Secure Private and Adaptive Matrix Multiplication Beyond the Singleton Bound

Christoph Hofmeister, Rawad Bitar, Marvin Xhemrishi and Antonia Wachter-Zeh
Institute for Communications Engineering, Technical University of Munich, Munich, Germany
{christoph.hofmeister, rawad.bitar, marvin.xhemrishi, antonia.wachter-zeh}@tum.de

Abstract

Consider the problem of designing secure and private codes for distributed matrix-matrix multiplication. A master server owns two private matrices $\mathbf{A}$ and $\mathbf{B}$ and hires worker nodes to help computing their multiplication. The matrices should remain information-theoretically private from the workers. Some of the workers are malicious and return corrupted results to the master. This work is motivated by the literature on myopic adversaries in network coding and distributed storage. Security beyond the Singleton bound is possible when the adversary has limited knowledge about the master's data and probabilistic decoding is acceptable. The key observation in this setting is that the master is the sender and the receiver. Therefore, the master enjoys a plethora of advantages that enable coding for security beyond the Singleton bound.

We design a framework for security against malicious adversaries in private matrix-matrix multiplication. Our main goal is to apply this security framework to schemes with adaptive rates previously introduced by a subset of the authors. Adaptive schemes divide the workers into clusters and thus provide flexibility in trading decoding complexity for efficiency. Checking the integrity of the computation per cluster has low complexity but costs deleting the results of a whole cluster with at least one malicious worker. Checking the integrity of the results per worker is more complex but allows an efficient use of the non-malicious workers. Our scheme, called SRPM3, provides a computationally efficient security check that detects malicious workers with high probability and can tolerate the presence of an arbitrary number of malicious workers. We provide simulation results that validate our theoretical findings.

Index Terms

Secure private rateless codes, double-sided private matrix multiplication, partial stragglers, information-theoretic privacy, security beyond the Singleton bound

I. Introduction

Motivated by distributed machine learning, we consider the core computation of matrix-matrix multiplication. Due to the tremendous amount of data being collected and processed, computing the matrix multiplication locally is becoming a bottleneck. Distributed computing emerged as a solution to alleviate the computation bottleneck. A master node possessing the matrices splits them into smaller chunks sent to worker nodes. The workers compute the smaller matrix multiplications in parallel and return the results to the master. The master aggregates the results received from the workers to obtain the initial desired computation. The main challenges of distributed computing, which we focus on, are: stragglers, privacy and security.

It is known that in distributed computing some workers, referred to as *stragglers*, will take much longer than others to return their results [1], [2]. Hence, waiting for all workers can outweigh the benefits of distributing the computation, e.g., [3]–[5]. Given the sensitivity of the data used for machine learning algorithms, e.g., genomes and medical data, privacy against potential eavesdroppers is a must. We are interested in information-theoretic privacy. The eavesdropper compromising a subset of the workers is assumed to have unbounded computational power. However, the assumption of information-theoretic privacy is a limit on the number of workers that can be compromised by the eavesdropper. In addition to stragglers and the eavesdropper, we consider the setting in which a malicious adversary compromises a subset of the workers. Those compromised/malicious workers send corrupted computation to the master in order to corrupt the whole computation process. When coding is used, for example to mitigate stragglers or to guarantee privacy, one malicious worker can corrupt the whole computation if care is not taken.

We are interested in applications where the workers have limited and different computation power and different communication bandwidth that can change with time. We use the term *resources* to refer to the computation power and communication bandwidth of the workers. Examples of such applications include edge computing and internet of things (IoT) networks in which small devices (such as sensors, phones and tablets) collaborate to run intensive computations. In such applications, devices can join and leave the network at any time and many factors such as battery level and geographic location dynamically affect the resources (and the performance) of the workers. We refer to this setting as heterogeneous and time-varying.

The goal of this paper is to propose a framework that can add a layer of security to existing private and straggler tolerant matrix multiplication schemes. In particular, we are interested in schemes that can adapt to heterogeneous environments.

This work was partly supported by the Technical University of Munich - Institute for Advanced Studies, funded by the German Excellence Initiative and European Union Seventh Framework Programme under Grant Agreement No. 291763.

Related work: Coding for straggler mitigation started in [5] where the authors proposed the use of MDS codes to mitigate stragglers in distributed linear computations, i.e., matrix-vector multiplication. Since then, the use of coding theory for distributed computing, referred to as coded computing, has witnessed a significant attention from the scientific community. Using codes provides a fast, private and reliable distributed computing. For example schemes improving on the work of [5], different settings and fundamental limits on coded computing can be found in [6]–[24]. For a relatively recent survey on coded computing we refer the reader to [25]. Information-theoretic privacy and straggler mitigation in coded computing (for polynomial evaluation, matrix-matrix and matrix-vector multiplication) is achieved by using secret sharing [26]–[39]. For a very recent survey on private distributed computing and its connections we refer the reader to [40]. Security against malicious workers is considered in [28], [41]–[44]. The works of [41] and [28] design codes in which each malicious worker can inflict a double damage. In other words, if n is the total number of workers and r is the number of malicious workers to be tolerated, the master can use at most $n-2r$ workers for legitimate computations. This idea holds true through the Singleton-bound for distributed storage, network coding and MDS codes. It has been shown for network coding and distributed storage [45]–[49] that when the malicious workers have limited knowledge about the master's data, the master can with high probability half the damage of the malicious workers and use $n-r$ workers for legitimate purposes. The advantage of distributed computing is that the master can be seen as the sender and the receiver. The master can thus use this fact to alleviate the assumption of limited knowledge of the malicious workers. In [43] and [44] variants of Reed-Solomon codes are used to half the damage of the malicious workers when the workers introduce random noise and when the workers introduce any kind of noise, respectively. The disadvantage of [44] is the high computational complexity incurred by the master. Concurrently and independently of our work, the authors of [50] present a scheme to tolerate malicious workers in coded computing with low computational complexity. The ideas used in [50] are similar to the ideas used in this work. However, [28], [41], [43], [44] and [50] consider security in settings where the workers are assumed to have similar resources. In addition, a maximum amount of stragglers is assumed. In this work we are interested in coded computing for heterogeneous environments where the number of stragglers can change over time. Our scheme enjoys a smaller computational complexity than that of [50] which can be adaptively increased to increase the efficiency of the scheme as will be explained in the sequel. In [42] secure matrix-vector multiplication in heterogeneous environments is ensured by using homomorphic hash functions. The master detects the malicious workers with high probability and removes them from the system. Our goal is to construct codes that allow privacy, security and adaptive straggler tolerance for matrix-matrix multiplication in heterogeneous environments. We build on the scheme introduced in [32] to achieve our goal. It is worth mentioning that the schemes presented in this work and in [42]–[44] allow the master to detect the presence of malicious workers and remove their corrupted computation with arbitrarily high probability.

Contributions: We introduce SRPM3, secure rateless and private matrix-matrix multiplication, scheme. SRPM3 allows the master to offload a matrix-matrix multiplication to workers that are malicious, curious (eavesdropper) and have different time-varying resources. In contrast to most coding theoretic frameworks, SRPM3 can tolerate the presence of an arbitrary number of malicious workers and still detect corruption of computation efficiently and with high probability. SRPM3 is based on RPM3, rateless and private matrix-matrix multiplication, scheme introduced by a subset of the authors [32]. Similarly to RPM3, SRPM3 works in rounds. In each round, the master divides the workers into clusters of workers that have similar resources, i.e., similar service time. The additional component of SRPM3 is a computation efficient verification of matrix-matrix multiplication based on Freivald's algorithm [51]. The verification of the computation is done per cluster. The master can thus verify with high probability whether the computation returned from each cluster is corrupted. Verifying the computation per clusters results in a more efficient verification than the scheme presented in [50]. As an extra layer of verification, if the computation of a certain cluster of workers is corrupted, the master can run Freivald's algorithm on the computation of each worker to detect, with high probability, the malicious workers and use the results of the honest workers.

Organization: In Section II we set the notation and define the model. We provide the details of our SRPM3 scheme in Section III. In Section IV, we prove that SRPM3 can detect the presence of malicious workers with high probability. We prove that SRPM3 maintains privacy of the input matrices in Section V. In Section VI we provide simulation results that validate our theoretical findings. We conclude the paper in Section VII.

