GASP Codes for Secure Distributed Matrix Multiplication

Rafael G. L. D'Oliveira, Salim El Rouayheb, *Member, IEEE*, and David Karpuk

***Abstract*— We consider the problem of secure distributed matrix multiplication (SDMM) in which a user wishes to compute the product of two matrices with the assistance of honest but curious servers. We construct polynomial codes for SDMM by studying a combinatorial problem on a special type of addition table, which we call the degree table. The codes are based on arithmetic progressions, and are thus named GASP (Gap Additive Secure Polynomial) Codes. GASP Codes are shown to outperform all previously known polynomial codes for secure distributed matrix multiplication in terms of download rate.**

***Index Terms*— Data privacy, distributed computing, security.**

I. Introduction

We consider the problem of secure distributed matrix multiplication (SDMM), in which a user has two matrices A and B and wishes to compute their product AB with the assistance of N servers, without leaking any information about A or B to any server. We assume that all servers are honest and responsive, but that they are curious, in that any T of them may collude to try to deduce information about either A or B.

When considering the problem of SDMM from an information-theoretic perspective, the primary performance metric used in the literature is that of the *download rate*, or simply *rate*, which we denote by $\mathcal{R}$. In our scenario, the user queries the servers to perform various matrix mulitplications, and the servers respond with answers that the user can use to piece together the final desired result AB. In this admittedly heuristic description, the rate $\mathcal{R}$ is the ratio of the size of the desired result AB (in bits) to the total amount of information (in bits) the user downloads to obtain the answers from the servers. The goal is to construct a SDMM scheme with rate $\mathcal{R}$ as large as possible.

Manuscript received March 14, 2019; revised January 29, 2020; accepted February 6, 2020. Date of publication February 20, 2020; date of current version June 18, 2020. The work of Rafael G. L. D'Oliveira and Salim El Rouayheb was supported in part by the NSF under Grant CNS-1801630. *(Corresponding author: Rafael G. L. D'Oliveira.)*

Rafael G. L. D'Oliveira was with the Department of Electrical and Computer Engineering, Rutgers University, Piscataway, NJ 08854 USA. He is now with the Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: rafaeld@mit.edu).

Salim El Rouayheb is with the Department of Electrical and Computer Engineering, Rutgers University, Piscataway, NJ 08854 USA (e-mail: salim.elrouayheb@rutgers.edu).

David Karpuk was with the Departamento de Matemáticas, Universidad de los Andes, Bogotá 111711, Colombia. He is now with F-Secure Corporation, 00180 Helsinki, Finland (e-mail: davekarpuk@gmail.com).

Communicated by B. Dey, Associate Editor for Coding Techniques.

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIT.2020.2975021

The problem of constructing polynomial codes for SDMM can be summarized as follows. We partition the matrices A and B as follows:

$$A = \begin{bmatrix} A_1 \\ \vdots \\ A_K \end{bmatrix}, \quad B = \begin{bmatrix} B_1 & \cdots & B_L \end{bmatrix}, \tag{1}$$

$$\text{so that} \quad AB = \begin{bmatrix} A_1B_1 & \cdots & A_1B_L \\ \vdots & \ddots & \vdots \\ A_KB_1 & \cdots & A_KB_L \end{bmatrix}, \tag{2}$$

making sure that all products A_kB_ℓ are well-defined and of the same size. Clearly, computing the product AB is equivalent to computing all subproducts A_kB_ℓ. One then constructs a polynomial $h(x)$ whose coefficients encode the submatrices A_kB_ℓ, and has N servers compute evaluations $h(a_1), \ldots, h(a_N)$. The polynomial h is constructed so that every T-subset of evaluations reveals no information about A or B, but so that the user can reconstruct all of AB given all N evaluations. This follows the general mantra of evaluation codes and, in particular, polynomial codes as originally introduced in [8] and [9].

One can view the parameters K and L as controlling the complexity of the matrix multiplication operations the servers must perform. Imagine a scenario in which one may hire as many servers N as one wants to assist in the SDMM computation, but the computational capacity of each server is limited. In this scenario, one may have fixed values of K and L, and then maximizing the rate $\mathcal{R}$ becomes a question of minimizing N. This is the general perspective we adopt in the SDMM problem.

A. Related Work

Let A and B be partitioned as in (1), and consider the problem of SDMM with N servers and T-security. In [1], a distributed matrix multiplication scheme is presented for the case $K = L$ which achieves a download rate of

$$\mathcal{R}_1 = \frac{K^2}{(K+T)^2}, \tag{3}$$

or equivalently,

$$\mathcal{R}_1 = \frac{\left(\left\lceil \sqrt{N} - T \right\rceil\right)^2}{\left(\left\lceil \sqrt{N} - T \right\rceil + T\right)^2}. \tag{4}$$

In [2], this is improved to

$$\mathcal{R}_2 = \frac{KL}{(K+T)(L+1)-1}, \tag{5}$$

where the polynomial code uses $N = (K+T)(L+1)-1$ servers. Given some fixed N and T, the authors of [2] then find a near-optimal solution to the problem of finding K and L such that $(K+T)(L+1)-1 \leq N$ and that the rate $\mathcal{R}_2$ as above is maximized. In [5], the authors study the case of $T=1$ and obtain a download rate of $\mathcal{R} = KL/(KL+K+L)$, which is the rate of [2] in this case. As far as the present authors are aware, [1], [2], [5] are the only works currently in the literature which study SDMM from the information-theoretic perspective.

We distinguish the SDMM problem from the case where only one of the matrices must be kept secure. In this case, one can use methods like Shamir's secret sharing [3] or Staircase codes [4], if one is also interested in straggler mitigation.

Polynomial codes were originally introduced in [8] in a slightly different setting, namely to mitigate stragglers in distributed matrix multiplication. This work was followed up by [9] which studied fundamental limits of this problem, introduced a generalization of polynomial codes known as *entangled* polynomial codes, and applied similar ideas to other problems in distributed computing. In [10], the authors develop MatDot and PolyDot codes for distributed matrix multiplication with stragglers, and show that while the communication cost is higher than that of the polynomial codes of [8], the recovery threshold, defined to be the minimum number of workers which need to respond to guarantee successful decoding, is much smaller than that of [8]. The MatDot codes of [10] were then applied to the problem of nearest neighbor estimation in [11]. More fundamental questions about the trade-off in computation cost and communication cost in distributed computing were previously addressed in [12]. However, the polynomial codes in these aforementioned works are not designed to ensure security, making them not applicable to settings where there are privacy concerns related to the data being used. This type of setting could range from training neural networks on personal devices to computations on medical data, where legislation requires that certain privacy conditions are met.

Another line of work is Lagrange Coded Computing, a polynomial coding strategy introduced in [6] to mitigate stragglers and adversaries in distributed polynomial coded computation. The results in [6] focus on minimizing the number of required servers for the computation subject to privacy, robustness, and polynomial degree constraints. However, applying the ideas of [6] to the current scenario yields only one-sided privacy, wherein either A or B is kept private, but not both. More related to the current work is that of Private Polynomial Computation [7], which does provide two-sided privacy, but focuses on generic strategies which work for all polynomials of a given degree, rather than polynomial coding strategies tailored for the problem of matrix multiplication. Lastly, it seems that concerns related to data partitioning and block length make the results of the present paper (and generally results on using polynomnial codes for SDMM) incomparable with those of [6] or [7].

TABLE I
THE DOWNLOAD RATE OF GASP CODES

Download Rate	Regions
$\frac{KL}{KL+K+L}$	$1=T<L\leq K$
$\frac{KL}{KL+K+L+T^2+T-3}$	$2\leq T<L\leq K$
$\frac{KL}{(K+T)(L+1)-1}$	$L\leq T<K$
$\frac{KL}{2KL+2T-1}$	$L\leq K\leq T$

B. Main Contribution

The main contributions of this work are as follows.

- In Section II we introduce our polynomial code GASP via an explicit example, in order to demonstrate all of the subtleties of the scheme construction. In Section III we formalize the notion of a polynomial code and introduce basic definitions.
- In Section IV we introduce the key notion of this paper, the *degree table* of a polynomial code. We prove, in Theorem 1, that to every degree table corresponds a secure distributed matrix multiplication scheme.
- In Section V, we present a secure distributed matrix multiplication scheme, $\mathsf{GASP}_{\text{big}}$. We show that $\mathsf{GASP}_{\text{big}}$ outperforms, for almost all parameters, all previously known schemes in the literature, in terms of the download rate.
- In Section VI, we present a secure distributed matrix multiplication scheme, $\mathsf{GASP}_{\text{small}}$. We show that $\mathsf{GASP}_{\text{small}}$ outperforms $\mathsf{GASP}_{\text{big}}$ when $T<\min\{K,L\}$.
- In Section VII, we present a secure distributed matrix multiplication scheme, GASP, by combining both $\mathsf{GASP}_{\text{small}}$ and $\mathsf{GASP}_{\text{big}}$. GASP outperforms all previously known schemes, for almost all parameters, in terms of the download rate.

  The rate of GASP, for $L\leq K$, is given in Table I. For $K<L$, the rate is given by interchanging K and L.

