $\mathbb{F}_q$ to define the *discrete Fourier transform* (DFT) codes [12]. The DFT codes give security against the user providing N divides $q-1$ and $N=L+2T$. In Section III, we see instances where the DFT codes can not be applied, but the secure MatDot codes can.

B. Explicit constructions

Notice that constructing a low communication rate secure MatDot code depends entirely on finding H polynomials with small support, i.e., many zeros. The paper follows a few specific constructions of H in various circumstances.

Corollary 2.6. *Suppose $q=p^t$ and $L\leq p$. There are T-secure MatDot codes where the user downloads data from either*

$$p\left(\left\lfloor\frac{2L+2T-1}{p}\right\rfloor+1\right)-L$$

fixed servers, or any $2L+2T-1$ servers. The upload cost is $(\frac{n}{L}-1)(ab+bc)$. The download cost is $\left(p\left(\lfloor\frac{2L+2T-1}{p}\rfloor+1\right)-L\right)acL^{-1}$ if the fixed servers finish before any $2L+2T-1$ servers, or $(2L+2T-1)\,acL^{-1}$ otherwise.

Proof. The field $\mathbb{F}_q=\{a_1,\dots,a_n\}$ has a subfield, say $F_1:=\{a_1,\dots,a_p\}$, isomorphic to $\mathbb{F}_p$. Consider the additive cosets of F_1 in $\mathbb{F}_q$: $F_1,F_2,\dots,F_M$, where $M:=\frac{q}{p}$. These cosets partition $\mathbb{F}_q$. Set $\ell:=\lfloor\frac{2L+2T-1}{p}\rfloor+1$ and

$$H(x):=\prod_{j=\ell+1}^{M}\prod_{a_i\in F_j}(x-a_i).$$

We claim that H satisfies the criteria in Theorem 2.1, meaning $H\in\mathbb{F}_q[x]_{<n-2L-2T+1}$ and $H(a_i)=\alpha$ for some $\alpha\in F_q\setminus\{0\}$ for all $i\in[p]$. Indeed,

$$\deg(H)=(M-\ell)|F_1|=\left(\frac{q}{p}-\left\lfloor\frac{2L+2T-1}{p}\right\rfloor-1\right)p$$
$$<q-2L-2T+1,$$

since all cosets have the same cardinality. By [13, Proposition 3.2], for any $j\in[M]$, there is a polynomial h_j such that $h_j(a)=0$ for all $a\in F_j$ and $h_j(a)=\alpha\in\mathbb{F}_q$ for all $a\in F_i$. Then, H is the product of some of these h_j's. To be more precise, $H=h_{\ell+1}\cdots h_M$. Hence, $H(x)=\prod_{j=\ell+1}^{M}\prod_{a_i\in F_j}(x-$

$a_i)$ is constant on F_1. Observe that $(x-a_i)$ does not divide $H(x)$ for any $i \in [L]$. Thus, $H(a_i) \neq 0$ for any $a_i \in F_1$. Therefore, $H(x)$ is non-zero and constant on F_1.

By Theorem 2.1, the secure MatDot codes determine AB using $n-L$ servers with an upload cost of $(\frac{n}{L}-1)(ab+bc)$. Recall $U = \{a_i \in \mathbb{F}_q : H(a_i) \neq 0\} \setminus F_1$. Note in this instance the zeros of H are apparent, so

$$U = \{a_i \in F_j \mid 1 \leq j \leq \ell\} \setminus \{a_1, \ldots, a_L\}.$$

If the servers indexed by U finish first than any $2L+2T-1$ servers, the download cost is

$$|U|\frac{ac}{L} = (\ell p - L)\,\frac{ac}{L} = \left(p\left(\left\lfloor\frac{2L+2T-1}{p}\right\rfloor+1\right)-L\right)\frac{ac}{L}.$$

Otherwise, the download cost is $\frac{(2L+2T-1)ac}{L}$. □

Previous result applies if $L \leq d$, where d divides either q or $q-1$. We illustrate now the case when d divides $q-1$.