II. Preliminaries

A. Notation

For any positive integer a we define $[a] \triangleq \{1, \dots, a\}$. We denote by n the total number of workers. For $i \in [n]$ we denote worker i by w_i. For a prime power q, we denote by $\mathbb{F}_q$ the finite field of size q. We denote vectors by bold letters, e.g., $\mathbf{a}$ and matrices by bold capital letters, e.g., $\mathbf{A}$. Random variables are denoted by typewriter characters, e.g., $\mathtt{A}$. Calligraphic letters are used for sets, e.g., $\mathcal{A}$. We denote by $H(\mathtt{A})$ the entropy of the random variable $\mathtt{A}$ and by $I(\mathtt{A};\mathtt{B})$ the mutual information between two random variables $\mathtt{A}$ and $\mathtt{B}$. All logarithms are to the base q.

B. Problem setting

The master nodes wants to multiply two private matrices $\mathbf{A} \in \mathbb{F}_q^{r\times s}$ and $\mathbf{B} \in \mathbb{F}_q^{s\times \ell}$ to obtain $\mathbf{C} = \mathbf{AB} \in \mathbb{F}_q^{r\times \ell}$. The matrices $\mathbf{A}$ and $\mathbf{B}$ are assumed to be uniformly distributed over their respective fields. To alleviate the computation complexity, the master offloads the computation to n workers. We consider an untrusted heterogeneous and time-varying environment in which the workers satisfy the following properties: *1)* The performance of the workers is different. The workers can be grouped into $c > 1$ clusters of workers that have similar performance (service time). We denote by n_u, $u = 1, \ldots, c$, the number of workers in cluster u and require that $\sum_{u \in [c]} n_u = n$.

2) The performance of the workers can change with time. Therefore, the clustering can also change throughout the multiplication of $\mathbf{A}$ and $\mathbf{B}$.

3) The workers have small memory and limited computational capacity.

4) An eavesdropper can compromise up to z, $1 \leq z < \min\limits_{u \in [c]} n_u$, workers to obtain information about $\mathbf{A}$ and/or $\mathbf{B}$. We say that z workers are colluding. If $z = 1$, we say the workers do not collude. The eavesdropper observing the data sent to the compromised workers is assumed to have unbounded computation power to crack cryptographic algorithms.

5) Workers are malicious. An arbitrary subset of the workers may collaboratively send noisy computation to the master to corrupt the whole computation process. Despite having arbitrary number of malicious workers collaborating to jam the computation, we assume that at most up to z of those malicious workers share information about $\mathbf{A}$ and $\mathbf{B}$ with each other, i.e., are compromised by an eavesdropper. This model is motivated by different malicious parties having interest in not allowing the master to successfully compute the matrix multiplication. However, those parties are themselves competing in learning information about the private matrices for their own benefit.

Remark 1. *The assumption that malicious workers do not share information is important for our scheme. Some entries must remain private from the workers to tolerate an arbitrary number of malicious workers . If those private entries are revealed to the workers, then our scheme can be modified to still guarantee security at the expense of an increase in computation complexity at the master as detailed in Remark 2.*

$\mathbf{A}$ is divided into m equally sized blocks of rows and $\mathbf{B}$ is divided into k equally sized blocks of columns[1] such that

$$\begin{bmatrix} \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_m \end{bmatrix} \begin{bmatrix} \mathbf{B}_1 & \ldots & \mathbf{B}_k \end{bmatrix} = \begin{bmatrix} \mathbf{C}_{1,1} & \ldots & \mathbf{C}_{1,k} \\ \vdots & \ddots & \vdots \\ \mathbf{C}_{m,1} & \ldots & \mathbf{C}_{m,k} \end{bmatrix},$$

such that $\mathbf{C}_{i,j} = \mathbf{A}_i \mathbf{B}_j$, $i \in [m], j \in [k]$. The master encodes and sends tasks to the workers until it can decode $\mathbf{C}$ based on the responses. Each task is of equivalent computational cost as computing one of the sub-matrices $\mathbf{C}_{i,j}$. The number of responses necessary for decoding depends on the rate of the scheme.

A scheme guarantees double-sided z-privacy in an information-theoretic sense if any collection of z colluding workers learns nothing about the input matrices $\mathbf{A}$ and $\mathbf{B}$. Such a scheme is said to be double-sided z-private as defined next. We introduce some notation first. Let $\mathtt{A}$ and $\mathtt{B}$ be the random variables representing $\mathbf{A}$ and $\mathbf{B}$. The set of random variables representing the collection of tasks assigned to worker w_i, $i = 1, \ldots, n$ is denoted by $\mathcal{W}_i$. For a set $\mathcal{A} \subseteq [n]$ we define $\mathcal{W}_\mathcal{A}$ as the set of random variables representing all tasks assigned to the workers indexed by $\mathcal{A}$, i.e., $\mathcal{W}_\mathcal{A} = \{\mathcal{W}_i | i \in \mathcal{A}\}$.

Definition 1 (Double-sided z-private scheme, [32])**.** *A scheme is said to be double-sided z-private if the following privacy constraint holds*

$$I\left(\mathtt{A}, \mathtt{B}; \mathcal{W}_\mathcal{Z}\right) = 0, \forall \mathcal{Z} \subset [n], \text{ s.t. } |\mathcal{Z}| = z. \tag{1}$$

Definition 2 (Probabilistic decoding)**.** *In this work we relax the decoding constraint from deterministic decoding to probabilistic decoding. Let $\mathcal{R}_i$ be the set of random variable representing all the computational results of w_i received at the master. Let $\mathtt{C}$ be the random variable representing the matrix $\mathbf{C}$. The decodability constraint can be expressed as*

$$H\left(\mathtt{C} | \mathcal{R}_1, \ldots, \mathcal{R}_n\right) < \varepsilon, \tag{2}$$

where ε is an arbitrarily small positive number. Deterministic decoding requires $H\left(\mathtt{C} | \mathcal{R}_1, \ldots, \mathcal{R}_n\right) = 0$.

Note that the sets $\mathcal{R}_i$ can be of different cardinality, and some may be empty, reflecting the heterogeneity of the system and the straggler tolerance.

III. SRPM3 Scheme

We introduce SRPM3: Secure, Rateless and Private Matrix-Matrix Multiplication. SRPM3 is based in large parts on the RPM3 scheme presented in [32] by a subset of the authors. We explain next the common aspects of RPM3 and SRPM3. The details of the additional part on the adversarial error detection are elaborated in the next section.

The matrix multiplication process is divided into rounds and for every round the workers are grouped into clusters. All workers start in round $t = 1$. Every time a cluster of workers completes a task, it is advanced one round.

[1]It is assumed, that $m|r$ and $k|l$. These conditions can always be fulfilled by padding $\mathbf{A}$ with up to $m-1$ rows and/or $\mathbf{B}$ with up to $k-1$ columns.

A. Encoding

A factored fountain code [19] encodes $\{\mathbf{A}_1, \dots, \mathbf{A}_m\}$ into $\{\widetilde{\mathbf{A}}_1, \widetilde{\mathbf{A}}_2, \dots\}$ and $\{\mathbf{B}_1, \dots, \mathbf{B}_k\}$ into $\{\widetilde{\mathbf{B}}_1, \widetilde{\mathbf{B}}_2, \dots\}$. Using factored fountain code ensures that $\widetilde{\mathbf{C}}_1 \triangleq \widetilde{\mathbf{A}}_1\widetilde{\mathbf{B}}_1, \widetilde{\mathbf{C}}_2 \triangleq \widetilde{\mathbf{A}}_2\widetilde{\mathbf{B}}_2, \dots$ are the symbols of a fountain code encoding $\{\mathbf{C}_{i,j} | i \in [m], j \in [k]\}$.

The maximum number of fountain coded sub-matrices of $\mathbf{C}$ that cluster u, consisting of n_u workers, can compute in a round given by

$$d_u \leq \begin{cases} \lfloor \frac{n_1 - 2z + 1}{2} \rfloor & \text{for } u = 1 \\ \lfloor \frac{n_u - z + 1}{2} \rfloor & \text{otherwise.} \end{cases} \tag{3}$$

Choosing d_u smaller than the upper bound increases the straggler tolerance of the cluster. Specifically, decreasing d_u by two increases the straggler tolerance of cluster u by one. We define d_{max} the maximum d_u in a round, i.e., $d_{\text{max}} \triangleq \max_{u \in [c]} d_u$.