We plot in Fig. 1 the rates obtained by GASP against those obtained by the polynomial code SDMM strategies of [1], [2]. Before launching into the construction of GASP, let us offer some intuitive explanation as to the large improvement in rate offered by GASP over [1], [2]. The polynomial codes of [1], [2], as well as those of the current work, all have the user decode the necessary blocks of AB by interpolating a polynomial $h(x)$, and obtaining the A_kB_ℓ as coefficients of this polynomial. The rate of all three strategies is completely determined by how many evaluations $h(x)$ requires to be interpolated completely, as this is the number of servers employed by the user. The strategies of [1], [2] force every coefficient of $h(x)$ to be potentially non-zero, and therefore interpolating $h(x)$ requires $\deg(h(x))+1$ evaluations. In contrast, GASP codes purposefully rig up $h(x)$ so that it has as many zero coefficients as possible, and that the user knows where these zero coefficients are located. This allows the user

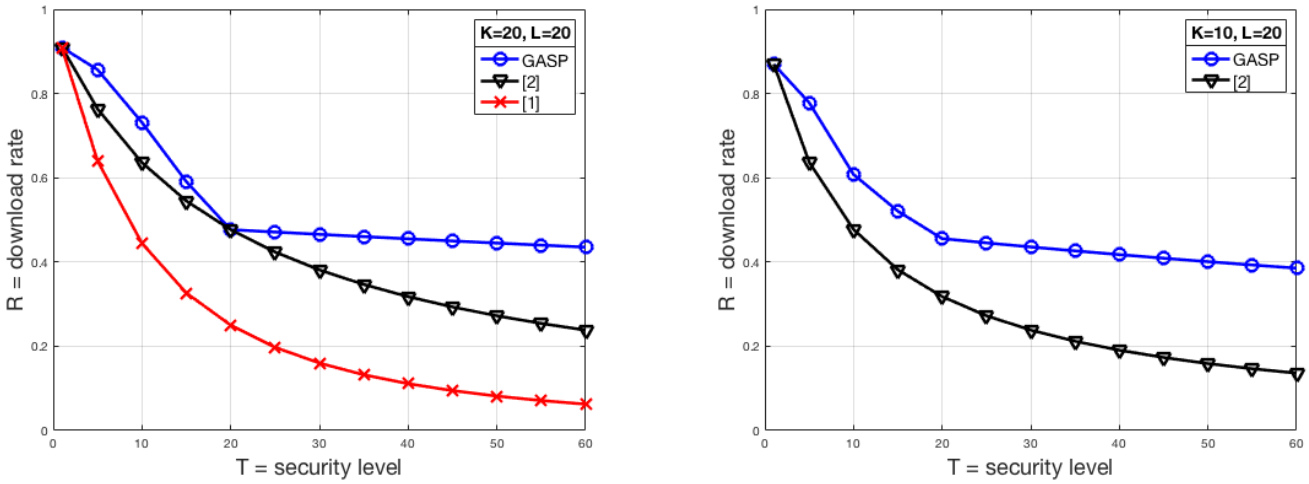

Fig. 1. Comparison of the Polynomial Code GASP with that of [1] and [2]. We plot the rate of the schemes for $K = 20$ and $L = 20$ on the left, and $K = 10$ and $L = 20$ on the right.

to interpolate $h(x)$ with substantially fewer than the expected number $\deg(h(x)) + 1$ of evaluations. While the polynomials $h(x)$ from the current work and those of [1], [2] have different degrees for the same parameters K, L, and T, this extra flexibility still allows us to generally use substantially fewer servers than the polynomial codes of [1], [2].

II. Motivating Example: $K = L = 3$ and $T = 2$

We begin our scheme description with the following example, which we present in as much detail as possible to showcase the essential ingredients of the scheme. In this example a user wishes to multiply two matrices A and B over a finite field $\mathbb{F}_q$, which are selected independently and uniformly at random from their respective ambient spaces. The user partitions the matrices as:

$$A = \begin{bmatrix} A_1 \\ A_2 \\ A_3 \end{bmatrix}, \quad B = \begin{bmatrix} B_1 & B_2 & B_3 \end{bmatrix}$$

so that all products $A_k B_\ell$ are well-defined and of the same size. The product AB is given by

$$AB = \begin{bmatrix} A_1B_1 & A_1B_2 & A_1B_3 \\ A_2B_1 & A_2B_2 & A_2B_3 \\ A_3B_1 & A_3B_2 & A_3B_3 \end{bmatrix}$$

We construct a scheme which computes each term $A_k B_\ell$, and therefore all of AB, via polynomial interpolation. The scheme is private against any $T = 2$ servers colluding to deduce the identities of A and B, and uses a total of $N = 18$ servers.

Let R_1 and R_2 be two matrices picked independently and uniformly at random with entries in $\mathbb{F}_q$, both of size equal to the A_k. Similarly, pick S_1 and S_2 independently and uniformly at random of size equal to that of the B_ℓ. Define polynomials

$$f(x) = A_1 x^{\alpha_1} + A_2 x^{\alpha_2} + A_3 x^{\alpha_3} + R_1 x^{\alpha_4} + R_2 x^{\alpha_5}$$
$$g(x) = B_1 x^{\beta_1} + B_2 x^{\beta_2} + B_3 x^{\beta_3} + S_1 x^{\beta_4} + S_2 x^{\beta_5}$$

where the α_k and β_ℓ are natural numbers that will be determined shortly.

As in [1], we will recover the products $A_k B_\ell$ by interpolating the product $h(x) = f(x)g(x)$. Specifically, for some evaluation points $a_n \in \mathbb{F}_q$, we will send $f(a_n)$ and $g(a_n)$ to server $n = 1, \ldots, N$, who then responds with $h(a_n) = f(a_n)g(a_n)$. These evaluations will suffice to interpolate all of $h(x)$. In particular, we will be able to retrieve the coefficients of $h(x)$, which in turn will allow us to decode all the $A_k B_\ell$.

The product $h(x) = f(x)g(x)$ is given by

$$h(x) = \sum_{1\le k,\ell\le 3} A_k B_\ell x^{\alpha_k+\beta_\ell} + \sum_{\substack{1\le k\le 3 \\ 4\le \ell\le 5}} A_k S_\ell x^{\alpha_k+\beta_\ell} + \sum_{\substack{4\le k\le 5 \\ 1\le \ell\le 3}} B_\ell R_k x^{\alpha_k+\beta_\ell} + \sum_{4\le k,\ell\le 5} R_k S_\ell x^{\alpha_k+\beta_\ell}$$

We wish to assign the exponents α_k and β_ℓ to guarantee decodability. Consider the following condition on the exponents:

$$\alpha_k + \beta_\ell \neq \alpha_{k'} + \beta_{\ell'},$$

for all $(k, \ell) \in [3] \times [3]$ and all $(k', \ell') \in [5] \times [5]$ such that $(k, \ell) \neq (k', \ell')$. That is, all of the exponents corresponding to the terms we wish to decode must be distinct from all the other exponents appearing in $h(x)$. This guarantees that each product $A_k B_\ell$ appears as the unique coefficient of a unique power of x. The immediate goal is to minimize the number of distinct powers of x appearing in $h(x)$, subject to the above condition. This will allow us to minimize the number of servers used by the scheme, thereby maximizing the rate.

The problem of assigning the α_k and β_ℓ can alternately be phrased as the following combinatorial problem. Consider the following addition table:

	β_1	β_2	β_3	β_4	β_5
α_1	$\alpha_1+\beta_1$	$\alpha_1+\beta_2$	$\alpha_1+\beta_3$	$\alpha_1+\beta_4$	$\alpha_1+\beta_5$
α_2	$\alpha_2+\beta_1$	$\alpha_2+\beta_2$	$\alpha_2+\beta_3$	$\alpha_2+\beta_4$	$\alpha_2+\beta_5$
α_3	$\alpha_3+\beta_1$	$\alpha_3+\beta_2$	$\alpha_3+\beta_3$	$\alpha_3+\beta_4$	$\alpha_3+\beta_5$
α_4	$\alpha_4+\beta_1$	$\alpha_4+\beta_2$	$\alpha_4+\beta_3$	$\alpha_4+\beta_4$	$\alpha_4+\beta_5$
α_5	$\alpha_5+\beta_1$	$\alpha_5+\beta_2$	$\alpha_5+\beta_3$	$\alpha_5+\beta_4$	$\alpha_5+\beta_5$

We call this table the *degree table* since it encodes the degrees that appear in $h(x) = f(x)g(x)$. With this in mind, we wish to pick $\alpha_k, \beta_\ell \in \mathbb{N}$ such that every term in the upper-left 3×3 block is distinct from every other number in the table. Outside this block, we wish to minimize the number of distinct integers that appear, in order to minimize the number of non-zero coefficients of $h(x)$ and therefore the number of required evaluation points.

Consider the assignment

$$\alpha_1 = 0,\ \alpha_2 = 1,\ \alpha_3 = 2,\ \alpha_4 = 9,\ \alpha_5 = 12$$

and

$$\beta_1 = 0,\ \beta_2 = 3,\ \beta_3 = 6,\ \beta_4 = 9,\ \beta_5 = 10,$$

for which the degree table becomes

	$\beta_1 = 0$	$\beta_2 = 3$	$\beta_3 = 6$	$\beta_4 = 9$	$\beta_5 = 10$
$\alpha_1 = 0$	0	3	6	9	10
$\alpha_2 = 1$	1	4	7	10	11
$\alpha_3 = 2$	2	5	8	11	12
$\alpha_4 = 9$	9	12	15	18	19
$\alpha_5 = 12$	12	15	18	21	22

which satisfies our decodability condition. Concretely, the polynomial $h(x)$ is now of the form

$$h(x) = A_1B_1 + \cdots + A_3B_3\ x^8 + C_9x^9 + C_{10}x^{10} + C_{11}x^{11} + \\ + C_{12}x^{12} + C_{15}x^{15} + C_{18}x^{18} + C_{19}x^{19} + C_{21}x^{21} + C_{22}x^{22}$$

which has $N = 18$ potentially non-zero coefficients. Here each C_j is a sum of products of matrices where each summand has either R_k or S_ℓ as a factor, and thus their precise nature is not important for decoding. We now show that over a suitable field $\mathbb{F}_q$, we can find $N = 18$ evaluation points a_n which suffice to interpolate $h(x)$, even though $\deg(h(x)) = 22$. This difference is subtle but crucial: the user knows exactly which coefficients of $h(x)$ are zero, and can thus interpolate the entire polynomial with fewer than the $\deg(h(x)) + 1$ evaluations one would normally need. This is in stark contrast with the strategies of [1], [2], where $f(x)$ and $g(x)$ are constructed so that every coefficient of $h(x)$ is non-zero (though for the same parameters K, L, and T, the polynomials $h(x)$ from [1], [2] are of a different degree than the $h(x)$ we obtain).

Let $\mathcal{J}$ be the set of exponents which occur in the above expression for $h(x)$, that is,

$$\mathcal{J} = \{0, \ldots, 8, 9, 10, 11, 12, 15, 18, 19, 21, 22\}$$

so that $|\mathcal{J}| = 18$. We wish to find an evaluation vector $\mathbf{a} = (a_1, \ldots, a_N) \in \mathbb{F}_q^N$ such that the 18×18 generalized Vandermonde matrix

$$GV(\mathbf{a}, \mathcal{J}) = \left[a_n^j\right], \quad 1 \leq n \leq 18,\ j \in \mathcal{J}$$

is invertible. One can easily check that for $q = 29$, the assignment $a_n = n \pmod{29}$ for $n = 1, \ldots, 18$ results in $\det(GV(\mathbf{a}, \mathcal{J})) = 20 \neq 0 \pmod{29}$. Thus the coefficients of $h(x)$, in particular the A_kB_ℓ, are uniquely decodable in the current scheme.

