Secure MatDot codes: a secure, distributed matrix multiplication scheme

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*—This paper presents secure MatDot codes, a family of evaluation codes that support secure distributed matrix multiplication via a careful selection of evaluation points that exploit the properties of the dual code. We show that the secure MatDot codes provide security against the user by using locally recoverable codes. These new codes complement the recently studied discrete Fourier transform codes for distributed matrix multiplication schemes that also provide security against the user. There are scenarios where the associated costs are the same for both families and instances where the secure MatDot codes offer a lower cost. In addition, the secure MatDot code provides an alternative way to handle the matrix multiplication by identifying the fastest servers in advance. In this way, it can determine a product using fewer servers, specified in advance, than the MatDot codes which achieve the optimal recovery threshold for distributed matrix multiplication schemes.**

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 the collusion of T servers reveals no information about A or B to the servers. In this paper, we are also interested in having security against the user, which means that the final user gets no partial information about A or B. We introduce locally recoverable codes (also known as codes with locality) into the polynomial code approach recently introduced in [1] to infuse security into the scheme and at times provide speed ups or lower costs. The core idea is using a specific set of evaluation points to allow the dual code to play a crucial role in the multiplication process.

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 on 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.

The work of the first two authors is supported in part by NSF DMS-2201075. The 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 DMS-1855136 and in part by the Commonwealth Cyber Initiative. (Corresponding author: Hiram H. López.)

As detailed below, the approach considered here is distinct from that given in [2], [3], [4] in its use of locally recoverable codes to provide security against the user (meaning the user cannot retrieve information about the A_i's and B_i's or the factors of AB) and allow for various sets of nodes (possibly non-systematic) to aid in the recovery of AB without revealing the polynomial h. Each coordinate j of a locally recoverable code C has a (small) set R_j of other coordinates such that for each codeword $c \in C$, the j^{th} coordinate of c depends only on the entries in the positions R_j. The set R_j is called a recovery set for j. Hence, combinations of recovery sets may be used to uncover AB rather than relying only on systematic nodes as done in [2], [3], [4]. This feature is a primary motivation to use locally recoverable codes and is one novel aspect of the new scheme. In terms of security, several schemes using polynomial codes have been proposed in the literature. For instance, in [1], [5], 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. There is extensive work on coded matrix multiplication; see [6], [7], [8], [9], [10], [11], [12].

This paper defines the secure MatDot codes, a T-SDMM scheme, which provides security against the user. An SDMM scheme provides *security against the user* when the user cannot get any information about the matrices A_i's, B_i's, A, or B from any collection of T servers. The fact that the secure MatDot scheme ensures no T servers can collude to reveal information about A or B relies on [18]. As we prove in Theorem 2.3, the secure MatDot codes provide security against the user by relying on specified servers. We compare the secure MatDot codes with the discrete Fourier transform (DFT) code [13] and improve MatDot by taking non-systemic nodes besides the classical systematic nodes. To the best of our knowledge, only DFT, which is a T-SDMM scheme that

provides security against the user. We will highlight examples that show that when the new scheme and the DFT can be used, our scheme provides a lower cost. In Corollary 2.7, we give conditions where the secure MatDot and the DFT codes have similar associated costs. Example 3.3 presents a setting in which the proposed secure MatDot codes can be used, but the DFT scheme could not be applied because of the restriction on the number of servers, which must divide $q-1$. Relaxing the condition on the number of servers is possible, but it results in the cost of DFT is greater than the secure MatDot codes. Remark 3.4 shows a scenario where DFT could not be applied because we would need as many servers as the field size, but the MatDot codes avoid that restriction.

We also consider in this paper 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. In this context, the MatDot codes developed in [14] outperform Polynomial codes [1] and the Algorithm-Based Fault Tolerance algorithm [15]. 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.