At the start of a round t the master does the following:

- chooses $d_{\text{max}} + z + n$ distinct elements $\{\alpha_{t,1}, \dots, \alpha_{t,d_u+z}, \beta_{t,1}, \dots, \beta_{t,n}\}$ of $\mathbb{F}_q$ uniformly at random[2], and
- draws $2z$ matrices $\{\mathbf{R}_{t,1}, \dots, \mathbf{R}_{t,z}\}$ and $\{\mathbf{S}_{t,1}, \dots, \mathbf{S}_{t,z}\}$ independently and uniformly at random from $\mathbb{F}_q^{r/m \times s}$ and $\mathbb{F}_q^{s \times l/k}$, respectively.

Then, for each cluster the master computes the fountain coded matrices $\{\widetilde{\mathbf{A}}_{t,1}^{(u)}, \dots, \widetilde{\mathbf{A}}_{t,d_u}^{(u)}\}$ and $\{\widetilde{\mathbf{B}}_{t,1}^{(u)}, \dots, \widetilde{\mathbf{B}}_{t,d_u}^{(u)}\}$ and fits the following two Lagrange polynomials of degree $d_u + z - 1$:

- $\mathbf{F}_t^{(u)}(x)$ to the set of points $\{(\alpha_{t,1}, \mathbf{R}_{t,1}), \dots, (\alpha_{t,z}, \mathbf{R}_{t,z}), (\alpha_{t,z+1}, \widetilde{\mathbf{A}}_{t,1}^{(u)}), \dots (\alpha_{t,z+d_u}, \widetilde{\mathbf{A}}_{t,d_u}^{(u)})\}$ and
- $\mathbf{G}_t^{(u)}(x)$ to the set of points $\{(\alpha_{t,1}, \mathbf{S}_{t,1}), \dots, (\alpha_{t,z}, \mathbf{S}_{t,z}), (\alpha_{t,z+1}, \widetilde{\mathbf{B}}_{t,1}^{(u)}), \dots (\alpha_{t,z+d_u}, \widetilde{\mathbf{B}}_{t,d_u}^{(u)})\}$.

Fitting the Lagrange polynomial $\mathbf{F}_t^{(u)}(x)$ and $\mathbf{G}_t^{(u)}(x)$ to the given set of points amounts to computing

$$\mathbf{F}_t^{(u)}(x) = \sum_{i=1}^{z} l_i(x)\mathbf{R}_{t,i} + \sum_{i=z+1}^{z+d_u} l_i(x)\widetilde{\mathbf{A}}_{t,i-z},$$

$$\mathbf{G}_t^{(u)}(x) = \sum_{i=1}^{z} l_i(x)\mathbf{S}_{t,i} + \sum_{i=z+1}^{z+d_u} l_i(x)\widetilde{\mathbf{B}}_{t,i-z}.$$

For every $i \in [z + du]$ the Lagrange basis polynomial $l_i(x) = \prod_{j \in [d_u+z] \setminus \{i\}} \frac{x - \alpha_{t,j}}{\alpha_{t,i} - \alpha_{t,j}}$ is 0 at $x = \alpha_i$ and 1 at $x = \alpha_j$ for all $j \in [z + d_u] \setminus \{i\}$. Consequently, for all $i = 1, \dots, z$, and $j = z+1, \dots, z+d_u$ it holds that $\mathbf{F}_t^{(u)}(\alpha_{t,i}) = \mathbf{R}_i$ and $\mathbf{F}_t^{(u)}(\alpha_{t,j}) = \widetilde{\mathbf{A}}_{y,j-z}$. The same holds for $\mathbf{G}_t^{(u)}(x)$. Define the polynomial $\mathbf{H}_t^{(u)}(x) \triangleq \mathbf{F}_t^{(u)}(x)\mathbf{G}_t^{(u)}(x)$; this polynomial evaluates to $\widetilde{\mathbf{C}}_{t,1} = \widetilde{\mathbf{A}}_{t,1}\widetilde{\mathbf{B}}_{t,1}, \dots, \widetilde{\mathbf{C}}_{t,d_u} = \widetilde{\mathbf{A}}_{t,d_u}\widetilde{\mathbf{B}}_{t,d_u}$ at $\alpha_{t,z+1}, \dots, \alpha_{t,z+d_u}$, respectively.

The master sends to worker w_i in cluster u $\mathbf{F}_t^{(u)}(\beta_{t,i})$ and $\mathbf{G}_t^{(u)}(\beta_{t,i})$ as a computational task. Each worker computes and returns $\mathbf{H}_t^{(u)}(\beta_{t,i}) = \mathbf{F}_t^{(u)}(\beta_{t,i})\mathbf{G}_t^{(u)}(\beta_{t,i})$.

B. Decoding

The degree of $\mathbf{H}_t^{(u)}(x)$ is $2d_u + 2z - 2$. Recall from (3) that for cluster $u = 1$, the number of workers n_1 satisfies $n_1 \geq 2d_1 + 2z - 1$. Hence, after obtaining $2d_1 + 2z - 1$ responses from cluster $u = 1$, the master can interpolate $\mathbf{H}_t^{(1)}(x)$. By design of $\mathbf{H}_t^{(u)}(x)$ it holds that $\mathbf{H}_t^{(1)}(\alpha_{t,i}) = \mathbf{H}_t^{(2)}(\alpha_{t,i}) = \dots = \mathbf{H}_t^{(c)}(x) = \mathbf{R}_{t,i}$ for all $i = 1, \dots, z$. Thus, the master needs only $2d_u + z - 1$ response from each cluster $u > 1$ to interpolate $\mathbf{H}_t^{(u)}(x)$. Evaluating $\mathbf{H}_t^{(u)}(x)$ at each of $\alpha_{t,1}, \dots, \alpha_{t,d_u}$ produces $\{\widetilde{\mathbf{C}}_{t,1}^{(u)}, \dots, \widetilde{\mathbf{C}}_{t,d_u}^{(u)}\}$.

At this point, the security check described in section IV is performed. If the security check passes, the master feeds the "correct" fountain coded matrices $\widetilde{\mathbf{C}}_{t,i}^{(u)}$ into a peeling decoder. The scheme is finished once the peeling decoder has decoded all sub-matrices of $\mathbf{C}$.

C. Clustering

In the first round, all workers are assigned to a single cluster. Over the course of the computation, the master measures the empirical response times (consisting of both communication and computation times) of the workers and updates the clustering such that workers with similar performance are assigned to the same cluster. For $t > 1$ the number of workers per cluster must satisfy $n_1 \geq 2z + 1$ and $n_u \geq z + 1$ for all other clusters; if at least one fountain coded sub-matrix of $\mathbf{C}$ shall be decoded from each cluster.

[2]Note that in the original RPM3 scheme $\alpha_1, \dots, \alpha_{d_{max}+z}, \beta_1, \dots, \beta_n$ are considered fixed and publicly known parameters of the scheme. In SRPM3, they are generated randomly for each round and are private to the master.