It is perhaps not obvious that the scheme we have described satisfies the 2-privacy condition. Let us show that this is indeed the case. As in Example 1 in [1], the 2-privacy condition will be satisfied provided that the matrices

$$P_{n,m} = \begin{bmatrix} a_n^9 & a_n^{12} \\ a_m^9 & a_m^{12} \end{bmatrix}, \quad Q_{n,m} = \begin{bmatrix} a_n^9 & a_n^{10} \\ a_m^9 & a_m^{10} \end{bmatrix}$$

are invertible for any pair $1 \leq n \neq m \leq 18$. We compute

$$\det(Q_{n,m}) = a_n^9 a_m^9 (a_m - a_n)$$

and

$$\det(P_{n,m}) = a_n^9 a_m^9 (a_m^3 - a_n^3).$$

Thus, provided that $a_n^3 \neq a_m^3$ for all $n \neq m$, and none of the a_n are zero, these matrices will all be invertible. However, we have $\gcd(3, q-1) = 1$, thus the map $x \mapsto x^3$ is a bijection from $\mathbb{F}_{29}$ to itself. Thus $a_n \neq a_m$ implies that $a_n^3 \neq a_m^3$ for all $n \neq m$, and we see that the determinants of the above matrices are all non-zero and hence the 2-privacy condition is satisfied.

For $K = L = 3$ and $T = 2$, let $\mathcal{R}_1$ denote the download rate of [2] and $\mathcal{R}_2$ that of [1]. We have

$$\mathcal{R}_1 = \frac{K^2}{(K+T)(K+1) - 1} = \frac{9}{19}$$

and

$$\mathcal{R}_2 = \frac{K^2}{(K+T)^2} = \frac{9}{25}$$

whereas the scheme we have presented above improves on these constructions to achieve a rate of

$$\mathcal{R} = \frac{K^2}{K^2 + 2K + T^2 + T - 3} = \frac{9}{18} = \frac{1}{2}.$$

While this improvement in this example is marginal, we will see later that for large parameters we achieve significant gains over the polynomial codes of [1], [2].

III. Polynomial Codes

Let A and B be matrices over a finite field $\mathbb{F}_q$, selected by a user independently and uniformly at random from the set of all matrices of their respective sizes, and partitioned as in equation (1) so that all products A_kB_ℓ are well-defined and of the same size. Then AB is the block matrix $AB = (A_kB_\ell)_{1 \leq k \leq K, 1 \leq \ell \leq L}$. A *polynomial code* is a tool for computing the product AB in a distributed manner, by computing each block A_kB_ℓ. Formally, we define a polynomial code as follows.

Definition 1: The *polynomial code* $\mathsf{PC}(K, L, T, N, \alpha, \beta)$ consists of the following data:

(i) positive integers K, L, T, and N,
(ii) $\alpha = (\alpha_1, \ldots, \alpha_{K+T}) \in \mathbb{N}^{K+T}$, and
(iii) $\beta = (\beta_1, \ldots, \beta_{L+T}) \in \mathbb{N}^{L+T}$.

A polynomial code $\mathsf{PC}(K, L, T, N, \alpha, \beta)$ is used to securely compute the product AB as follows. A user chooses T matrices R_t over $\mathbb{F}_q$ of the same size as the A_k independently and uniformly at random, and T matrices S_t of the same size as the B_ℓ independently and uniformly at random. They define polynomials $f(x)$ and $g(x)$ by

$$f(x) = \sum_{k=1}^{K} A_k x^{\alpha_k} + \sum_{t=1}^{T} R_t x^{\alpha_{K+t}}$$

and

$$g(x) = \sum_{\ell=1}^{L} B_\ell x^{\beta_\ell} + \sum_{t=1}^{T} S_t x^{\beta_{L+t}}$$

and let

$$h(x) = f(x)g(x). \tag{6}$$

Given N servers, a user chooses evaluation points $a_1, \ldots, a_N \in \mathbb{F}_{q^r}$ in some finite extension $\mathbb{F}_{q^r}$ of $\mathbb{F}_q$. They then send $f(a_n)$ and $g(a_n)$ to server $n = 1, \ldots, N$, who computes the product $f(a_n)g(a_n) = h(a_n)$ and transmits it back to the user. The user then interpolates the polynomial $h(x)$ given all of the evaluations $h(a_n)$, and attempts to recover all products $A_k B_\ell$ from the coefficients of $h(x)$. We omit the evaluation vector $\mathbf{a}$ from the notation $\mathsf{PC}(K, L, T, N, \alpha, \beta)$ because as we will shortly show, it does not really affect any important analysis of the polynomial code.

Definition 2: A polynomial code $\mathsf{PC}(K, L, T, N, \alpha, \beta)$ is *decodable and T-secure* if there exists some evaluation vector $\mathbf{a} = (a_1, \ldots, a_N) \in \mathbb{F}_{q^r}^N$ for some $r > 0$ such that for any A and B as above, the following two conditions hold.

(i) (Decodability) All products $A_k B_\ell$ for $k = 1, \ldots, K$ and $\ell = 1, \ldots, L$ are completely determined by the evaluations $h(a_n)$ for $n = 1, \ldots, N$.

(ii) (T-security) For any T-tuple $\{n_1, \ldots, n_T\} \subseteq [N]$, we have

$$I(f(a_{n_1}), g(a_{n_1}), \ldots, f(a_{n_T}), g(a_{n_T}); A, B) = 0.$$

where $I(\cdot;\cdot)$ denotes mutual information between two random variables.

Definition 3: Suppose that the polynomial code $\mathsf{PC}(K, L, T, N, \alpha, \beta)$ is decodable and T-secure. The *download rate*, or simply the *rate*, of this polynomial code is defined to be

$$\mathcal{R} = \frac{KL}{N}.$$

Given parameters K, L, and T, the goal of polynomial coding is to construct a decodable and T-secure polynomial code $\mathsf{PC}(K, L, T, \alpha, \beta)$ with download rate as large as possible. This is equivalent to minimizing the number of servers N, or equivalently, the number of evaluation points needed by the code.

IV. Degree Table

In this section we relate the construction of polynomial codes for SDMM with a certain combinatorial problem. This connection will guide our constructions and aid us in proving that our polynomial codes are decodable and T-secure.

Definition 4: Let $\alpha \in \mathbb{N}^{K+T}$ and $\beta \in \mathbb{N}^{L+T}$. The *outer sum* $\alpha \oplus \beta \in \mathbb{N}^{(K+T)\times(L+T)}$ of α and β is defined to be the matrix

$$\alpha \oplus \beta = \begin{bmatrix} \alpha_1 + \beta_1 & \cdots & \alpha_1 + \beta_{L+T} \\ \vdots & \ddots & \vdots \\ \alpha_{K+T} + \beta_1 & \cdots & \alpha_{K+T} + \beta_{L+T} \end{bmatrix}.$$

Definition 5: Let $\alpha \in \mathbb{N}^{K+T}$ and $\beta \in \mathbb{N}^{L+T}$. We say that the outer sum $\alpha \oplus \beta$ is *decodable and T-secure* if the following two conditions hold:

(i) (Decodability) $(\alpha \oplus \beta)_{k,\ell} \neq (\alpha \oplus \beta)_{k',\ell'}$ for all $(k, \ell) \in [K] \times [L]$ and all $(k', \ell') \in [K + T] \times [L + T]$.

(ii) (T-security) $\alpha_{K+t} \neq \alpha_{K+t'}$ and $\beta_{L+t} \neq \beta_{L+t'}$ for every $t \neq t' \in [T]$.

Constructing α and β so that $\alpha \oplus \beta$ is decodable and T-secure can be realized as the following combinatorial problem, displayed in Table II. The condition of decodability from Definition 5 simply states that each $\alpha_k + \beta_\ell$ in the red block must be distinct from every other entry in $\alpha \oplus \beta$. The condition of T-security states that all α_{K+t} in the green block must be pairwise distinct, and all β_{L+t} in the blue block must be pairwise distinct. We refer to this table as the degree table.

Definition 6: Let A be a matrix with entries in $\mathbb{N}$. We define the *terms* of A to be the set

$$\mathrm{t}(A) = \{n \in \mathbb{N} : \exists (i, j), A_{ij} = n\}.$$

The next lemma and theorem allow us to reduce the construction of Polynomial Codes for SDMM to the combinatorial problem of constructing α and β such that the degree table, $\alpha \oplus \beta$, is decodable and T-secure. The proof of the lemma is straightforward and thus omitted.

Lemma 1: Consider the polynomial code $\mathsf{PC}(K, L, T, N, \alpha, \beta)$, with associated polynomials

$$f(x) = \sum_{k=1}^{K} A_k x^{\alpha_k} \quad \text{and} \quad g(x) = \sum_{\ell=1}^{L} B_\ell x^{\beta_\ell}.$$

Then we can express the product $h(x)$ of $f(x)$ and $g(x)$ as

$$h(x) = f(x)g(x) = \sum_{j \in \mathcal{J}} C_j x^j \tag{7}$$

for some matrices C_j, where $\mathcal{J} = \mathrm{t}(\alpha \oplus \beta)$.

Thus, the terms in the outer sum $\alpha \oplus \beta$ correspond to the terms in the polynomial $h(x) = f(x)g(x)$. Because of this, we refer to the table representation of $\alpha \oplus \beta$ in Table II as the *degree table* of the polynomial code $\mathsf{PC}(K, L, T, N, \alpha, \beta)$. The following theorem allows us to reduce the construction of polynomial codes to the construction of degree tables which are decodable and T-secure.

Theorem 1: Let $\mathsf{PC}(K, L, T, N, \alpha, \beta)$ be a polynomial code, where $N = |\,\mathrm{t}(\alpha \oplus \beta)|$. Suppose that the degree table, $\alpha \oplus \beta$, satisfies the decodability and T-security conditions of Definition 5. Then the polynomial code $\mathsf{PC}(K, L, T, N, \alpha, \beta)$ is decodable and T-secure.