Corollary 2.7. *Assume $L \leq d$, where d divides $q-1$. There are T-secure MatDot codes where the user downloads data from either*

$$d\left(\left\lfloor\frac{2L+2T-1}{d}\right\rfloor+1\right)-L+1$$

fixed servers, or any $2L+2T-1$ servers.

Proof. As d divides $q-1$, there exists a subgroup F_1 of the cyclic group $\mathbb{F}_q \setminus \{0\}$ of size $|F_1| = d$. The proof follows the same lines as the proof of Corollary 2.6. □

Remark 2.8. There are more constructions for the polynomial $H(x)$. See, for instance, [16].

III. Examples

This section gives comparative examples of the MatDot, the secure MatDot, and the DFT codes. The DFT codes provide security against the user provided that the field $\mathbb{F}_q$ contains the N-th roots of the unity. The DFT codes are used in Example 3.4. As we showed in Theorem 2.5, there are instances where the secure MatDot codes also provide security against the user. The secure MatDot codes with security against the user are utilized in Example 3.5. The associated costs of the DFT and the secure MatDot codes used in Examples 3.4 and 3.5 are the same. Example 3.6 presents a scenario where the proposed secure MatDot codes can be used, but the DFT scheme cannot be applied.

First, consider the case where we want to multiply the matrices A and B with entries in $\mathbb{F}_9$ using 9 servers. Assume that we have the inner product partitions $A = [A_1 A_2 A_3]$ and $B = [B_1 B_2 B_3]$ such that $AB = A_1B_1 + A_2B_2 + A_3B_3$.

Example 3.1. (MatDot codes) Define the following polynomials in $\mathbb{F}_9[x]$.

$$f(x) := A_0 + A_1x + A_2x^2 \text{ and } g(x) := B_0 + B_1x + B_2x^2.$$

Assume $\mathbb{F}_9 = \{a_1, \ldots, a_9\}$. For $i \in \{1, \ldots, 9\}$, upload the elements $f(a_i)$ and $g(a_i)$ to the server P_i. The server computes $f(a_i)g(a_i)$. When a server finishes, it sends the data to the user. As the polynomial $h(x) := f(x)g(x)$ has degree 4, the user interpolates the polynomial $h(x)$, and as a consequence, recovers the values A_iB_i, after the first 5 servers have finished.

The number 5 is optimal and cannot be improved. There is no scheme where the user can recover AB after any 4 servers have finished. The following example shows that if we can identify the 3 fastest servers in advance, we can obtain AB after these 3 servers have finished. We remark this is an alternative way to compute AB. Thus, AB can also be calculated if any other 5 servers end before the 3 fastest servers.

Example 3.2. (Secure MatDot codes) Assume that $\mathbb{F}_3 = \{a_1, a_2, a_3\}$, and $\mathbb{F}_9 = \{a_1, \ldots, a_9\}$, where $\{a_4, a_5, a_6\} = 1 + \mathbb{F}_3$, and $\{a_7, a_8, a_9\} = 2 + \mathbb{F}_3$. In other words, the sets $\{a_1, a_2, a_3\}, \{a_4, a_5, a_6\}$, and $\{a_7, a_8, a_9\}$ are the cosets of $\mathbb{F}_3 \subset \mathbb{F}_9$. Consider that servers P_7, P_8 and P_9 are the fastest. Define the polynomials $f(x)$ and $g(x)$ such that

$$f(a_1) = A_1, \quad f(a_2) = A_2, \quad f(a_3) = A_3, \quad \text{and}$$
$$g(a_1) = B_1, \quad g(a_2) = B_2, \quad g(a_3) = B_3.$$