Algorithm 1: Freivald's algorithm used to efficiently verify the correctness of a single matrix-matrix multiplication with high probability. If the multiplication is correct, the algorithm always returns *True*. Otherwise, the algorithm returns *False* with high probability, cf., Proposition 1.

```
Input : $\mathbf{X}_1 \in \mathbb{F}_q^{r\times s}$, $\mathbf{X}_2 \in \mathbb{F}_q^{s\times l}$, $\mathbf{X}_3 \in \mathbb{F}_q^{r\times l}$
Result: True or False
$\boldsymbol{\nu} \leftarrow$ uniformly random vector from $\mathbb{F}_q^l$;
Compute $\boldsymbol{\nu}' \leftarrow \mathbf{X}_2\boldsymbol{\nu}$;
Compute $\mathbf{X}_1\boldsymbol{\nu}'$;
Compute $\mathbf{X}_3\boldsymbol{\nu}$;
if $\mathbf{X}_1\boldsymbol{\nu}' = \mathbf{X}_3\boldsymbol{\nu}$ then
    return True;
else
    return False;
end
```

IV. Adversarial Error Detection

A. Error Detection in Matrix-Matrix Multiplication

Freivalds' algorithm [51] is an efficient way to verify the correctness of a single matrix-matrix multiplication with high probability, cf., Algorithm 1. Freivald's algorithm is based on the fact that for $\mathbf{X}_1 \in \mathbb{F}_q^{r\times s}$, $\mathbf{X}_2 \in \mathbb{F}_q^{s\times l}$ and $\mathbf{X}_3 \in \mathbb{F}_q^{r\times l}$, if $\mathbf{X}_1\mathbf{X}_2 = \mathbf{X}_3$, then $\mathbf{X}_1\mathbf{X}_2\boldsymbol{\nu} = \mathbf{X}_3\boldsymbol{\nu}$ holds for all possible $\boldsymbol{\nu} \in \mathbb{F}_q^l$. Consequently if $\mathbf{X}_1\mathbf{X}_2 = \mathbf{X}_3$, then Algorithm 1 always returns *True*. On the other hand, if $\mathbf{X}_1\mathbf{X}_2 \neq \mathbf{X}_3$, the algorithm returns *False* with high probability (Proposition 1). Freivald's algorithm probabilistically verifies the matrix-matrix multiplication with only three matrix-vector multiplications: $\nu' \triangleq \mathbf{X}_2\boldsymbol{\nu}$, $\mathbf{X}_1\boldsymbol{\nu}'$ and $\mathbf{X}_3\boldsymbol{\nu}$. This is in contrast to re-computing $\mathbf{X}_1\mathbf{X}_2$ and deterministically verifying the computation.

Proposition 1. *The probability that Algorithm 1 returns* True *when applied to matrices* $\mathbf{X}_1 \in \mathbb{F}_q^{r\times s}$*,* $\mathbf{X}_2 \in \mathbb{F}_q^{s\times l}$ *and* $\mathbf{X}_3 \in \mathbb{F}_q^{r\times l}$ *with* $\mathbf{X}_1\mathbf{X}_2 \neq \mathbf{X}_3$ *is at most* $1/q$*.*

Proof. The relation $\mathbf{X}_1\mathbf{X}_2\boldsymbol{\nu} = \mathbf{X}_3\boldsymbol{\nu}$ is equivalent to $(\mathbf{X}_1\mathbf{X}_2 - \mathbf{X}_3)\boldsymbol{\nu} = \mathbf{0}$ and holds if and only if $\boldsymbol{\nu}$ is in the nullspace of $\mathbf{E} \triangleq \mathbf{X}_1\mathbf{X}_2 - \mathbf{X}_3$. Given that $\mathbf{X}_1\mathbf{X}_2 \neq \mathbf{X}_3$, the rank of $\mathbf{E}$ is at least 1. Hence, the nullspace of $\mathbf{E}$ has dimension at most $l-1$. There are q^l distinct vectors in $\mathbb{F}_q^l$, of which at most q^{l-1} are in the nullspace of $\mathbf{E}$. The probability that $\boldsymbol{\nu}$ lies in the nullspace of $\mathbf{E}$ when drawn uniformly at random from $\mathbb{F}_q^l$ is at most $q^{l-1}/q^l = 1/q$. ☐

B. Error Detection in Polynomial Multiplication

A polynomial multiplication can be verified similarly [51]. Consider three polynomials $p_1(x)$, $p_2(x)$ and $p_3(x)$ over a field $\mathbb{F}_q$ with $p_1(x)p_2(x) \neq p_3(x)$ and $\deg(p_1(x)p_2(x)) \geq \deg(p_3(x))$. For an evaluation point γ drawn uniformly at random from a subset $\mathcal{S} \subseteq \mathbb{F}_q$, the probability that $p_1(\gamma)p_2(\gamma) = p_3(\gamma)$ is at most $\frac{\deg(p_1(x)p_2(x))}{|\mathcal{S}|}$ by the Schwartz-Zippel lemma [52], [53].

C. Adversarial Error Detection in SRPM3

We combine the methods for the verification of matrix-matrix multiplications and polynomial multiplications to efficiently verify that the multiplication of polynomial matrices $\mathbf{H}_t^{(u)}(x) = \mathbf{F}_t^{(u)}(x)\mathbf{G}_t^{(u)}(x)$ is correct in every cluster $u \in [c]$. Theorem 1 summarizes the main result of adversarial error detection in SRPM3.

Theorem 1. *For every cluster* $u \in [c]$ *and for every round* t *of SRPM3, given the three polynomial matrices* $\mathbf{H}_t^{(u)}(x) \in \mathbb{F}_q^{r\times\ell}$*,* $\mathbf{F}_t^{(u)}(x) \in \mathbb{F}_q^{r\times s}$ *and* $\mathbf{G}_t^{(u)}(x) \in \mathbb{F}_q^{s\times\ell}$*, the correctness of the polynomial multiplication* $\mathbf{H}_t^{(u)}(x) = \mathbf{F}_t^{(u)}(x)\mathbf{G}_t^{(u)}(x)$ *is verifiable with high probability. The probability of* not *detecting an error is bounded from above as*

$$\Pr(\textit{not detecting an error}) \leq \frac{\deg(\mathbf{H}_t^{(u)}(x))}{q - \deg(\mathbf{H}_t^{(u)}(x)) - 1} + \frac{1}{q} \tag{4}$$

even if all n_u *workers collaborate the attack.*

The complexity of the verification is $\mathcal{O}(rs + s\ell + r\ell)$*; which is* $\mathcal{O}(r^2)$ *if* r*,* s *and* ℓ *scale together.*

Proof: The idea is to efficiently implement Algorithm 1. For clarity of presentation, we omit the subscript t and superscript (u) since the following holds for every cluster $u \in [c]$ and for each round t. We choose the evaluation point γ to be one of the α_i's, $i = z+1, \ldots, z+d_u$, say α_{z+1}. This results in a more efficient verification since no extra computational effort is necessary for computing $\mathbf{F}(\alpha_{z+11}) = \widetilde{\mathbf{A}}_1$ and $\mathbf{G}(\alpha_{z+1}) = \widetilde{\mathbf{B}}_1$ ($\mathbf{H}(\alpha_1) = \widetilde{\mathbf{C}}_1$ must be computed anyway since we want to feed it into the peeling decoder of the fountain code).

To prove the bound on the probability of error detection, we need the following intermediate result.

Claim 1. *As long as at most z workers collude, the values of $\alpha_1, \dots, \alpha_{z+d_u}, \beta_1, \dots, \beta_n$ are private in an information theoretic sense from the malicious workers.*

The proof of Claim 1 is given in Section V. The importance of Claim 1 stems from the observation that if the malicious workers know $\alpha_1, \dots, \alpha_{z+d_u}, \beta_1, \dots, \beta_n$, then any number of malicious workers larger than $n_u/2+1$ can introduce errors that Algorithm 1 cannot detect with probability 1.

Let $\texttt{Fv}(\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3)$ be the event that Algorithm 1 takes $\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3$ as input and returns *True*. We bound the probability of not detecting an error as follows

$$\begin{aligned}\Pr(\texttt{Fv}(\widetilde{\mathbf{A}}_1, \widetilde{\mathbf{B}}_1, \widetilde{\mathbf{C}}_1) | \mathbf{H}(x) \neq \mathbf{F}(x)\mathbf{G}(x)) &= \Pr(\texttt{Fv}(\widetilde{\mathbf{A}}_1, \widetilde{\mathbf{B}}_1, \widetilde{\mathbf{C}}_1) | \widetilde{\mathbf{A}}_1\widetilde{\mathbf{B}}_1 = \widetilde{\mathbf{C}}_1) P(\widetilde{\mathbf{A}}_1\widetilde{\mathbf{B}}_1 = \widetilde{\mathbf{C}}_1 | \mathbf{H}(x) \neq \mathbf{F}(x)\mathbf{G}(x)) \\ &\quad + \Pr(\texttt{Fv}(\widetilde{\mathbf{A}}_1, \widetilde{\mathbf{B}}_1, \widetilde{\mathbf{C}}_1) | \widetilde{\mathbf{A}}_1\widetilde{\mathbf{B}}_1 \neq \widetilde{\mathbf{C}}_1) P(\widetilde{\mathbf{A}}_1\widetilde{\mathbf{B}}_1 \neq \widetilde{\mathbf{C}}_1 | \mathbf{H}(x) \neq \mathbf{F}(x)\mathbf{G}(x)) \quad (5) \\ &\leq \Pr(\mathbf{F}(\alpha_{z+1})\mathbf{G}(\alpha_{z+1}) = \mathbf{H}(\alpha_{z+1}) | \mathbf{H}(x) \neq \mathbf{F}(x)\mathbf{G}(x)) + \frac{1}{q} \quad (6) \\ &\leq \frac{\deg(\mathbf{H}(x))}{q - \deg(\mathbf{H}(x)) - 1} + \frac{1}{q}. \quad (7)\end{aligned}$$