Proof: The proof is an application of the Schwarz-Zippel Lemma. One finds sufficient conditions for decodability and T-security that reduce to the simultaneous non-vanishing of $2\binom{N}{T} + 1$ determinants. One can find a point $\mathbf{a} \in \mathbb{F}_{q^r}^N$ for some $r > 0$ at which none of these polynomials is zero. We relegate a detailed proof to the Appendix. ∎

Thanks to Theorem 1, constructing a polynomial code scheme for secure distributed matrix multiplication can be done by constructing α and β such that the degree table, $\alpha \oplus \beta$, is decodable and T-secure. For this reason, the visualization in Table II is extremely useful, both as a guide for constructing polynomial codes for SDMM and as a method for calculating the corresponding download rate. In this context, maximizing the download rate is equivalent to minimizing $|\,\mathrm{t}(\alpha \oplus \beta)|$,

TABLE II

THE COMBINATORIAL PROBLEM OF CONSTRUCTING α AND β SO THAT $\alpha \oplus \beta$ IS DECODABLE AND T-SECURE

	β_1	$\cdots$	β_L	β_{L+1}	$\cdots$	β_{L+T}
α_1	$\alpha_1+\beta_1$	$\cdots$	$\alpha_1+\beta_L$	$\alpha_1+\beta_{L+1}$	$\cdots$	$\alpha_1+\beta_{L+T}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
α_K	$\alpha_K+\beta_1$	$\cdots$	$\alpha_K+\beta_L$	$\alpha_K+\beta_{L+1}$	$\cdots$	$\alpha_K+\beta_{L+T}$
α_{K+1}	$\alpha_{K+1}+\beta_1$	$\cdots$	$\alpha_{K+1}+\beta_L$	$\alpha_{K+1}+\beta_{L+1}$	$\cdots$	$\alpha_{K+1}+\beta_{L+T}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
α_{K+T}	$\alpha_{K+T}+\beta_1$	$\cdots$	$\alpha_{K+T}+\beta_L$	$\alpha_{K+T}+\beta_{L+1}$	$\cdots$	$\alpha_{K+T}+\beta_{L+T}$

TABLE III

THE DEGREE TABLE, $\alpha \oplus \beta$, FOR THE VECTORS α AND β AS PER (8)

	$\beta_1=0$	$\cdots$	$\beta_L=K(L-1)$	$\beta_{L+1}=KL$	$\beta_{L+2}=KL+1$	$\cdots$	$\beta_{L+T}=KL+T-1$
$\alpha_1=0$	0	$\cdots$	$K(L-1)$	KL	$KL+1$	$\cdots$	$KL+T-1$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$\alpha_K=K-1$	$K-1$	$\cdots$	$KL-1$	$KL+K-1$	$KL+K$	$\cdots$	$KL+K+T-2$
$\alpha_{K+1}=KL$	KL	$\cdots$	$2KL-K$	$2KL$	$2KL+1$	$\cdots$	$2KL+T-1$
$\alpha_{K+2}=KL+1$	$KL+1$	$\cdots$	$2KL-K+1$	$2KL+1$	$2KL+2$	$\cdots$	$2KL+T$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$\alpha_{K+T}=KL+T-1$	$KL+T-1$	$\cdots$	$2KL-K+T-1$	$2KL+T-1$	$2KL+T$	$\cdots$	$2KL+2T-2$

the number of distinct integers in the degree table shown in Table II, subject to decodability and T-security.

Remark 1: Suppose that the degree table, $\alpha \oplus \beta$, is decodable and T-secure, and let $\mathbb{K}$ be an algebraic closure of $\mathbb{F}_q$. One can show that the set of all $\mathbf{a} = (a_1, \ldots, a_N)$ such that $\mathsf{PC}(K, L, T, N, \alpha, \beta)$ is decodable and T-secure is a Zariski open subset of $\mathbb{K}^N$. In practice, this means that given K, L, T, α, β, if we choose $\mathbf{a} \in \mathbb{F}_{q^r}^N$ uniformly at random, then the probability that the polynomial code $\mathsf{PC}(K, L, T, N, \alpha, \beta)$ is decodable and T-secure goes to 1 as $r \to \infty$. Thus, finding such evaluation vectors is not a difficult task.

V. POLYNOMIAL CODE FOR BIG T

In this section, we construct a polynomial code, $\mathsf{GASP}_{\text{big}}$, which has better rate than all previous schemes in the literature. The scheme construction chooses α and β to attempt to minimize the number of distinct integers in the degree table, $\alpha \oplus \beta$. The scheme construction proceeds by choosing α_k and β_ℓ to belong to certain arithmetic progressions, and minimizes the number of terms in the lower-right $T \times T$ block of the degree table, $\alpha \oplus \beta$, shown in Table III.

Definition 7: Given K, L, and T, define the polynomial code $\mathsf{GASP}_{\text{big}}$ as follows. Let α and β be given by

$$\alpha_k = \begin{cases} k-1 & \text{if } 1 \le k \le K \\ KL+t-1 & \text{if } k = K+t,\ 1 \le t \le T \end{cases}, \tag{8}$$

$$\beta_\ell = \begin{cases} K(\ell-1) & \text{if } 1 \le \ell \le L \\ KL+t-1 & \text{if } \ell = L+t,\ 1 \le t \le T \end{cases} \tag{9}$$

if $L \le K$ and

$$\alpha_\ell = \begin{cases} K(\ell-1) & \text{if } 1 \le \ell \le L \\ KL+t-1 & \text{if } \ell = L+t,\ 1 \le t \le T \end{cases}, \tag{10}$$

$$\beta_k = \begin{cases} k-1 & \text{if } 1 \le k \le K \\ KL+t-1 & \text{if } k = K+t,\ 1 \le t \le T \end{cases} \tag{11}$$

if $K < L$.

Lastly, define $N = |\,\mathrm{t}(\alpha \oplus \beta)|$. Then $\mathsf{GASP}_{\text{big}}$ is defined to be the polynomial code $\mathsf{PC}(K, L, T, N, \alpha, \beta)$.

A. Decodability and T-Security

Theorem 2: The polynomial code $\mathsf{GASP}_{\text{big}}$ is decodable and T-secure.

Proof: We show that $\alpha \oplus \beta$ is decodable and T-secure, and the result then follows from Theorem 1. If $L \le K$, then α and β are as in (8). Suppose that $k \in [K]$ and $\ell \in [L]$, so that $\alpha_k + \beta_\ell = k - 1 + K(\ell - 1)$. As k and ℓ range over all of $[K]$ and $[L]$, respectively, each such number gives a unique integer in the interval $[0, KL-1]$. As every other term in the outer sum $\alpha \oplus \beta$ is greater than or equal to KL, we see that the decodability condition of Definition 5 is satisfied. As for T-security, it is clear that all α_{K+t} for $t \in [T]$ are distinct, and all β_{L+t} for $t \in [T]$ are distinct. Therefore the T-security condition of Definition 5 is satisfied. If $K < L$ then the same argument holds by interchanging α and β. ∎

B. Download Rate

To compute the number of terms in the degree table of $\mathsf{GASP}_{\text{big}}$, we divide the table into four regions.

- Upper Left: $\mathrm{UL} = \{(\alpha \oplus \beta)_{ij} : 1 \le i \le K, 1 \le j \le L\}$.
- Upper Right: $\mathrm{UR} = \{(\alpha \oplus \beta)_{ij} : 1 \le i \le K, L+1 \le j \le L+T\}$.
- Lower Left: $\mathrm{LL} = \{(\alpha \oplus \beta)_{ij} : K+1 \le i \le K+T, 1 \le j \le L\}$.

- Lower Right: $\mathrm{LR} = \{(\alpha \oplus \beta)_{ij} : K+1 \le i \le K+T, L+1 \le j \le L+T\}$.

Then, we compute the number of terms in each of these regions and use the inclusion-exclusion principle to obtain the number of terms in the whole table.

Theorem 3: Let $N = |\,\mathrm{t}(\alpha \oplus \beta)|$, where α and β are as in Definition 7. Then N is given by

$$N = \begin{cases} (K+T)(L+1)-1 & \text{if } T < K \\ 2KL + 2T - 1 & \text{if } T \ge K \end{cases} \tag{12}$$

if $L \le K$, and

$$N = \begin{cases} (L+T)(K+1)-1 & \text{if } T < L \\ 2KL + 2T - 1 & \text{if } T \ge L \end{cases} \tag{13}$$

if $K < L$.

Consequently, the polynomial code $\mathsf{GASP}_{\text{big}}(K, L, T)$ has rate $\mathcal{R} = KL/N$, where N is as in (12) or (13).

Proof: The degree table, $\alpha \oplus \beta$, is shown in Table III. We first prove for the case where $L \le K$. We denote by $[A : B]$ the set of all integers in the interval $[A, B]$. We can describe the terms of the four blocks of $\alpha \oplus \beta$ as follows:

$$\begin{aligned} \mathrm{t(UL)} &= [0 : KL-1] \\ \mathrm{t(UR)} &= [KL : KL+K+T-2] \\ \mathrm{t(LL)} &= \bigcup_{\ell=0}^{L-1} [KL + K(\ell-1) : KL + K(\ell-1) + T - 1] \\ \mathrm{t(LR)} &= [2KL : 2KL + 2T - 2] \end{aligned} \tag{14}$$

The sizes of these sets is given by

$$\begin{aligned} |\,\mathrm{t(UL)}| &= KL \\ |\,\mathrm{t(UR)}| &= K + T - 1 \\ |\,\mathrm{t(LL)}| &= \begin{cases} LT & \text{if } T \le K \\ KL - K + T & \text{if } T \ge K \end{cases} \\ |\,\mathrm{t(LR)}| &= 2T - 1 \end{aligned} \tag{15}$$

Since the largest term in UL is smaller than any term on the other blocks, $\mathrm{t(UL)}$ is disjoint from the terms of the other blocks. One then observes that the pairwise intersections of the sets of terms of the blocks are given by

$$\begin{aligned} \mathrm{t(UR)} \cap \mathrm{t(LL)} &= \begin{cases} [K : K+T-1] & \text{if } L = 1 \\ I_1 & \text{if } L \ge 2 \end{cases} \\ \mathrm{t(LL)} \cap \mathrm{t(LR)} &= [2KL : 2KL - K + T - 1] \\ \mathrm{t(UR)} \cap \mathrm{t(LR)} &= [2KL : KL + K + T - 2], \end{aligned} \tag{16}$$

where $I_1 = [KL : KL+T-1] \cup [KL+K : KL+K+T-2]$. The sizes of these pairwise intersections are now calculated to be

$$|\,\mathrm{t(UR)} \cap \mathrm{t(LL)}| = \begin{cases} T & \text{if } L = 1 \\ 2T-1 & \text{if } L \ge 2, T \le K \\ K+T-1 & \text{if } L \ge 2, T \ge K \end{cases}$$

$$|\,\mathrm{t(LL)} \cap \mathrm{t(LR)}| = \begin{cases} 0 & T \le K \\ T-K & T \ge K \end{cases}$$

$$|\,\mathrm{t(UR)} \cap \mathrm{t(LR)}| = \begin{cases} 0 & T \le K(L-1)+1 \\ I_2 & T \ge K(L-1)+1 \end{cases},$$

where $I_2 = T - (K(L-1)+1)$. Finally, the triple intersection is given by

$$\begin{aligned} &\mathrm{t(UR)} \cap \mathrm{t(LL)} \cap \mathrm{t(LR)} = \\ &\quad [2KL : \min\{2KL - K + T - 1, KL + K + T - 2\}] \end{aligned}$$

We have $2KL - K + T - 1 \le KL + K + T - 2$ if and only if $L = 1$. One now computes that

$$|\cap| = \begin{cases} 0 & \text{if } L = 1, T \le K \\ T-K & \text{if } L = 1, T \ge K \\ 0 & \text{if } L \ge 2, T \le K(L-1)+1 \\ T-(K(L-1)+1) & \text{if } L \ge 2, T \ge K(L-1)+1 \end{cases}$$

where $\cap = \mathrm{t(UR)} \cap \mathrm{t(LL)} \cap \mathrm{t(LR)}$.