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. Let $\mathbb{F}_q$ be the finite field with q elements and $\mathbb{F}_q^* := \mathbb{F}_q \setminus \{0\}$. 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 *Upload Cost* is the total number of symbols the scheme requires to upload to all servers. The *Download Cost* is the total number of symbols the scheme must transmit from all the servers to the user to calculate AB. The *Total Cost* is the upload cost plus the download cost.

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

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

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 $\mathrm{RS}_q(k)$.

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 $\mathrm{RS}_q(k)^\perp = \mathrm{RS}_q(n-k)$. We 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 [14], and the locally recoverable codes considered by Tamo and Barg [16]; see also [17]. From now on, 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 $A_i \in \mathbb{F}_q^{a\times \frac{b}{L}}$, $B_i \in \mathbb{F}_q^{\frac{b}{L}\times c}$, and $AB = A_1B_1 + \cdots + A_LB_L$. 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 $H|_{F_1} = \alpha \in \mathbb{F}_q^*$. 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 $R_i \in \mathbb{F}_q^{a\times b}$ and $S_i \in \mathbb{F}_q^{b\times c}$ are matrices chosen independently and at random and $\deg f = \deg g = L + T - 1$. Set $h(x) := f(x)g(x)$. As $\deg(h) = 2L + 2T - 2$, $ev(h) \in \mathrm{RS}_q(2L + 2T - 1)$. 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 [18, 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 $\sum_{i=1}^{L} h(a_i)$. □

We can now calculate the secure MatDot codes costs.

Lemma 2.2. *Given the setup in Theorem 2.1, the T-secure MatDot codes have the following costs:*

$$\textit{Upload } = (n-L)\left(\frac{ab}{L}+\frac{bc}{L}\right) \textit{ and}$$

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

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

Proof. 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. For 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, the result follows. □

Observe that, in general, the MatDot codes and the secure MatDot codes provide no security against the user 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 provide security against the user when we do not use all the servers.

Theorem 2.3. *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 secure MatDot codes provide security against the user with the following costs:*

$$\textit{Upload } = |U|\left(\frac{ab}{L}+\frac{bc}{L}\right) \quad \textit{and} \quad \textit{Download } = |U|ac.$$

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.

We can construct the polynomials f and g from the proof of Theorem 2.1 via Lagrange interpololation [19]. As $U \subseteq \mathbb{F}_q\setminus F_1$, for every element u in U, we have that $h(u)$ is a linear combination with nonzero constants of the elements A_iB_j, A_iS_j, R_iB_j, and R_iS_j. Thus, the element $h(u)$ is not leaking information about A_i or B_i. □

Remark 2.4. Recall N represents the number of servers. Mital et al. use the N-th roots of the unity of $\mathbb{F}_q$ to define the DFT codes [13] with the following cost:

$$\text{Upload } = N\left(\frac{ab}{L}+\frac{bc}{L}\right).$$

The DFT codes give security against the user providing $N \mid q-1$ and $N=L+2T$.

If the number of servers is fixed, we see in Section III instances where the DFT codes may not be utilized because of the restriction that $N \mid q-1$. But if we allow to easy the restriction on the number of servers, the secure MatDot costs are lower than the DFT costs.

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 2.5. *The threshold of the MatDot codes is $2L+2T-1$. If $T=0$, we obtain the MatDot codes, whose threshold is $2L-1$. The secure MatDot codes determine AB as fast as the MatDot codes, and faster when $|U|<2L-1$ and servers indexed by U finish before 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 2.6. The secure 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.

A. Explicit constructions

Notice that constructing a 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. It is important to remark that there are more constructions. See, for instance, [16], [20], [21].