Equation (5) follows from the law of total probability. In Equation (6) we use the following

- $\Pr(\texttt{Fv}(\widetilde{\mathbf{A}}_1, \widetilde{\mathbf{B}}_1, \widetilde{\mathbf{C}}_1) | \widetilde{\mathbf{A}}_1\widetilde{\mathbf{B}}_1 = \widetilde{\mathbf{C}}_1) = 1$.
- $\Pr(\widetilde{\mathbf{A}}_1\widetilde{\mathbf{B}}_1 = \widetilde{\mathbf{C}}_1 | \mathbf{H}(x) \neq \mathbf{F}(x)\mathbf{G}(x)) = \Pr(\mathbf{F}(\alpha_{z+1})\mathbf{G}(\alpha_{z+1}) = \mathbf{H}(\alpha_{z+1}) | \mathbf{H}(x) \neq \mathbf{F}(x)\mathbf{G}(x))$.
- $\Pr(\texttt{Fv}(\widetilde{\mathbf{A}}_1, \widetilde{\mathbf{B}}_1, \widetilde{\mathbf{C}}_1) | \widetilde{\mathbf{A}}_1\widetilde{\mathbf{B}}_1 \neq \widetilde{\mathbf{C}}_1) \leq \frac{1}{q}$ (see Proposition 1).
- $\Pr(\widetilde{\mathbf{A}}_1\widetilde{\mathbf{B}}_1 \neq \widetilde{\mathbf{C}}_1 | \mathbf{H}(x) \neq \mathbf{F}(x)\mathbf{G}(x)) \leq 1$.

To obtain Equation (7) observe that given $\mathcal{B}$, the set of β_i's used to fit $\mathbf{H}(x)$, α_{z+1} is a random variable distributed uniformly over $\mathbb{F}_q \setminus \mathcal{B}$. Since from Claim 1 the value of α_i's and β_i's are not known to the malicious workers, then the discussion of Section IV-B holds. Therefore, we can write

$$\Pr(\mathbf{F}(\alpha_{z+1})\mathbf{G}(\alpha_{z+1}) = \mathbf{H}(\alpha_{z+1}) | \mathbf{H}(x) \neq \mathbf{F}(x)\mathbf{G}(x)) \leq \frac{\deg(\mathbf{H}(x))}{q - \deg \mathbf{H}(x) - 1}.$$

Computation complexity: Since no additional computations are needed for the evaluations, the computational cost of error detection comes down to running Algorithm 1 once per cluster. The computational cost of running Algorithm 1 once consists of $rs + sl + rl$ multiplications and $r(s-1) + s(l-1) + r(l-1)$ additions in $\mathbb{F}_q$. In addition, l random elements (the vector $\boldsymbol{\nu}$) need to be generated from $\mathbb{F}_q$ and up to r comparisons (to compare the results) need to be made. Thus, the complexity of running Algorithm 1 is $\mathcal{O}(rs + sl + rl)$. If r, s, and l scale together, the computational complexity is $\mathcal{O}(r^2)$.

For comparison, the cost of verifying the multiplication directly and deterministically by computing the matrix products is equal to $\deg(\mathbf{H}(x))rsl$ multiplications in $\mathbb{F}_q$ for the classic matrix multiplication algorithm. When $r = s = l$ and Strassen's Algorithm [54] is used the cost of the deterministic verification becomes approximately $\deg(\mathbf{H}(x))\mathcal{O}(r^{2.8})$.

■

Remark 2. *If the privacy does not hold, i.e., if more than z workers collude and we cannot guarantee, that $\alpha_1, \dots, \alpha_{z+d_u}$, $\beta_1, \dots, \beta_n$ are private from the malicious workers, then the scheme can be modified slightly in order to still guarantee that errors are detected with high probability. The master draws γ uniformly at random from $\mathbb{F}_q$. It then needs to compute the evaluations $\mathbf{F}(\gamma)$, $\mathbf{G}(\gamma)$ and $\mathbf{H}(\gamma)$ and call Algorithm 1 on the results. We can prove by the same steps as above, that the probability of not detecting an error in a round of this modified scheme is at most $\frac{\deg(\mathbf{H}_t^{(u)}(x))}{q} + \frac{1}{q}$. The modified scheme has higher computational cost since one extra evaluation of $\mathbf{F}$, $\mathbf{G}$ and $\mathbf{H}$ has to be computed in every round and for every cluster.*

Remark 3. *In the proposed version of SRPM3 (Section III-B) we run Algorithm 1 once per cluster. However, the algorithm can be modified to the following. An additional amount of $b_u > 0$ workers is added to cluster u such that $n_u \geq 2d_u + z + b_u - 1$ for $u > 1$ and $n_1 \geq 2d_u + 2z + b_1 - 1$. Once Algorithm 1, run on the polynomials, detects an error in cluster u, the master runs Algorithm 1 for each worker to detect the malicious workers. Those workers are removed from the system. Moreover, if at most b_u workers are malicious, the master uses the results from the remaining workers to interpolate $\mathbf{H}_t^{(u)}(x)$.*

A key difference between SRPM3 and the work of [50] is that the scheme in [50] verifies each individual worker, hence adding computational complexity to the scheme.

V. Proof of Privacy

We now show that the scheme does not leak information about $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ as well as the interpolation and evaluation points $\alpha_{t,1}, \dots, \alpha_{t,z+d_{\max}}, \beta_{t,1}, \dots, \beta_{t,n}$ of every round. Since the random matrices $\mathbf{R}_t$'s and $\mathbf{S}_t$'s used to ensure privacy and the α_t's and β_t's are generated independently for every round, it suffices to show that at each individual round the colluding workers do not obtain any information about the private entities.

For convenience of notation we group several random variables. Let $\mathcal{W}_{t,\mathcal{A}}$ denote the set of random variable representing the tasks received in round t by a set of workers $\mathcal{A} \subseteq \{1,\dots,n\}$. Define $\mathcal{N}_t \triangleq \{\mathtt{R}_{t,1},...,\mathtt{R}_{t,z},\mathtt{S}_{t,1},...,\mathtt{S}_{t,z}\}$ to be the set of random variables representing the randomly generated matrices for round t. We group the private entities in $\mathcal{M}_t \triangleq \{\mathtt{A},\mathtt{B},\mathtt{C},\alpha_{t,1},...,\alpha_{t,z+d_{\max}},\beta_{t,1},...,\beta_{t,n}\}$ where we abuse notation and use the Greek letters α and β to denote a random variable and its realization. The distinction is clear from the context.

Lemma 1 is a standard lemma in the literature on secret sharing, private distributed storage and private coded computing. We replicate the lemma and its proof here for completeness. The additional twist we add to the lemma is the privacy of the interpolation and evaluation points $\alpha_{t,1},\dots,\alpha_{t,z},\beta_{t,1},\dots,\beta_{t,n}$. The intuition behind the proof is that the random matrices are chosen to have entropy equal to the observation of the colluding workers. In addition, the colluding workers can neither interpolate $\mathbf{F}_t^{(u)}(x)$ nor interpolate $\mathbf{G}_t^{(u)}(x)$. Therefore, they obtain no information about the interpolation points $\alpha_{t,1},\dots,\alpha_{t,z+d_u}$ and the evaluation points $\beta_{t,1},\dots,\beta_{t,n}$.