We can now compute $N = |\,\mathrm{t}(\alpha \oplus \beta)|$ by using the inclusion-exclusion principle, as

$$\begin{aligned} N = |\,\mathrm{t}(\alpha \oplus \beta)| &= |\,\mathrm{t(UL)}| + |\,\mathrm{t(UR)}| + |\,\mathrm{t(LL)}| + |\,\mathrm{t(LR)}| \\ &\quad - |\,\mathrm{t(UR)} \cap \mathrm{t(LL)}| - |\,\mathrm{t(LL)} \cap \mathrm{t(LR)}| \\ &\quad - |\,\mathrm{t(UR)} \cap \mathrm{t(LR)}| \\ &\quad + |\,\mathrm{t(UR)} \cap \mathrm{t(LR)} \cap \mathrm{t(LL)}|. \end{aligned}$$

For $K < L$, the proof is analogous by interchanging α and β. ∎

Remark 2: If we take $K = L = 1$, then the polynomial code $\mathsf{GASP}_{\text{big}}(1, 1, T)$ uses $N = 2T+1$ servers. Thus for any N we can construct a polynomial code for any $T \le \lfloor \frac{N-1}{2} \rfloor$, which is the same range of allowable T as in [2].

C. Performance

We now compare $\mathsf{GASP}_{\text{big}}$ with the polynomial codes of [1] and [2]. Indeed, we show that it outperforms them for most parameters. To do this it suffices, because of (5), to show that N, as defined in Theorem 3, is smaller than $(K+T)(L+1)-1$.

Theorem 4: Let N be defined as in Theorem 3. Then, $N \le (K+T)(L+1)-1$.

Proof: Suppose $L \le K$. Then N is as in (12). We will analyze each case.

- If $T < K$: then $(K+T)(L+1)-1 = N$.
- If $T \ge K$: then $0 \le (L-1)(T-K) = ((K+T)(L+1)-1) - (2KL+2T-1)$.
  Thus, $(K+T)(L+1)-1 \ge 2KL+2T-1 = N$.

The result for $K < L$ follows by switching the roles of K and L in the above calculation. ∎

VI. Polynomial Code for Small T

In this section, we construct a polynomial code $\mathsf{GASP}_{\text{small}}$ which outperforms $\mathsf{GASP}_{\text{big}}$ when $T < \min\{K, L\}$. This is done by choosing the α_k and β_ℓ to lie in certain arithmetic progressions so that the columns of the upper-right block of $\alpha \oplus \beta$, shown in Table IV, overlap as much as possible, and similarly for the columns of the lower-left block. For T small relative to K and L, these two blocks are much bigger than the lower-right block, which the scheme construction essentially ignores.

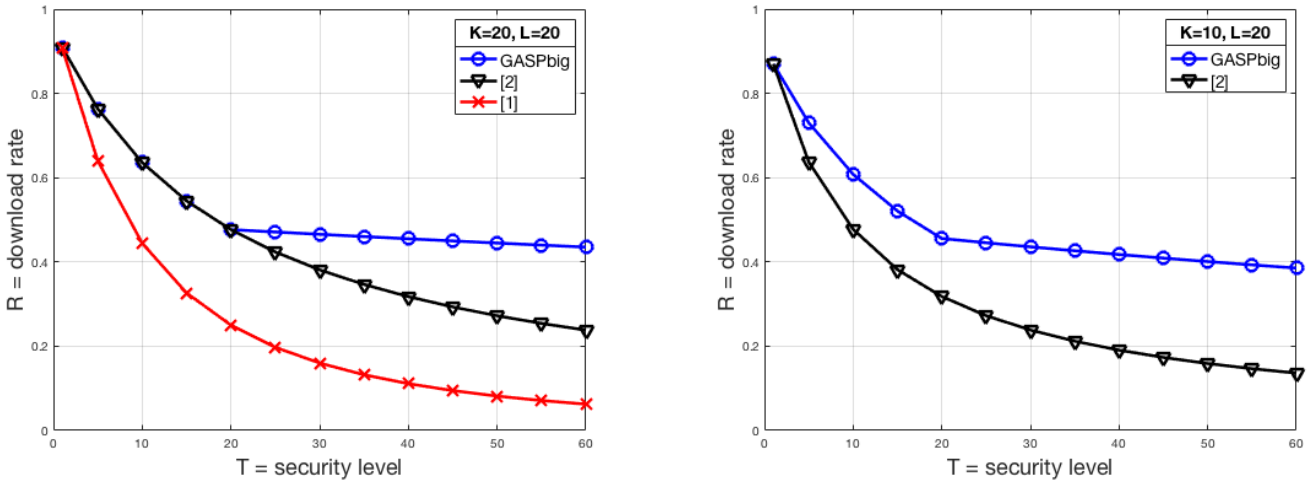

Fig. 2. Comparison of the Polynomial Code $\mathsf{GASP}_{\text{big}}$ with that of [2]. We plot the rate of both schemes for $K = L = 20$ on the left, and $K = 10, L = 20$ on the right.

TABLE IV
THE DEGREE TABLE, $\alpha \oplus \beta$, OF THE VECTORS α AND β AS PER DEFINITION 8

	$\beta_1 = 0$	$\cdots$	$\beta_L = K(L-1)$	$\beta_{L+1} = KL$	$\beta_{L+2} = KL+1$	$\cdots$	$\beta_{L+T} = KL+T-1$
$\alpha_1 = 0$	0	$\cdots$	$K(L-1)$	KL	$KL+1$	$\cdots$	$KL+T-1$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$\alpha_K = K-1$	$K-1$	$\cdots$	$KL-1$	$KL+K-1$	$KL+K$	$\cdots$	$KL+K+T-2$
$\alpha_{K+1} = KL$	KL	$\cdots$	$2KL-K$	$2KL$	$2KL+1$	$\cdots$	$2KL+T-1$
$\alpha_{K+2} = KL+K$	$KL+K$	$\cdots$	$2KL$	$2KL+K$	$2KL+K+1$	$\cdots$	$2KL+K+T-1$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$\alpha_{K+T} = KL+K(T-1)$	$KL+K(T-1)$	$\cdots$	$2KL+K(T-2)$	$2KL+K(T-1)$	$2KL+K(T-1)+1$	$\cdots$	$2KL+(K+1)(T-1)$

Definition 8: Given K, L, and T, define the polynomial code $\mathsf{GASP}_{\text{small}}$ as follows. Let α and β be given by

$$\alpha_k = \begin{cases} k-1 & \text{if } 1 \le k \le K \\ KL + K(t-1) & \text{if } k = K+t,\ 1 \le t \le T \end{cases}, \quad (17)$$

$$\beta_\ell = \begin{cases} K(\ell-1) & \text{if } 1 \le \ell \le L \\ KL + t - 1 & \text{if } \ell = L+t,\ 1 \le t \le T. \end{cases} \quad (18)$$

if $K \le L$, and

$$\alpha_\ell = \begin{cases} K(\ell-1) & \text{if } 1 \le \ell \le L \\ KL + t - 1 & \text{if } \ell = L+t,\ 1 \le t \le T. \end{cases}, \quad (19)$$

$$\beta_k = \begin{cases} k-1 & \text{if } 1 \le k \le K \\ KL + K(t-1) & \text{if } k = K+t,\ 1 \le t \le T \end{cases} \quad (20)$$

if $L < K$.

Lastly, define $N = |\,\mathrm{t}(\alpha \oplus \beta)|$. Then $\mathsf{GASP}_{\text{small}}$ is defined to be the polynomial code $\mathsf{PC}(K, L, T, N, \alpha, \beta)$.

The example in Section II is exactly the polynomial code $\mathsf{GASP}_{\text{small}}$ when $K = L = 3$ and $T = 2$. In what follows we show that $\mathsf{GASP}_{\text{small}}$ is decodable and T-secure, and compute its download rate. Throughout this section, α and β will be as in Definition 8.

A. Decodability and T-Security

Theorem 5: The polynomial code $\mathsf{GASP}_{\text{small}}(K, L, T)$ is decodable and T-secure.

Proof: Analogous to Theorem 2. ∎

B. Download Rate

We now find the download rate of $\mathsf{GASP}_{\text{small}}$ by computing $N = |\,\mathrm{t}(\alpha \oplus \beta)|$.