For $i \in \{1, \ldots, 9\}$, upload to server P_i the elements $f(a_i)$ and $g(a_i)$. The server computes $f(a_i)g(a_i)$. When a server finishes, it sends the data to the user. Note that the polynomials $f(x)$ and $g(x)$ have degree 2 each. Thus $h(x) := f(x)g(x)$ has degree 4. This means that the vector $(h(a_1), \ldots, h(a_9))$ is an element of the $[9,5]$ RS code over $\mathbb{F}_9$. Define the polynomial

$$H(x) := (x-a_4)(x-a_5)(x-a_6).$$

Note that the vector $(H(a_1), \ldots, H(a_9))$ is an element of the $[9,4]$ RS code over $\mathbb{F}_9$, which is the dual of the $[9,5]$ RS code over $\mathbb{F}_9$. Thus, we have (i) below is valid.

(i) $\sum_{i=1}^{9} h(a_i)H(a_i) = 0$.
(ii) $H(a_4) = H(a_5) = H(a_6) = 0$.
(iii) $\alpha_1 := H(a_1) = H(a_2) = H(a_3) \in \mathbb{F}_9$.
(iv) $\alpha_2 := H(a_7) = H(a_8) = H(a_9) \in \mathbb{F}_9$.

Note that (ii) comes from the definition of $H(x)$. (iii) and (iv) come from the definition of $H(x)$ and the fact that $\{a_1, a_2, a_3\}, \{a_4, a_5, a_6\}$, and $\{a_7, a_8, a_9\}$ are the cosets of $\mathbb{F}_3 \subset \mathbb{F}_9$.

Combining previous properties of $H(x)$, we have

$$A = \sum_{i=1}^{3} A_iB_i = \sum_{i=1}^{3} f(a_i)g(a_i) = \sum_{i=1}^{3} h(a_i)$$
$$= -\frac{\alpha_2}{\alpha_1}\sum_{i=7}^{9} h(a_i)H(a_i).$$

Consequently, the user recovers A when either the three servers P_7, P_8, and P_9, or any 5 other servers have finished. The number 5 comes from Example 3.1.

Observe that the upload costs for the MatDot and the secure MatDot codes utilized in Examples 3.1 and 3.2 are the same. If the three fastest servers finished first as expected, the download cost for the secure MatDot code is less than the MatDot code download cost. In the worst-case scenario where there is a delay for one of the fastest servers, the download costs for the MatDot and the secure MatDot codes are the same.

Remark 3.3. Note that for an arbitrary L, the polynomials f and g in the MatDot scheme will have degree $L-1$. Hence, $h(x) = f(x)g(x)$ will have degree $2L-2$ and therefore the user will need to contact $2L-1$ servers to interpolate. In particular, the information from any $2L-1$ servers will be enough to determine AB. Hence, if each server transmits $h(a_i)$ as soon as computing, the MatDot scheme will be as fast as the fastest $2L-1$ servers to compute and transmit.

A. Security against the user

Consider the case where we want to multiply the matrices A and B with entries in $\mathbb{F}_{43}$, security $T=2$, with the help of 7 servers. Assume that we have the inner product partitions $A = [A_1 A_2 A_3]$ and $B = [B_1 B_2 B_3]$ such that $AB = A_1B_1 + A_2B_2 + A_3B_3$. Let R_1 and R_2 be random matrices with entries in $\mathbb{F}_{43}$ of size as A_i. Let S_1 and S_2 be random matrices with entries in $\mathbb{F}_{43}$ of size as B_i.

Example 3.4. (DFT codes) Let $\alpha_7, \alpha_7^2, \ldots, \alpha_7^7$ be the 7-th roots of the unity in $\mathbb{F}_{43}$. Define the polynomials

$$f(x) := A_1 + A_2x + A_3x^2 + R_1x^3 + R_2x^4, \quad \text{and}$$
$$g(x) := B_1 + B_2x^{-1} + B_3x^{-2} + S_1x^{-5} + S_2x^{-6}.$$

For $i \in \{1, \ldots, 7\}$, the source node uploads to the server P_i the elements $f(\alpha^i)$ and $g(\alpha^i)$. The server computes $f(\alpha^i)g(\alpha^i)$. When a server finishes, it sends the data to the user. By [12, Section III], the user recovers the matrix AB after receiving the 7 symbols $f(\alpha^i)g(\alpha^i)$. Note that the user cannot recover the polynomial $h(x) = f(x)g(x)$, as the information of the 7 symbols is not enough.