Lemma 1. *In every round t of the SRPM3, the random variables of the set $\mathcal{M}_t$ are kept information theoretically private from any set of z colluding workers, i.e.,*

$$I(\mathcal{M}_t;\mathcal{W}_{t,\mathcal{Z}})=0,\forall \mathcal{Z}\subseteq\{1,...,n\},\ \textit{s.t.}\ |\mathcal{Z}|=z.$$

Proof: The conditional mutual information $I(\mathcal{N}_t;\mathcal{W}_{\mathcal{Z}}|\mathcal{M}_t)$ can be written in the following two ways [55]

$$I(\mathcal{N}_t;\mathcal{W}_{t,\mathcal{Z}}|\mathcal{M}_t)=H(\mathcal{W}_{t,\mathcal{Z}}|\mathcal{M}_t)-H(\mathcal{W}_{t,\mathcal{Z}}|\mathcal{N}_t,\mathcal{M}_t) \tag{8}$$
$$=H(\mathcal{N}_t|\mathcal{M}_t)-H(\mathcal{N}_t|\mathcal{W}_{t,\mathcal{Z}},\mathcal{M}_t). \tag{9}$$

Having (8) and (9), we can now write

$$H(\mathcal{W}_{t,\mathcal{Z}}|\mathcal{M}_t)=H(\mathcal{N}_t|\mathcal{M}_t)-H(\mathcal{N}_t|\mathcal{W}_{t,\mathcal{Z}},\mathcal{M}_t)+H(\mathcal{W}_{t,\mathcal{Z}}|\mathcal{N}_t,\mathcal{M}_t) \tag{10}$$
$$H(\mathcal{N}_t|\mathcal{M}_t)-H(\mathcal{N}_t|\mathcal{W}_{t,\mathcal{Z}},\mathcal{M}_t) \tag{11}$$
$$H(\mathcal{N}_t|\mathcal{M}_t) \tag{12}$$
$$H(\mathcal{N}_t) \tag{13}$$
$$H(\mathcal{W}_{t,\mathcal{Z}}). \tag{14}$$

Equation (11) follows from $H(\mathcal{W}_{t,\mathcal{Z}}|\mathcal{N}_t,\mathcal{M}_t)=0$ since the master computes the tasks only based on the contents of $\mathcal{N}_t$ and $\mathcal{M}_t$, i.e., $\mathcal{W}_{t,\mathcal{Z}}$ is a deterministic function of $\mathcal{N}_t$ and $\mathcal{M}_t$. To show that (12) holds, we show in Appendix A that $H(\mathcal{N}_t|\mathcal{W}_{t,\mathcal{Z}},\mathcal{M}_t)=0$ since one can reconstruct the random matrices $\mathbf{R}_{t,1},\dots,\mathbf{R}_{t,z},\mathbf{S}_{t,1},\dots,\mathbf{S}_{t,z}$ from any collection of z tasks and the realizations of the random variables in $\mathcal{M}_t$. Equation (13) follows by drawing the matrices in $\mathcal{N}_t$ independently from the random variables in $\mathcal{M}_t$, i.e., $H(\mathcal{N}_t\,|\,\mathcal{M}_t)=H(\mathcal{N}_t)$. The last step is to note that $H(\mathcal{N}_t)\geq H(\mathcal{W}_{t,\mathcal{Z}})$ since both $\mathcal{N}_t$ and $\mathcal{W}_{t,\mathcal{Z}}$ consist of z random matrices from $\mathbb{F}_q^{r\times s}$ and $\mathbb{F}_q^{s\times l}$ each and since $\mathcal{N}_t$ has maximum entropy because its elements are independently and uniformly distributed. Equality is achieved because conditioning never increases entropy and thus $H(\mathcal{W}_{t,\mathcal{Z}}|\mathcal{M}_t)\leq H(\mathcal{W}_{t,\mathcal{Z}})$.

It follows that

$$I(\mathcal{M}_t;\mathcal{W}_{t,\mathcal{Z}})=H(\mathcal{W}_{t,\mathcal{Z}})-H(\mathcal{W}_{t,\mathcal{Z}}|\mathcal{M}_t)=0,\forall\mathcal{Z}\subseteq\{w_1,\dots,w_n\},\ \text{s.t.}\ |\mathcal{Z}|=z.$$

■

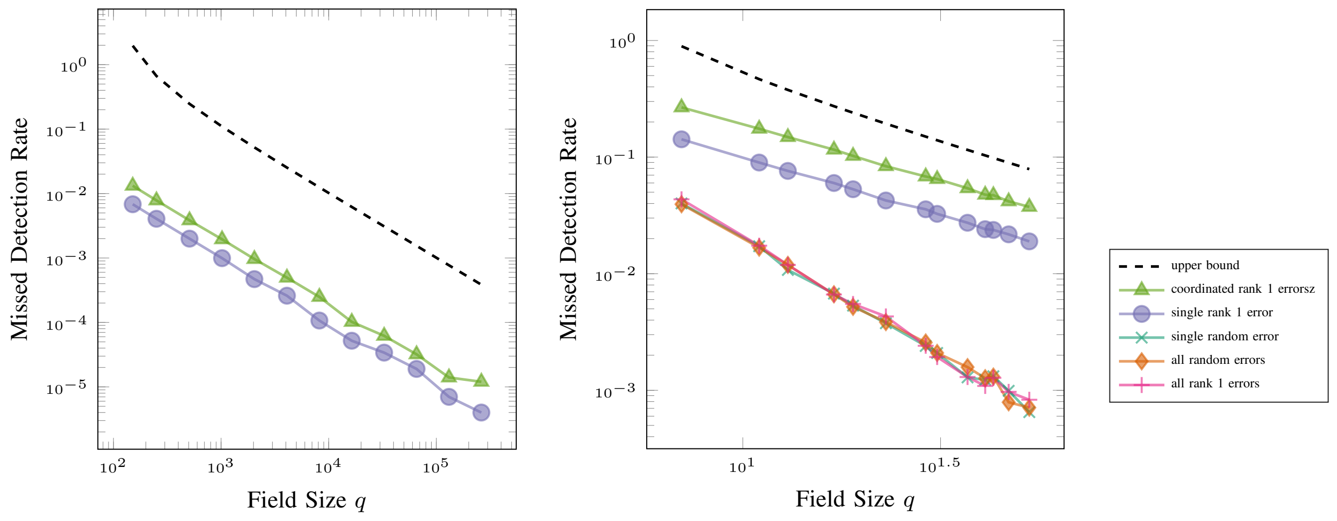
VI. Simulation Results

We provide simulation results showing the empirical rate of not detecting adversarial errors and showing the computational overhead incurred by our error detection algorithm. The simulations are shown for one cluster of workers. The rate of missed detection is compared against the field size q to show its decrease with q and how far is it from the provided upper bound on missed detection, cf., Figure 1a and Figure 1b. The computational overhead of the security check is compared against the number of workers per cluster. The overhead of the security check decreases with the increase of the number of workers per cluster, cf., Figure 2.

A. Rate of Missed Detection

We simulate different attack strategies that the malicious workers can run. For each strategy we measure the number of times the master does not detect the errors introduced by the workers. The rate of missed detection is the ratio of the number of times the master does not detect the error to the total number of times the computation is simulated. All simulations are considered per one cluster of workers. Figure 1 shows the rate of missed detection for the following attack strategies.

1) *Single random error:* A single worker adds a uniformly random non-zero matrix to the result. The other workers return the correct result.
2) *All random errors:* Every worker adds an independent uniformly random non-zero matrix to the result.



(a) Rate of missed detections in the first cluster consisting of $n_u = 100$ workers of a round with $r = s = l = 10$ in one million simulated rounds.

(b) Rate of missed detections in the first cluster consisting of $n_u = 3$ workers of a round with $r = s = l = 2$ in a hundred thousand simulated rounds.

Fig. 1: Simulated rate of missed detections.

3) *Single rank–*1 *error:* A single worker adds a random rank–1 matrix to the result. Recall that rank–1 matrices are the hardest to detect.
4) *All rank–*1 *errors:* Every worker adds an independent random rank–1 matrix to the result.
5) *Coordinated rank–*1 *errors:* Every worker adds a random rank–1 matrix to the result. All of the added matrices are linearly dependent which makes the error detection harder for the master.

Figure 1a shows the rate of missed detection for one million rounds of computations for error models 4 and 5, i.e., *all rank–*1 *errors* and *coordinated rank–*1 *errors*. The simulation parameters for this figure are $n_u = 100$ workers and $r = s = l = 10$. We plot in addition the upper bound on the probability of missed detections given in Theorem 1. For error models 1, 2 and 3, this simulation yielded no missed detections in one million rounds, so the missed detection rate is likely below 10^{-6}.