Corollary 2.7. *Assume $N=2T+L$. If $L \mid N-1$, $L \mid q-1$, and $N \mid q-1$, then the secure MatDot codes and DFT codes have the same costs.*

Proof. Consider $N-1=(r-1)L$, for some integer r. Let F_1 be the set of L-th roots of the unitity in $\mathbb{F}_q$ and $F_1,\dots,F_r,\dots,F_{\frac{q-1}{L}}$ the multiplicative cosets of F_1. Define $U := F_2 \cup \ldots \cup F_r \cup \{0\}$ and the annulator polynomial

$$H(x) := \prod_{j=r+1}^{\frac{q-1}{L}} \prod_{a_i \in F_j} (x-a_i).$$

As $H|_{F_1}=\alpha \in \mathbb{F}_q^*$, $\deg(H)=\left(\frac{q-1}{L}-r\right)L = q-1-rL = n-2L-2T$, and $U=\{a_i \in \mathbb{F}_q : H(a_i)\neq 0\}\setminus F_1$, then we can apply Theorem 2.1. Note that $|U|<2T+2L-1$. So we obtain security against the user by Theorem 2.3. Finally, as $|U|=N$, the costs from the secure MatDot code and the DFT codes are the same by Remark 2.4 and Theorem 2.3. □

Corollary 2.8. *Suppose $q = p^t$ and $L \leq p$. There are 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.

Proof. The field $\mathbb{F}_q = \{a_1, \ldots, a_n\}$ has a subfield, say $F_1 := \{a_1, \ldots, a_p\}$, isomorphic to $\mathbb{F}_p$. Consider the additive cosets of F_1 in $\mathbb{F}_q$: $F_1, F_2, \ldots, F_M$, 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 $\alpha \in F_q \backslash \{0\}$ for all $i \in [p]$. Indeed,

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

since all cosets have the same cardinality. By [16, 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.

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

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

Thus, the user downloads data from

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

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

Remark 2.9. Previous result applies if $L \leq d$, where $d \mid q$ or $d \mid q - 1$. We illustrate now the case when $d \mid q - 1$.

Corollary 2.10. *Assume $L \leq d$, where $d \mid q - 1$. There are secure MatDot codes where the user downloads data from*

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

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

Proof. As $d \mid q - 1$, 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 2.8. □

III. Examples

This section gives comparative examples of the secure MatDot, the DFT, and the MatDot codes.

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 $A = [A_1 A_2 A_3]$ 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 3.1. (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

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

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 [13, 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 3.2. (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 $F_1 := \{a_1, a_2, a_3\}$ 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 multiplicative cosets of F_1. Define the polynomials $f(x)$ and $g(x)$ such that

$$\begin{aligned}&f(a_i) = A_i, f(a_4) = R_1, f(a_5) = R_2, \\ &g(a_i) = B_i, g(a_4) = S_1, \text{ and } g(a_5) = S_2, \text{ for } i \in \{1, 2, 3\}.\end{aligned}$$

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 $f(x)$ and $g(x)$ have degree 4. Thus $h(x) := f(x)g(x)$ has degree 8 and $(h(a_1), \ldots, h(a_{43})) \in \mathrm{RS}_{43}(9)$. Define

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

As $\deg(H(x)) = 33$, the vector $(H(a_1), \ldots, H(a_{43}))$ is an element of $\mathrm{RS}_{43}(34)$, which is the dual of $\mathrm{RS}_{43}(9)$. Thus, we have $\sum_{i=1}^{43} h(a_i)H(a_i) = 0$. Observe that $H(a_{10}) = \cdots = H(a_{42}) = 0$. Moreover, $\alpha := H(a_1) = H(a_2) = H(a_3) \in \mathbb{F}_{43}$ since the F_i's are the cosets of $F_1 \subseteq \mathbb{F}_{43}^*$. Thus,

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

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

The DFT codes utilized in Example 3.1 and the secure MatDot codes constructed in Example 3.2 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 3.3 presents a scenario where the secure MatDot codes can be used, but the DFT scheme cannot be applied. If we decide to relax the condition on the number of servers, then the cost of DFT is greater than the secure MatDot codes.