Example 3.5. (Secure MatDot codes with security against the user) Let $\mathbb{F}_{43}^* = \{a_1, \ldots, a_{42}\}$ be the multiplicative group of $\mathbb{F}_{43}$ and $F_1 := \{a_1, \ldots, a_3\}$ the 3-rd roots of the unity in $\mathbb{F}_{43}$. Assume $a_{43} := 0 \in \mathbb{F}_{43}$ and $F_1, \ldots, F_{14}$ are the cosets of $F_1 \subset \mathbb{F}_{43}^*$. Define the polynomials $f(x)$ and $g(x)$ such that

$$f(a_1) = A_1, \quad f(a_2) = A_2, \quad f(a_3) = A_3, \quad f(a_4) = R_1,$$
$$f(a_5) = R_2, \quad g(a_1) = B_1, \quad g(a_2) = B_2, \quad g(a_3) = B_3$$
$$g(a_4) = S_1, \quad \text{and} \quad g(a_5) = S_2.$$

The source node uploads to server P_i the elements $f(a_{i+3})$ and $g(a_{i+3})$, for $i \in \{1, \ldots, 6\}$, and the elements $f(a_{43})$ and $g(a_{43})$ to server P_7. Every server computes $f(a_i)g(a_i)$. When a server finishes, it sends the data to the user. Note that the polynomials $f(x)$ and $g(x)$ have degree 4 each. Thus $h(x) := f(x)g(x)$ has degree 8. This means that the vector $(h(a_1), \ldots, h(a_{43}))$ is an element of the $[43, 9]$ RS code over $\mathbb{F}_{43}$. Define the polynomial

$$H(x) := \prod_{i=4}^{14} \prod_{a \in F_i} (x - a).$$

As $\deg(H(x)) = 33$, the vector $(H(a_1), \ldots, H(a_{43}))$ is an element of the $[43, 34]$ RS code over $\mathbb{F}_{43}$, which is the dual of the $[43, 9]$ RS code over $\mathbb{F}_{43}$. Thus, we have (i) below is valid.

(i) $\sum_{i=1}^{43} h(a_i)H(a_i) = 0$.
(ii) $H(a_{10}) = \cdots = H(a_{42}) = 0$.
(iii) $\alpha := H(a_1) = H(a_2) = H(a_3) \in \mathbb{F}_{43}$.

Observe that (ii) comes from the definition of $H(x)$. (iii) is valid because of the definition of $H(x)$ and the fact that the F_i's are the cosets of $F_1 \subset \mathbb{F}_{43}^*$. Combining previous properties of $H(x)$, we have

$$A = \sum_{i=1}^{3} A_iB_i = \sum_{i=1}^{3} f(a_i)g(a_i) = \sum_{i=1}^{3} h(a_i)$$
$$= -\frac{1}{\alpha}\left(\sum_{i=4}^{9} h(a_i)H(a_i) + h(a_{43}H(a_{43}))\right).$$

Consequently, the user recovers A after receiving the 7 symbols from the servers.

The DFT codes utilized in Example 3.4 and the secure MatDot codes constructed in Example 3.5 provide security against the user. In both cases, the user cannot recover the matrices A_iB_i as the information downloaded from the 7 servers is not enough to interpolate the polynomial $h(x) = f(x)g(x)$. In addition, the associated costs of the two codes are the same.

Example 3.6 presents a scenario where the proposed secure MatDot codes can be used, but the DFT scheme cannot be applied.