To measure the missed detection rates for all error models, we change the simulation parameters to $n_u = 3$ and $r = s = l = 2$ which allow the use of a smaller field size, thus higher rate of missed detections. For those simulations, one hundred thousand runs were enough to see missed detections. The rates of missed detections for all models are shown in Figure 1b.

Simulation results validate that rank–1 errors have the highest probability of not being detected by Algorithm 1. Further, they show that our upper bound of Theorem 1 is a loose bound for the probability of missed detection. Furthermore, the simulations show that the rate of missed detection is roughly $1/q$ for error models 1 and 2.

B. Computational Overhead

We plot in Figure 2 the ratio of CPU times for the security check to that spent for encoding and decoding the Lagrange polynomials in the first cluster of a round over the cluster size n_1. The simulations were performed with a large prime field ($q \approx 2^{62}$).

The computational complexity of encoding and decoding is $\mathcal{O}((rs + sl + rl)n_1 \log^2 n_1)$ with an FFT based algorithm for interpolation and evaluation as described in [56, Chapter 11], whereas the complexity of the security check is $O(rs + sl + rl)$. Simulation results validate our theoretical results by showing that the computational cost of verifying the correctness of the computations is minimal compared to the computational cost of the rest of the scheme. In addition, simulations show that the computational overhead of the security check decreases with the increase of the number of workers per cluster.

VII. Conclusion

We considered the heterogeneous and time-varying setting of secure and private distributed matrix-matrix multiplication. We introduced SRPM3, a secure private and rateless matrix-matrix multiplication scheme that allows a master to offload matrix-matrix multiplications to malicious, curious and heterogeneous workers. The scheme is a modification of RPM3 introduced by a subset of the authors. The additional component is the efficient verification of the correctness of the results sent by the workers using Freivald's algorithm. In contrast to distributed matrix-matrix multiplication schemes based on coding theory, SRPM3 tolerates the presence of an arbitrary number of malicious workers. The efficiency of the scheme is increased by grouping the workers in clusters and verifying the computation of the whole cluster at once. As an extra layer of security, for

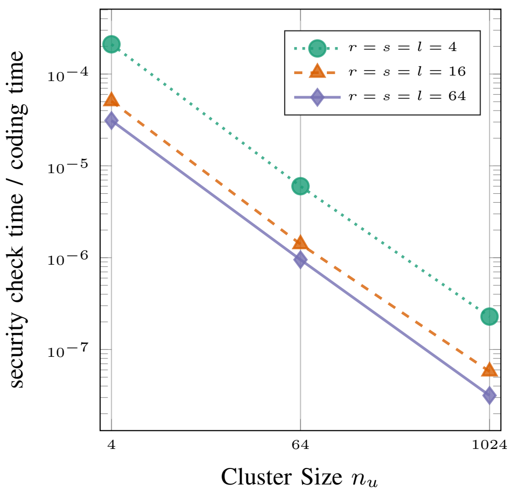


Fig. 2: Ratio of CPU times spent for the security check to CPU times spent for encoding/decoding the Lagrange polynomials for one cluster. The computational overhead of the security check is negligible and decreases with the increase of the number of workers per cluster.

a cluster where an error is detected, the master can run Freivald's algorithm on the results sent by each worker to detect the malicious workers and remove them from the system. Furthermore, redundant workers can be added per cluster so that the master can use the results sent by honest workers of each cluster.

As an extension of this work, we analyse the computation complexity of SRPM3 when the malicious workers share information with each other, i.e., the α's and β's are no longer private. In addition, we analyse the effect of finite field arithmetic on the precision of the computation for machine learning algorithms. Furthermore, we investigate cases in which the cost of encoding/decoding Lagrange polynomials is smaller than running the computation locally.

References

[1] J. Dean and L. A. Barroso, "The tail at scale," *Communications of the ACM*, vol. 56, no. 2, pp. 74–80, 2013.
[2] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le, *et al.*, "Large scale distributed deep networks," in *Advances in neural information processing systems*, pp. 1223–1231, 2012.
[3] G. Ananthanarayanan, A. Ghodsi, S. Shenker, and I. Stoica, "Effective straggler mitigation: Attack of the clones," in *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, pp. 185–198, 2013.
[4] G. Liang and U. C. Kozat, "Fast cloud: Pushing the envelope on delay performance of cloud storage with coding," *IEEE/ACM Transactions on Networking*, vol. 22, no. 6, pp. 2012–2025, 2014.
[5] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1514–1529, 2017.
[6] A. B. Das and A. Ramamoorthy, "Coded sparse matrix computation schemes that leverage partial stragglers," *arXiv preprint arXiv:2012.06065*, 2020.
[7] E. Vedadi and H. Seferoglu, "Adaptive coding for matrix multiplication at edge networks," *arXiv preprint arXiv:2103.04247*, 2021.
[8] A. Mallick, M. Chaudhari, and G. Joshi, "Rateless codes for near-perfect load balancing in distributed matrix-vector multiplication," *arXiv preprint arXiv:1804.10331*, 2018.
[9] A. Reisizadeh, S. Prakash, R. Pedarsani, and A. S. Avestimehr, "Coded computation over heterogeneous clusters," *IEEE Transactions on Information Theory*, vol. 65, no. 7, pp. 4227–4242, 2019.
[10] T. Baharav, K. Lee, O. Ocal, and K. Ramchandran, "Straggler-proofing massive-scale distributed matrix multiplication with d-dimensional product codes," in *IEEE International Symposium on Information Theory (ISIT)*, pp. 1993–1997, 2018.
[11] E. Ozfatura, S. Ulukus, and D. Gündüz, "Straggler-aware distributed learning: Communication–computation latency trade-off," *Entropy*, vol. 22, no. 5, p. 544, 2020.
[12] A. Ramamoorthy, L. Tang, and P. O. Vontobel, "Universally decodable matrices for distributed matrix-vector multiplication," in *IEEE International Symposium on Information Theory (ISIT)*, pp. 1777–1781, IEEE, 2019.
[13] S. Wang, J. Liu, and N. Shroff, "Coded sparse matrix multiplication," *arXiv preprint arXiv:1802.03430*, 2018.
[14] Q. Yu, M. Maddah-Ali, and S. Avestimehr, "Polynomial codes: an optimal design for high-dimensional coded matrix multiplication," in *Advances in Neural Information Processing Systems (NIPS)*, pp. 4403–4413, 2017.
[15] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, "A fundamental tradeoff between computation and communication in distributed computing," *IEEE Transactions on Information Theory*, vol. 64, no. 1, pp. 109–128, 2018.
[16] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding," *IEEE Transactions on Information Theory*, vol. 66, no. 3, pp. 1920–1933, 2020.
[17] M. Fahim, H. Jeong, F. Haddadpour, S. Dutta, V. Cadambe, and P. Grover, "On the optimal recovery threshold of coded matrix multiplication," in *55th Annual Allerton Conference on Communication, Control, and Computing*, pp. 1264–1270, 2017.
[18] Y. Keshtkarjahromi, Y. Xing, and H. Seferoglu, "Dynamic heterogeneity-aware coded cooperative computation at the edge," in *IEEE 26th International Conference on Network Protocols (ICNP)*, pp. 23–33, 2018.
[19] A. K. Pradhan, A. Heidarzadeh, and K. R. Narayanan, "Factored LT and factored raptor codes for large-scale distributed matrix multiplication," *CoRR*, vol. abs/1907.11018, 2019.
[20] H. A. Nodehi and M. A. Maddah-Ali, "Secure coded multi-party computation for massive matrix operations," *arXiv preprint arXiv:1908.04255*, 2019.
[21] A. Behrouzi-Far and E. Soljanin, "Efficient replication for straggler mitigation in distributed computing," *arXiv preprint arXiv:2006.02318*, 2020.

[22] P. Peng, E. Soljanin, and P. Whiting, "Diversity vs. parallelism in distributed computing with redundancy," in *IEEE International Symposium on Information Theory (ISIT)*, pp. 257–262, 2020.

[23] S. Wang, J. Liu, and N. Shroff, "Coded sparse matrix multiplication," in *Proceedings of the 35th International Conference on Machine Learning* (J. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, (Stockholmsmässan, Stockholm Sweden), pp. 5152–5160, PMLR, 10–15 Jul 2018.

[24] 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.