Theorem 6: Let $N = |\,\mathrm{t}(\alpha \oplus \beta)|$, where α and β are as in Definition 8. Then N is given by

$$N = \begin{cases} 2K + T^2 & \text{if } L = 1, T < K \\ KT + K + T & \text{if } L = 1, T \ge K \\ KL + K + L & \text{if } L \ge 2, 1 = T < K \\ I_1 & \text{if } L \ge 2, 2 \le T < K \\ I_2 & \text{if } L \ge 2, K \le T \le K(L-1)+1 \\ I_3 & \text{if } L \ge 2, K(L-1)+1 \le T \end{cases} \quad (21)$$

if $K \le L$, and

$$N = \begin{cases} 2L + T^2 & \text{if } K = 1, T < L \\ LT + L + T & \text{if } K = 1, T \ge L \\ KL + K + L & \text{if } K \ge 2, 1 = T < L \\ I_1 & \text{if } K \ge 2, 2 \le T < L \\ I_4 & \text{if } K \ge 2, L \le T \le L(K-1)+1 \\ I_5 & \text{if } K \ge 2, L(K-1)+1 \le T \end{cases} \quad (22)$$

if $L < K$, where

$$I_1 = KL + K + L + T^2 + T - 3,$$

$$I_2 = KL + KT + L + 2T - 3 - \left\lfloor \frac{T-2}{K} \right\rfloor,$$

$$I_3 = 2KL + KT - K + T,$$

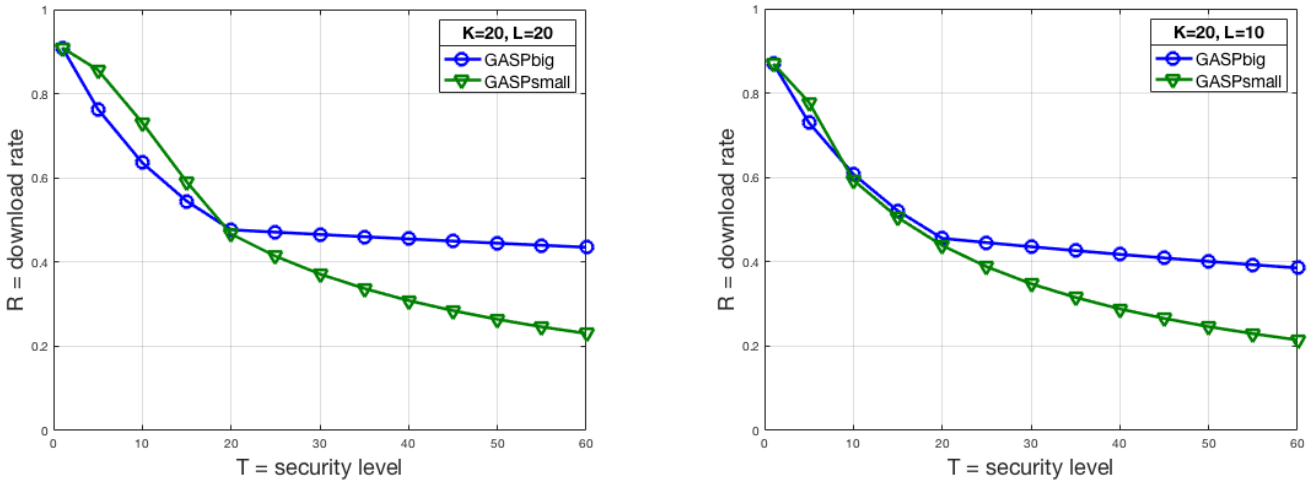

Fig. 3. Comparison between $\mathsf{GASP}_{\text{small}}$ and $\mathsf{GASP}_{\text{big}}$. We plot the rate of both schemes for $K = 20$ and $L = 20$ on the left, and $K = 20$ and $L = 10$ on the right. As shown, $\mathsf{GASP}_{\text{small}}$ outperforms $\mathsf{GASP}_{\text{big}}$ for $T < \max\{K, L\}$.

$$I_4 = KL + LT + K + 2T - 3 - \left\lfloor \frac{T-2}{L} \right\rfloor,$$
$$I_5 = 2KL + LT - L + T.$$

Consequently, the polynomial code $\mathsf{GASP}_{\text{small}}$ has rate $\mathcal{R} = KL/N$, where N is as in (21) or (22).

Proof: The proof is in Appendix B. ∎

Remark 3: The above construction for small T is from where the name GASP (Gap Additive Secure Polynomial) is derived. The construction allows for gaps in the degrees of monomials appearing as summands of $h(x)$, as was observed in the example in Section II. Allowing for these gaps gives one more flexibility in how the vectors α and β are chosen to attempt to minimize $N = |\,\mathrm{t}(\alpha \oplus \beta)|$. Note that for very large T, the inequality $T \geq K(L-1)+1$ has forced the outer sum $\alpha \oplus \beta$ to contain every integer from 0 to $2KL+(K+1)(T-1)$, with no gaps.

C. Performance

We now show that $\mathsf{GASP}_{\text{small}}$ outperforms $\mathsf{GASP}_{\text{big}}$ when $T < \min\{K, L\}$.

Theorem 7: Let $T < \min\{K, L\}$. Then $N_{\text{small}} \leq N_{\text{big}}$.

Proof: We will analyze each case.

- If $T = 1$: then $N_{\text{big}} = KL + K + L = N_{\text{small}}$.
- If $T \geq 2$: then $0 \leq (T+1)(T-1) - T^2 + 2 \leq L(T-1) - T^2 + 2 = N_{\text{big}} - N_{\text{small}}$.

  Thus, $N_{\text{big}} \geq N_{\text{small}}$.

∎

VII. Combining Both Schemes

In this section, we construct a polynomial, GASP, by combining both $\mathsf{GASP}_{\text{small}}$ and $\mathsf{GASP}_{\text{big}}$. By construction, GASP has a better rate than all previous schemes.

Definition 9: Given K, L, and T, we define the polynomial code GASP to be

$$\mathsf{GASP} = \begin{cases} \mathsf{GASP}_{\text{small}} & \text{if } T < \min\{K, L\} \\ \mathsf{GASP}_{\text{big}} & \text{if } T \geq \min\{K, L\}. \end{cases} \tag{23}$$

Theorem 8: For $L \leq K$, the polynomial code GASP has rate,

$$\mathcal{R} = \begin{cases} \dfrac{KL}{KL+K+L} & \text{if } 1 = T < L \leq K \\ \dfrac{KL}{KL+K+L+T^2+T-3} & \text{if } 2 \leq T < L \leq K \\ \dfrac{KL}{(K+T)(L+1)-1} & \text{if } L \leq T < K \\ \dfrac{KL}{2KL+2T-1} & \text{if } L \leq K \leq T \end{cases}$$

For $K < L$, the rate is given by interchanging K and L.

Proof: Follows immediately from Theorems 3, 6, and 7. ∎

A. Fixed Computation Load

We now compare the rate of GASP with those of [1] and [2] when K and L are fixed. Throughout this section, we let

$$\mathcal{R}_1 = \frac{K^2}{(K+T)^2} \quad \text{and} \quad \mathcal{R}_2 = \frac{KL}{(K+T)(L+1)-1}. \tag{24}$$

Here $\mathcal{R}_1$ and $\mathcal{R}_2$ are the rates of the polynomial codes in [1] and [2], respectively.

B. Fixed Number of Workers

To deepen the comparison with [2], we plot the download rates $\mathcal{R}$ and $\mathcal{R}_2$ as functions of the total number N of servers and the security level T. For GASP and the polynomial code of [2], given some N and T, we must calculate a K and L for which the expression for the required number of servers is less than the given N, and which ideally maximizes the rate function. In [2, Theorem 1], the authors propose the solution

$$\hat{L} = \max\left\{1, \left\lceil -\frac{3}{2} + \sqrt{\frac{1}{4} + \frac{N}{T}} \right\rceil \right\}, \quad \hat{K} = \left\lfloor \frac{N+1}{\hat{L}+1} - T \right\rfloor \tag{25}$$

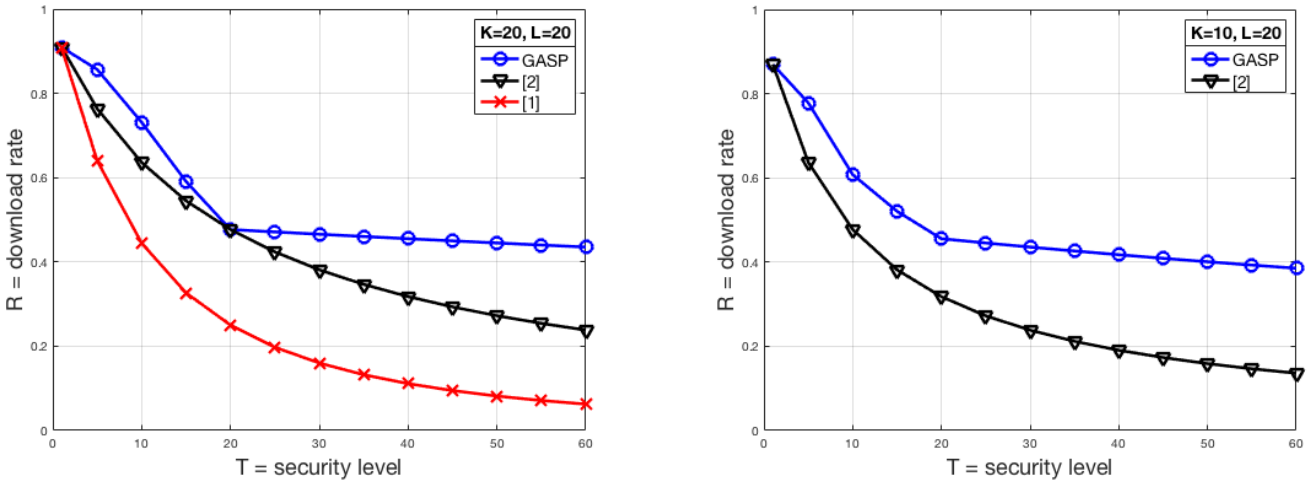

Fig. 4. Comparison of the Polynomial Code GASP with that of [1] and [2]. We plot the rate of the schemes for $K = 20$ and $L = 20$ on the left, and $K = 10$ and $L = 20$ on the right.

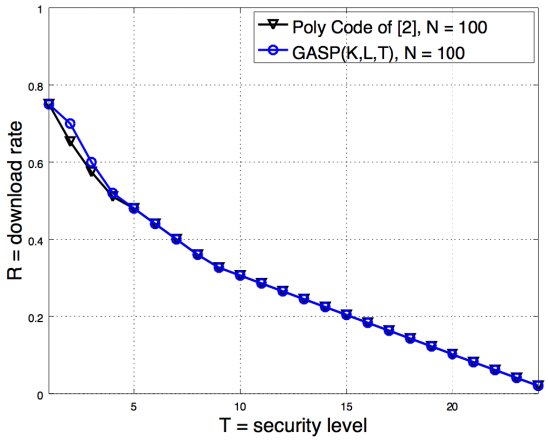

Fig. 5. The rate of GASP and of the the polynomial code of [2], as a function of the level of security T, for $N = 50$ servers. The optimal rate of GASP for a given N and T was computed by finding a solution to (26) by brute force.

which, for a given N and T, is shown to satisfy $(\hat{K} + T)(\hat{L} + 1) - 1 \leq N$ and nearly maximize the rate function $\mathcal{R}_1$.

For a given N and T, optimizing the rate of GASP presents one with the following optimization problem:

$$\begin{aligned} \max_{K,L} \quad & \mathcal{R}_{\max} = \frac{KL}{\min\{N_{\text{small}}, N_{\text{big}}\}} \\ \text{subject to} \quad & \min\{N_{\text{small}}, N_{\text{big}}\} \leq N \end{aligned} \tag{26}$$

Due to the complicated nature of the expressions for N_{small} and N_{big}, we will not attempt to solve this optimization problem analytically. Instead, for the purposes of the present comparison with [2], we simply solve (26) by brute force for each specific value of N and T.