Example 3.3. Suppose 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 $A = [A_1A_2A_3]$ and $B = [B_1B_2B_3]$ such that $AB = A_1B_1 + A_2B_2 + A_3B_3$. As $7 \nmid 18$, $\mathbb{F}_{19}$ has no 7-th roots of the unity and we cannot use DFT codes. As $\mathbb{F}_{19}$ has the 3-rd roots of the unity, we can follow the same procedure as Example 3.2 to use the secure MatDot codes. The upload cost is $\frac{7}{3}(ab + bc)$. If we relax the condition on the number of servers, then we can use the DFT codes with $N = 6$. As $T = 2$ and $N = L + 2T$, then $L = 2$. By Remark 2.4, the upload cost for the DFT code is $\frac{6}{2}(ab + bc)$.

Remark 3.4. Note that if $q = 2^t$ and $q - 1$ is a prime, for instance, $q = 32$, then the DFT could not be applied because we would need as many servers as the size of the field. By Remark 2.9, we can use secure MatDot codes because there are groups of order 2^ℓ in $\mathbb{F}_q$, for all $\ell \leq t$.

Suppose now we want to multiply the matrices A and B with entries in $\mathbb{F}_9$ using 9 servers. Assume that $A = [A_1A_2A_3]$ and $B = [B_1B_2B_3]$ such that $AB = A_1B_1 + A_2B_2 + A_3B_3$.

Example 3.5. (MatDot codes) Consider

$$f(x) := A_0 + A_1x + A_2x^2, g(x) := B_0 + B_1x + B_2x^2 \in \mathbb{F}_9[x].$$

Assume $\mathbb{F}_9 = \{a_1, \ldots, a_9\}$. For $i \in \{1, \ldots, 9\}$, upload $f(a_i)$ 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$h(x) := f(x)g(x)$ has degree 4, the user interpolates $h(x)$, and then 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.

The next example highlights the use of non-systematic nodes.

Example 3.6. (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 $\mathbb{F}_3 \subseteq \mathbb{F}_9$. 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, $\sum_{i=1}^{9} h(a_i)H(a_i) = 0$. Note that $H(a_4) = H(a_5) = H(a_6) = 0$. Moreover, $\alpha_1 := H(a_1) = H(a_2) = H(a_3), \alpha_2 := H(a_7) = H(a_8) = H(a_9) \in \mathbb{F}_9$ since $\{a_1, a_2, a_3\}, \{a_4, a_5, a_6\}$, and $\{a_7, a_8, a_9\}$ are the cosets of $\mathbb{F}_3 \subseteq \mathbb{F}_9$. Thus,

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

Consequently, the user recovers A when either the three servers P_7, P_8, and P_9, or any 5 other servers have finished.

Observe that the upload costs for the MatDot and the secure MatDot codes utilized in Examples 3.5 and 3.6 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 3.7. 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.

IV. Conclusion

This paper introduces secure MatDot codes by utilizing locally recoverable codes in the MatDot construction. The secure MatDot codes provide security against the user when the source uses specific servers. The secure MatDot codes return the product faster than the original MatDot scheme when particular servers finish first. Advantages of the new scheme over DFT are considered, though a full investigation of the complexities remains.

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] Q. Yu, S. Li, N. Raviv, S. M. M. Kalan, M. Soltanolkotabi, and S. A. Avestimehr, "Lagrange coded computing: Optimal design for resiliency, security, and privacy," in *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and M. Sugiyama, Eds., vol. 89. PMLR, 16–18 Apr 2019, pp. 1215–1225. [Online]. Available: https://proceedings.mlr.press/v89/yu19b.html

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

[4] ——, "Private and secure distributed matrix multiplication with flexible communication load," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2722–2734, 2020.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

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

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

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] Q. Yu, N. Raviv, J. So, and A. S. Avestimehr, "Lagrange coded computing: Optimal design for resiliency, security and privacy," in *AISTATS*, 2019.

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

[21] J. Liu, S. Mesnager, and L. Chen, "New constructions of optimal locally recoverable codes via good polynomials," *IEEE Transactions on Information Theory*, vol. 64, no. 2, pp. 889–899, 2018.