[25] S. Li and S. Avestimehr, "Coded computing," *Foundations and Trends® in Communications and Information Theory*, vol. 17, no. 1, 2020.

[26] R. Bitar, P. Parag, and S. El Rouayheb, "Minimizing latency for secure coded computing using secret sharing via Staircase codes," *IEEE Transactions on Communications*, 2020.

[27] R. G. D'Oliveira, S. El Rouayheb, and D. Karpuk, "GASP codes for secure distributed matrix multiplication," *IEEE Transactions on Information Theory*, vol. 66, no. 7, pp. 4038–4050, 2020.

[28] 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 *The 22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 1215–1225, 2019.

[29] 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.

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

[31] R. Bitar, Y. Xing, Y. Keshtkarjahromi, V. Dasari, S. El Rouayheb, and H. Seferoglu, "PRAC: Private and Rateless Adaptive Coded Computation at the Edge," in *SPIE - Defense + Commercial Sensing*, vol. 11013, 2019.

[32] R. Bitar, M. Xhemrishi, and A. Wachter-Zeh, "Rateless codes for private distributed matrix-matrix multiplication," in *IEEE International Symposium on Information Theory and its Applications (ISITA)*, 2020.

[33] R. Bitar, Y. Xing, Y. Keshtkarjahromi, V. Dasari, S. El Rouayheb, and H. Seferoglu, "Private and rateless adaptive coded matrix-vector multiplication," *EURASIP Journal on Wireless Communications and Networking*, vol. 2021, no. 1, pp. 1–25, 2021.

[34] 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, 2018.

[35] J. Kakar, S. Ebadifar, and A. Sezgin, "Rate-efficiency and straggler-robustness through partition in distributed two-sided secure matrix computation," *arXiv preprint arXiv:1810.13006*, 2018.

[36] J. Kakar, S. Ebadifar, and A. Sezgin, "On the capacity and straggler-robustness of distributed secure matrix multiplication," *IEEE Access*, vol. 7, pp. 45783–45799, 2019.

[37] Z. Jia and S. A. Jafar, "Cross subspace alignment codes for coded distributed batch matrix multiplication," *arXiv preprint arXiv:1909.13873*, 2019.

[38] N. Mital, C. Ling, and D. Gunduz, "Secure distributed matrix computation with discrete fourier transform," *arXiv preprint arXiv:2007.03972*, 2020.

[39] J. Kakar, A. Khristoforov, S. Ebadifar, and A. Sezgin, "Uplink cost adjustable schemes in secure distributed matrix multiplication," in *IEEE International Symposium on Information Theory (ISIT)*, pp. 1124–1129, IEEE, 2020.

[40] S. Ulukus, S. Avestimehr, M. Gastpar, S. Jafar, R. Tandon, and C. Tian, "Private retrieval, computing and learning: Recent progress and future challenges," *arXiv preprint arXiv:2108.00026*, 2021.

[41] C.-S. Yang and A. S. Avestimehr, "Coded computing for secure boolean computations," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 1, pp. 326–337, 2021.

[42] Y. Keshtkarjahromi, R. Bitar, V. Dasari, S. El Rouayheb, and H. Seferoglu, "Secure coded cooperative computation at the heterogeneous edge against byzantine attacks," in *IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, 2019.

[43] A. M. Subramaniam, A. Heidarzadeh, and K. R. Narayanan, "Collaborative decoding of polynomial codes for distributed computation," in *IEEE Information Theory Workshop (ITW)*, pp. 1–5, IEEE, 2019.

[44] M. Soleymani, R. E. Ali, H. Mahdavifar, and A. S. Avestimehr, "List-decodable coded computing: Breaking the adversarial toleration barrier," *arXiv preprint arXiv:2101.11653*, 2021.

[45] B. K. Dey, S. Jaggi, and M. Langberg, "Sufficiently myopic adversaries are blind," *IEEE Transactions on Information Theory*, vol. 65, no. 9, pp. 5718–5736, 2019.

[46] S. Li, R. Bitar, S. Jaggi, and Y. Zhang, "Network coding with myopic adversaries," in *IEEE International Symposium on Information Theory (ISIT)*, 2021.

[47] Q. Zhang, S. Kadhe, M. Bakshi, S. Jaggi, and A. Sprintson, "Talking reliably, secretly, and efficiently: A "complete" characterization," in *2015 IEEE Information Theory Workshop (ITW)*, pp. 1–5, IEEE, 2015.

[48] J. Song, Q. Zhang, M. Bakshi, S. Jaggi, and S. Kadhe, "Multipath stealth communication with jammers," in *2018 IEEE International Symposium on Information Theory (ISIT)*, pp. 761–765, IEEE, 2018.

[49] R. Bitar and S. Jaggi, "Communication efficient secret sharing in the presence of malicious adversary," in *IEEE International Symposium on Information Theory (ISIT)*, pp. 548–553, IEEE, 2020.

[50] T. Tang, R. E. Ali, H. Hashemi, T. Gangwani, S. Avestimehr, and M. Annavaram, "Verifiable coded computing: Towards fast, secure and private distributed machine learning," *arXiv preprint arXiv:2107.12958*, 2021.

[51] R. Freivalds, "Fast probabilistic algorithms," in *Mathematical Foundations of Computer Science 1979* (J. Bečvář, ed.), (Berlin, Heidelberg), pp. 57–69, Springer Berlin Heidelberg, 1979.

[52] J. T. Schwartz, "Fast probabilistic algorithms for verification of polynomial identities," *J. ACM*, vol. 27, p. 701–717, Oct. 1980.

[53] R. Zippel, "Probabilistic algorithms for sparse polynomials," in *Symbolic and Algebraic Computation* (E. W. Ng, ed.), (Berlin, Heidelberg), pp. 216–226, Springer Berlin Heidelberg, 1979.

[54] V. Strassen, "Gaussian elimination is not optimal," *Numerische Mathematik*, vol. 13, pp. 354–356, Aug 1969.

[55] T. M. Cover, *Elements of information theory*. John Wiley & Sons, 1999.

[56] J. v. Zur Gathen, *Modern computer algebra*. Cambridge [u.a.]: Cambridge Univ. Press, 3. ed. ed., 2013.

Appendix A
Proof That $\mathcal{N}_t$ Can Be Recovered From $\mathcal{W}_{\mathcal{Z}}$ and $\mathcal{M}_t$

Proposition 2. *The values in $\mathcal{N}_t$ can be derived from the values in $\mathcal{W}_{t,\mathcal{Z}}$ and $\mathcal{M}_t$.*

Proof: For every cluster u the polynomial $\mathbf{F}_t^{(u)}(x)$ is the Lagrange polynomial fit to the set of points

$$\{(\alpha_{t,1}, \mathbf{R}_{t,1}), \dots, (\alpha_{t,z}, \mathbf{R}_{z,t}), (\alpha_{t,z+1,t}, \widetilde{\mathbf{A}}_{t,1}^{(u)}), \dots, (\alpha_{t,z+d_u}, \widetilde{\mathbf{A}}_{t,d_u}^{(u)})\}.$$

It can be written as

$$\mathbf{F}_t^{(u)}(x) = \sum_{i=1}^{z} l_{t,i}(x)\mathbf{R}_{t,i} + \sum_{i=1}^{d_u} l_{t,z+i}(x)\widetilde{\mathbf{A}}_{t,i}^{(u)},$$

where $l_{t,1}(x), \ldots, l_{t,z+d_u}$, are the Lagrange basis functions only depending on $\alpha_{t,1}, \ldots, \alpha_{t,z+d_u}$. If $\mathcal{M}_t$ is known, so is the value of the second sum. It can be subtracted from $\mathbf{F}_t^{(u)}(x)$ producing

$$\hat{\mathbf{F}}_t(x) = \sum_{i=1}^{z} l_{t,i}(x)\mathbf{R}_{t,i},$$

which is the same for each cluster u. As worker's tasks in $\mathcal{W}_{\mathcal{Z}}$ give us z independent evaluations of $\hat{\mathbf{F}}_t(x)$, the resulting linear system can be solved for $\mathbf{R}_{t,1}, ..., \mathbf{R}_{z,t}$. Performing the analogous steps for $\mathbf{G}_t^{(u)}(x)$ shows how $\mathbf{S}_{t,1}, ..., \mathbf{S}_{z,t}$ can be recovered.

■