In Fig. 5 we plot the download rate of GASP versus the download rate of the polynomial code of [2], for $N = 50$ and $N = 100$ servers. The optimal values of K and L for GASP were computed by solving (26) by brute force. The values of K and L for the scheme of [2] were those of (25).

The apparent equality in rate of the two schemes outside of the 'small T' regime can be explained as follows. One can show easily that when $T > N/6$ we have $\hat{L} = 1$, and hence the rate from [2] is given by $\mathcal{R}_2 = \frac{\hat{K}}{2\hat{K}+2T-1}$, where $\hat{K} = \left\lfloor \frac{N+1}{2} - T \right\rfloor$. Now the rate of GASP in this regime is that of GASP_{big}, so $\mathcal{R} = \frac{KL}{2KL+2T-1}$. Optimizing the rate of GASP_{big} for fixed N is now simply a matter of picking the optimal value of KL. Whatever this optimal value happens to be, it only depends on the product KL and not the individual values of K and L. So when optimizing the rate of GASP_{big} for fixed N and T, one is free to set $L = 1$ without loss of generality. The rates of the GASP_{big} and the scheme of [2] are then easily seen to agree.

APPENDIX

A. Proof of Theorem 1

We will require the following definition throughout the proof of Theorem 1.

Definition 10: Let $\mathbb{F}_q$ be a finite field, let $\mathbf{a} = (a_1, \ldots, a_N) \in \mathbb{F}_q^N$, and let $\mathcal{J}$ be a set of non-negative integers of size $|\mathcal{J}| = N$. We define the *Generalized Vandermonde Matrix* $GV(\mathbf{a}, \mathcal{J}) \in \mathbb{F}_q^{N \times N}$ to be

$$GV(\mathbf{a}, \mathcal{J}) = \left[a_n^j\right], \quad 1 \leq n \leq N, \ j \in \mathcal{J}.$$

Note that if $\mathcal{J} = \{0, 1, \ldots, N-1\}$ and the a_n are all chosen distinct, then $GV(\mathbf{a}, \mathcal{J})$ is the familiar $N \times N$ Vandermonde matrix associated with the a_n, and is invertible if and only if the a_n are distinct.

We begin proving Theorem 1 by stating the following useful Lemma. For all practical purposes, this reduces checking decodability and T-security to checking polynomial conditions. We say a matrix has the *MDS property* if every maximal minor has non-zero determinant. Equivalently, the matrix is the generator matrix of an MDS code.

Lemma 2: Let $\mathsf{PC}(K, L, T, \alpha, \beta)$ be a polynomial code, such that $\alpha \oplus \beta$ is decodable and T-secure. Suppose that there is an evaluation vector $\mathbf{a} = (a_1, \ldots, a_N) \in \mathbb{F}_{q^r}^N$ such that the following properties hold:

(i) (Decodability) The Generalized Vandermonde Matrix $GV(\mathbf{a}, \mathcal{J})$ is invertible.
(ii) (T-privacy) The $T \times N$ matrices

$$P = \left[a_n^{\alpha_K+t}\right] \quad \text{and} \quad Q = \left[a_n^{\beta_L+t}\right],$$

where $1 \le t \le T$ and $1 \le n \le N$, have the MDS property.

Then $\mathsf{PC}(K, L, T, \alpha, \beta)$ is decodable and T-secure.

Proof: Since the matrix $GV(\mathbf{a}, \mathcal{J})$ is invertible, the polynomial $h(x) = \sum_{j\in\mathcal{J}} C_j x^j$ can be interpolated from the evaluations $h(a_n)$, for $n = 1, \ldots, N$. Thus the user can recover all of the coefficients of $h(x)$. By the decodability condition of the outer sum $\alpha \oplus \beta$, the user can then recover all products $A_k B_\ell$.

The argument for T-privacy is familiar and follows the proof of T-security in Equation (28) in the proof of Theorem 2 in [1]. One shows that, given the above condition, any T-tuple of matrices $f(a_{n_1}), \ldots, f(a_{n_T})$ is uniform random on the space of all T-tuples of matrices of the appropriate size, and is independent of A. The same argument works for B. ∎

Let us now finish the proof of Theorem 1. Let $\mathbf{X} = (X_1, \ldots, X_N)$ be a vector of variables and consider the polynomial

$$D(\mathbf{X}) = \det(GV(\mathbf{X}, \mathcal{J})). \tag{27}$$

Additionally, if $\mathcal{T} = \{n_1, \ldots, n_T\} \subseteq [N]$ is any set of size T, define

$$P_{\mathcal{T}}(\mathbf{X}) = \det\left[X_{n_t}^{\alpha_K+t}\right] \quad \text{and} \quad Q_{\mathcal{T}}(\mathbf{X}) = \det\left[X_{n_t}^{\beta_L+t}\right]. \tag{28}$$

By Lemma 2, it suffices to find an evaluation vector $\mathbf{a} \in \mathbb{F}_{q^r}^N$ such that $D(\mathbf{a}) \neq 0$, $P_{\mathcal{T}}(\mathbf{a}) \neq 0$, and $Q_{\mathcal{T}}(\mathbf{a}) \neq 0$ for all $\mathcal{T} \subseteq [N]$ of size T. By the assumption that $\alpha \oplus \beta$ is decodable and T-secure, none of the polynomials $D(\mathbf{X})$, $P_{\mathcal{T}}(\mathbf{X})$, and $Q_{\mathcal{T}}(\mathbf{X})$ are zero, and all have degree bounded by $J := \sum_{j\in\mathcal{J}} j$.

Now consider a finite extension $\mathbb{F}_{q^r}$ of $\mathbb{F}_q$ and a subset $G \subseteq \mathbb{F}_{q^r}$ of size $G > \left(2\binom{N}{T} + 1\right) J$. Sample each entry a_n of $\mathbf{a}$ uniformly at random from G. Let E be the union of the events $D(\mathbf{a}) = 0$, $P_{\mathcal{T}}(\mathbf{a}) = 0$, and $Q_{\mathcal{T}}(\mathbf{a}) = 0$ for all $\mathcal{T} \subseteq [N]$ of size T. To finish the proof, it suffices to show that $\Pr(\mathsf{E}) < 1$. By the union bound and the Schwarz-Zippel Lemma, we have

$$\Pr(\mathsf{E}) \le \Pr(D(\mathbf{a}) = 0) + \sum_{\substack{\mathcal{T}\subseteq[N]\\|\mathcal{T}|=T}} \Pr(P_{\mathcal{T}}(\mathbf{a}) = 0) + \sum_{\substack{\mathcal{T}\subseteq[N]\\|\mathcal{T}|=T}} \Pr(Q_{\mathcal{T}}(\mathbf{a}) = 0) \le \left(2\binom{N}{T} + 1\right)\frac{J}{G} < 1.$$

This completes the proof of the Theorem. ∎

B. Proof of Theorem 6.

The degree table, $\alpha \oplus \beta$, is is shown in Table IV. We first prove for the case where $L \le K$. As in section V-B, we let UL, UR, LL, and LR be the upper-left, upper-right, lower-left, and lower-right blocks, respectively, of $\alpha \oplus \beta$. We first count the number in each block, and then study the intersections of the blocks.

It will be convenient to adopt the following notation. For integers A, B, and C, let $[A : B \mid C]$ be the set of all multiples of C in the interval $[A, B]$. If $A = DC$ is a multiple of C, we have

$$|[DC : B \mid C]| = \left(\left\lfloor \frac{B}{C} \right\rfloor - D + 1\right)^+$$

where $x^+ = \max\{x, 0\}$. If $C = 1$ then we write $[A : B]$ instead of $[A : B \mid 1]$, so that $[A : B]$ denotes all the integers in the interval $[A, B]$.

The sets $\mathrm{t(UL)}$, $\mathrm{t(UR)}$, $\mathrm{t(LL)}$, and $\mathrm{t(LR)}$ are given by

$$\begin{aligned}
\mathrm{t(UL)} &= [0 : KL - 1] \\
\mathrm{t(UR)} &= [KL : KL + K + T - 2] \\
\mathrm{t(LL)} &= [KL : 2KL + K(T-2) \mid K] \\
\mathrm{t(LR)} &= \bigcup_{t=1}^{T} [2KL + K(t-1) : 2KL + K(t-1) + T - 1]
\end{aligned} \tag{29}$$

From these expressions, one can count the sizes of the above sets to be

$$\begin{aligned}
|\,\mathrm{t(UL)}| &= KL \\
|\,\mathrm{t(UR)}| &= K + T - 1 \\
|\,\mathrm{t(LL)}| &= L + T - 1 \\
|\,\mathrm{t(LR)}| &= \begin{cases} T^2 & \text{if } T < K \\ KT - K + T & \text{if } T \ge K \end{cases}
\end{aligned} \tag{30}$$

To understand the last expression above, note that the intervals in the union expression for $\mathrm{t(LR)}$ consist of all integers from $2KL$ to $2KL + (K+1)(T-1)$ exactly when $T \ge K$.

As for intersections, clearly $\mathrm{t(UL)}$ intersects none of the sets of terms from the other blocks. Thus it suffices to understand the pairwise intersections among the other three blocks, and the triple intersection of the other three blocks. Two of these pairwise intersections and their sizes are easily understood:

$$\begin{aligned}
|\,\mathrm{t(UR)} \cap \mathrm{t(LL)}| &= [KL : KL + K + T - 2 \mid K], \\
|\,\mathrm{t(UR)} \cap \mathrm{t(LL)}| &= \left\lfloor \frac{T-2}{K} \right\rfloor + 2, \\
|\,\mathrm{t(LL)} \cap \mathrm{t(LR)}| &= [2KL : 2KL + K(T-2) \mid K], \\
|\,\mathrm{t(LL)} \cap \mathrm{t(LR)}| &= T - 1
\end{aligned} \tag{31}$$

Understanding the intersection $\mathrm{t(UR)} \cap \mathrm{t(LR)}$ is a bit more subtle, and we break the problem into two cases. If $T \ge K$, then $\mathrm{t(LR)} = [2KL : 2KL + (K+1)(T-1)]$. Since $\mathrm{t(UR)} = [KL : KL + K + T - 2]$, we see that $\mathrm{t(UR)} \cap \mathrm{t(LR)} = [2KL : KL + K + T - 2]$. In the case $T < K$ we have $2KL > KL + K + T - 2$ and thus the intersection is empty, unless $L = 1$, in which case $\mathrm{t(UR)} \cap \mathrm{t(LR)} = [2K : 2K + T - 2]$. It follows that

$$|\,\mathrm{t(UR)} \cap \mathrm{t(LR)}| = \begin{cases} T - 1 & \text{if } T < K, L = 1 \\ 0 & \text{if } T < K, L \ge 2 \\ (T - (K(L-1)+1))^+ & \text{if } T \ge K \end{cases} \tag{32}$$

It remains to count the size of the triple intersection. First suppose that $T < K$, which we break into two subcases: (i) $L = 1$ and (ii) $L \geq 2$. If $L = 1$ and $T = 1$, then the triple intersection is empty, but if $T > 1$ then all three blocks intersect in the lone terms $2K$. If $L \geq 2$, then the triple intersection is again empty by the above paragraph. Now suppose that $T \geq K$. In this case the intersection is the set $[2KL : KL + K + T - 2 \mid K]$, which has size $\left(\left\lfloor \frac{T-2}{K} \right\rfloor - L + 2\right)^+$. We therefore have

$$|\cap| = \begin{cases} 0 & \text{if } L = 1, 1 = T < K \\ 1 & \text{if } L = 1, 2 \leq T < K \\ 0 & \text{if } L \geq 2, T < K \\ \left(\left\lfloor \frac{T-2}{K} \right\rfloor - L + 2\right)^+ & \text{if } T \geq K \end{cases} \tag{33}$$

where $\cap = \mathrm{t}(\mathrm{UR}) \cap \mathrm{t}(\mathrm{LR}) \cap \mathrm{t}(\mathrm{LL})$.