Example 3.6. Consider the case where we want to multiply the matrices A and B with entries in $\mathbb{F}_{19}$, security $T=2$, with the help of 7 servers. Assume that we have the inner product partitions $A = [A_1A_2A_3]$ and $B = [B_1B_2B_3]$ such that $AB = A_1B_1 + A_2B_2 + A_3B_3$. As 7 does not divide 18, the field $\mathbb{F}_{19}$ has no 7-th roots of the unity. So we cannot use DFT codes. As the field $\mathbb{F}_{19}$ has the 3-rd roots of the unity. We can follow the same procedure as Example 3.5 to use the secure MatDot codes. The associated costs are exactly the costs of Example 3.5.

IV. Conclusion

This paper introduces secure MatDot codes by utilizing locally recoverable codes into the MatDot construction. They allow users to access the product AB of two matrices over a finite field while obtaining no information about A or B. Under some conditions, they return the product using fewer than the number of servers required by the original MatDot scheme. In addition, secure MatDot applies in some settings where the DFT codes do not.

References

[1] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, “Polynomial codes: An optimal design for high-dimensional coded matrix multiplication,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS’17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 4406–4416.

[2] W.-T. Chang and R. Tandon, “On the capacity of secure distributed matrix multiplication,” in *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1–6.

[3] G. Joshi, Y. Liu, and E. Soljanin, “On the delay-storage trade-off in content download from coded distributed storage systems,” *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 5, pp. 989–997, 2014.

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

[5] A. Severinson, A. Graell i Amat, and E. Rosnes, “Block-diagonal and lt codes for distributed computing with straggling servers,” *IEEE Transactions on Communications*, vol. 67, no. 3, pp. 1739–1753, 2019.

[6] H. Yang and J. Lee, “Secure distributed computing with straggling servers using polynomial codes,” *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 1, pp. 141–150, 2019.

[7] U. Sheth, S. Dutta, M. Chaudhari, H. Jeong, Y. Yang, J. Kohonen, T. Roos, and P. Grover, “An application of storage-optimal matdot codes for coded matrix multiplication: Fast k-nearest neighbors estimation,” in *2018 IEEE International Conference on Big Data (Big Data)*, 2018, pp. 1113–1120.

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

[9] M. Kim, H. Yang, and J. Lee, “Private coded matrix multiplication,” *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1434–1443, 2020.

[10] S. Dutta, M. Fahim, F. Haddadpour, H. Jeong, V. Cadambe, and P. Grover, “On the optimal recovery threshold of coded matrix multiplication,” *IEEE Transactions on Information Theory*, vol. 66, no. 1, pp. 278–301, 2020.

[11] K.-H. Huang and J. A. Abraham, “Algorithm-based fault tolerance for matrix operations,” *IEEE Transactions on Computers*, vol. C-33, no. 6, pp. 518–528, 1984.

[12] N. Mital, C. Ling, and D. Gunduz, “Secure distributed matrix computation with discrete fourier transform,” 2021.

[13] I. Tamo and A. Barg, “A family of optimal locally recoverable codes,” *IEEE Transactions on Information Theory*, vol. 60, no. 8, pp. 4661–4676, 2014.

[14] A. Barg, I. Tamo, and S. Vlăduţ, “Locally recoverable codes on algebraic curves,” *IEEE Transactions on Information Theory*, vol. 63, no. 8, pp. 4928–4939, 2017.

[15] R. A. Machado, R. G. L. D’Oliveira, S. E. Rouayheb, and D. Heinlein, “Field trace polynomial codes for secure distributed matrix multiplication,” in *2021 XVII International Symposium ”Problems of Redundancy in Information and Control Systems” (REDUNDANCY)*, 2021, pp. 188–193.

[16] G. Micheli, “Constructions of locally recoverable codes which are optimal,” *IEEE Transactions on Information Theory*, vol. 66, no. 1, pp. 167–175, 2020.