We can now compute $N = |\mathrm{t}(\alpha \oplus \beta)|$ by using the inclusion-exclusion principle, as

$$\begin{aligned} N = |\mathrm{t}(\alpha \oplus \beta)| = {} & |\mathrm{t}(\mathrm{UL})| + |\mathrm{t}(\mathrm{UR})| + |\mathrm{t}(\mathrm{LL})| + |\mathrm{t}(\mathrm{LR})| \\ & - |\mathrm{t}(\mathrm{UR}) \cap \mathrm{t}(\mathrm{LL})| - |\mathrm{t}(\mathrm{LL}) \cap \mathrm{t}(\mathrm{LR})| \\ & - |\mathrm{t}(\mathrm{UR}) \cap \mathrm{t}(\mathrm{LR})| \\ & + |\mathrm{t}(\mathrm{UR}) \cap \mathrm{t}(\mathrm{LR}) \cap \mathrm{t}(\mathrm{LL})|. \end{aligned}$$

The above computation is straightforward given that we have already calculated the sizes of each of the individual sets. The only subtlety in deriving the formula (21) for N arises in the case that $L \geq 2$ and $K(L-1)+1 \leq T$. In this case, one uses the fact that

$$T - K(L-1) + 1 \geq 0 \Leftrightarrow \left\lfloor \frac{T-2}{K} \right\rfloor - L + 2 \geq 0.$$

From this equivalence and equations (32) and (33) one can use inclusion-exclusion to compute the value of N.

For $K < L$, the proof is analogous by interchanging α and β.

This completes the proof of the Theorem.

C. Note on the Communication Rate

The results in this paper were presented in terms of the download rate, not accounting for the upload rate and, therefore, the total communication rate of the scheme. This was done since the previous literature on this subject, [1], [2], and [5], all used the download rate as their measure of performance. We will now see that both the download rate and the upload rate for polynomial codes both depend on the number, N, of servers.

Let $A \in \mathbb{F}_q^{r \times s}$ and $B \in \mathbb{F}_q^{s \times t}$. As in (1), partition them as follows:

$$A = \begin{bmatrix} A_1 \\ \vdots \\ A_K \end{bmatrix}, \quad B = \begin{bmatrix} B_1 & \cdots & B_L \end{bmatrix},$$

so that

$$AB = \begin{bmatrix} A_1B_1 & \cdots & A_1B_L \\ \vdots & \ddots & \vdots \\ A_KB_1 & \cdots & A_KB_L \end{bmatrix}.$$

Thus, each $A_i \in \mathbb{F}_q^{\frac{r}{K} \times s}$ and each $B_i \in \mathbb{F}_q^{s \times \frac{t}{L}}$. The random matrices will also belong, respectively, to these spaces.

Using a polynomial code a user will send a linear combination of the A's and R's and another one of the B's and S's to each server, requiring an upload of $rs/K + st/L$ symbols per server, for a total upload cost of $N(rs/K + st/L)$ symbols. Each server will then multiply the two matrices they received and send the user a matrix of dimensions $r/K \times t/L$, for a total download cost of Nrt/KL. Thus, under our framework, minimizing the download, upload, or total communication costs are all equivalent to minimizing the number of servers, N.

A more thorough analysis on the communication and computational costs in SDMM can be found in [13].

Let us conclude by briefly discussing the difference in total communication cost between GASP and the scheme of [2]. As we saw in Section VII-B, for fixed N and T satisfying $T > N/6$ the download rates of these schemes are the same. The scheme of [2] achieves this rate by setting $L = 1$, while GASP achieves this rate by calculating the optimal value of KL, and choosing any values of K and L that yield this product. For fixed values of N, T, and KL, minimizing the communication cost is equivalent to minimizing the upload cost $N(rs/K + st/L)$. This is accomplished by choosing K and L to be as close to each other as possible, which GASP allows for. In contrast, the scheme of [2] which sets $L = 1$ ends up maximizing the upload cost subject to the given conditions. For example, when $N = 20$ and $T = 6$, the scheme of [2] sets $K = 4$ and $L = 1$, while GASP sets $K = L = 2$. This results in a 20% decrease in upload cost when $r = s = t$.

ACKNOWLEDGMENT

The authors kindly thank the authors of [2] for pointing out errors in the plots of Fig. 5 in the original version of this paper. These plots have since been corrected. Part of this work was completed while D. Karpuk was visiting the research group of S. El Rouayheb at Rutgers University. R. G. L. D'Oliveira is currently with the Massachusetts Institute of Technology. D. Karpuk is currently with F-Secure Corporation, Helsinki, Finland.

REFERENCES

[1] W.-T. Chang and R. Tandon, "On the capacity of secure distributed matrix multiplication," 2018, *arXiv:1806.00469*. [Online]. Available: http://arxiv.org/abs/1806.00469

[2] J. Kakar, S. Ebadifar, and A. Sezgin, "Rate-efficiency and straggler-robustness through partition in distributed two-sided secure matrix computation," 2018, *arXiv:1810.13006*. [Online]. Available: http://arxiv.org/abs/1810.13006

[3] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[4] R. Bitar, P. Parag, and S. E. Rouayheb, "Minimizing latency for secure distributed computing," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2017, pp. 2900–2904.

[5] H. Yang and J. Lee, "Secure distributed computing with straggling servers using polynomial codes," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 1, pp. 141–150, Jan. 2019.

[6] Q. Yu, S. Li, N. Raviv, S. M. M. Kalan, M. Soltanolkotabi, and S. Avestimehr, "Lagrange coded computing: Optimal design for resiliency, security and privacy," 2018, *arXiv:1806.00939*. [Online]. Available: http://arxiv.org/abs/1806.00939

[7] N. Raviv and D. A. Karpuk, "Private polynomial computation from Lagrange encoding," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 553–563, Jul. 2019.

[8] Q. Yu, M. Ali Maddah-Ali, and A. Salman Avestimehr, "Polynomial codes: An optimal design for high-dimensional coded matrix multiplication," 2017, *arXiv:1705.10464*. [Online]. Available: http://arxiv.org/abs/1705.10464
[9] Q. Yu, M. Ali Maddah-Ali, and A. Salman Avestimehr, "Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding," 2018, *arXiv:1801.07487*. [Online]. Available: http://arxiv.org/abs/1801.07487
[10] S. Dutta, M. Fahim, F. Haddadpour, H. Jeong, V. Cadambe, and P. Grover, "On the optimal recovery threshold of coded matrix multiplication," 2018, *arXiv:1801.10292*. [Online]. Available: http://arxiv.org/abs/1801.10292
[11] U. Sheth *et al.*, "An application of storage-optimal MatDot codes for coded matrix multiplication: Fast k-Nearest neighbors estimation," 2018, *arXiv:1811.11811*. [Online]. Available: http://arxiv.org/abs/1811.11811
[12] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, "A fundamental tradeoff between computation and communication in distributed computing," *IEEE Trans. Inf. Theory*, vol. 64, no. 1, pp. 109–128, Jan. 2018.
[13] R. G. L. D'Oliveira, S. El Rouayheb, D. Heinlein, and D. Karpuk, "Notes on communication and computation in secure distributed matrix multiplication," 2020, *arXiv:2001.05568*. [Online]. Available: http://arxiv.org/abs/2001.05568

Rafael G. L. D'Oliveira received the B.A. and M.S. degrees in mathematics and the Ph.D. degree in applied mathematics from the University of Campinas, Brazil, in 2009, 2012, and 2017, respectively. He was a Post-Doctoral Research Associate with Rutgers University from 2018 to 2019 and the Illinois Institute of Technology in 2017, and did a research internship at Telecom Paristech from 2015 to 2016. He is currently a Post-Doctoral Research Associate with the Massachusetts Institute of Technology. His research interests include discrete mathematics and coding theory.

Salim El Rouayheb (Member, IEEE) received the Diploma degree in electrical engineering from the Faculty of Engineering, Lebanese University, Roumieh, Lebanon, in 2002, the M.S. degree from the American University of Beirut, Lebanon, in 2004, and the Ph.D. degree in electrical engineering from Texas A&M University, College Station, in 2009. He was a Post-Doctoral Research Fellow with the University of California at Berkeley from 2010 to 2011 and a Research Scholar with Princeton University from 2012 to 2013. He is currently an Assistant Professor with the ECE Department, Rutgers University, NJ, USA. His research interests are in information theory and coding theory with a focus on applications to data security and privacy. He was a recipient of the NSF Career Award and the Google Faculty Research Award.

David Karpuk received the B.A. degree in mathematics from Boston College in 2006 and the Ph.D. degree in mathematics from The University of Maryland, College Park, in 2012. He was a Post-Doctoral Researcher with the Department of Mathematics and Systems Analysis, Aalto University, Helsinki, Finland, from 2012 to 2017, and an Assistant Professor with the Department of Mathematics, Universidad de los Andes, Bogotá, Colombia, from 2017 to 2019. He is currently a Senior Data Scientist with F-Secure Corporation, Helsinki, where his research interests include applications of machine learning to cyber security. He was a recipient of Post-Doctoral Researcher grants from the Academy of Finland and the Magnus Ehrnrooth Foundation.