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 multiplication. This work is motivated by the literature on myopic adversaries in network coding. 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; thus has 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. We apply this security framework to schemes with adaptive rates that divide the workers into clusters and thus provide flexibility in trading decoding complexity for efficiency. Our new 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. *Proofs and additional explanation can be found in [1].*

1 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.

Some workers, referred to as *stragglers*, may be significantly slower than others [2, 3], thus outweighing the benefits of parallelising the computation [4–6]. 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 of both input matrices, where a limited subset of workers with unbounded computational power are eavesdroppers. In addition, some of the workers are malicious and deliberately send corrupted computation to the master aiming 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 a heterogeneous and time-varying setting. The response time of the workers is different and varies over time. Examples of such applications include edge computing and internet of things (IoT) networks in which small devices collaborate to run intensive computations.

Related work: Coding for straggler mitigation witnessed a significant attention from the scientific community. Using codes provides fast, private and reliable distributed computing, see e.g., [6–14], and the survey [15]. Information-theoretic privacy and straggler mitigation in coded computing is achieved by using secret sharing, see e.g., [16–23], and the survey [24]. Security against malicious workers is considered in [25–29]. The works of [25, 26] design codes in which each malicious worker can inflict a double damage. This idea holds through the Singleton-bound for distributed storage, network coding and MDS codes. It has been shown for network coding and distributed storage [30–34] that when the malicious workers have limited knowledge about the master's data, the master can half the damage of the malicious workers with high probability. The advantage of distributed computing is that the master can be seen as the sender and the receiver. The master can thus alleviate the assumption of limited knowledge of the malicious workers. In [27] 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. In [28, 29] 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 [29] is the high computational complexity incurred by the master. Concurrently and independently of our work, the authors of [35] present a scheme to tolerate malicious workers in coded computing with low computational complexity. The ideas used in [35] are similar to the ideas used in this work. However, [28, 29, 35] consider security in settings where the workers are assumed to have similar resources. In addition, a maximum amount of stragglers is assumed.

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 scheme introduced by a subset of the authors [18]. 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 [36]. 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 [35]. 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.

2 Preliminaries

Notation For any positive integer a we define $[a] \triangleq \{1, \ldots, a\}$. 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.

Problem setting A master node 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 response time of the workers is different. The workers can be grouped into $c > 1$ clusters of workers that have similar response 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 response time of the workers can change during the multiplication process. 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) Up to z, $1 \leq z < \min_{u\in[c]} n_u$, workers collude to eavesdrop on $\mathbf{A}$ and/or $\mathbf{B}$. If $z = 1$, we say the workers do not collude.
5) Workers are malicious. An arbitrary subset of the workers may collaboratively send noisy computations to the master to corrupt the whole computation process. Despite their collaboration to jam the computation, we assume that at most up to z malicious workers share information about $\mathbf{A}$ and $\mathbf{B}$ with each other, i.e., are colluding. 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.

$\mathbf{A}$ is divided into m equally sized blocks of rows and $\mathbf{B}$ is divided into k equally sized blocks of columns[1] such that $\mathbf{C}_{i,j} = \mathbf{A}_i\mathbf{B}_j$, $i \in [m], j \in [k]$. 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 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

[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.

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, [18]). *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 $\varepsilon > 0$ is arbitrarily small. 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.

3 SRPM3 Scheme

SRPM3 is based in large parts on the RPM3 scheme presented in [18] by a subset of the authors. We briefly explain the common aspects of RPM3 and SRPM3. More details about the scheme and its metrics of interest can be found in [18, 19]. The additional part on the adversarial error detection is 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.

Encoding A factored fountain code [12] encodes $\{\mathbf{A}_1, \ldots, \mathbf{A}_m\}$ into $\{\widetilde{\mathbf{A}}_1, \widetilde{\mathbf{A}}_2, \ldots\}$ and $\{\mathbf{B}_1, \ldots, \mathbf{B}_k\}$ into $\{\widetilde{\mathbf{B}}_1, \widetilde{\mathbf{B}}_2, \ldots\}$. Using a factored fountain code ensures that $\widetilde{\mathbf{C}}_1 \triangleq \widetilde{\mathbf{A}}_1\widetilde{\mathbf{B}}_1, \widetilde{\mathbf{C}}_2 \triangleq \widetilde{\mathbf{A}}_2\widetilde{\mathbf{B}}_2, \ldots$ 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 is 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., $d_{\max} \triangleq \max_{u \in [c]} d_u$.

At the start of a round t the master does the following: *i)* chooses $d_{\max} + z + n$ distinct elements $\{\alpha_{t,1}, \ldots, \alpha_{t,d_{\max}+z}, \beta_{t,1}, \ldots, \beta_{t,n}\}$ of $\mathbb{F}_q$ uniformly at random[2]; and *ii)* draws

[2] Note that in the original RPM3 scheme $\alpha_1, \ldots, \alpha_{d_{max}+z}, \beta_1, \ldots, \beta_n$ 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.

$2z$ matrices $\{\mathbf{R}_{t,1},\ldots,\mathbf{R}_{t,z}\}$ and $\{\mathbf{S}_{t,1},\ldots,\mathbf{S}_{t,z}\}$ 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 $\{\widetilde{\mathbf{A}}_{t,1}^{(u)},\ldots,\widetilde{\mathbf{A}}_{t,d_u}^{(u)}\}$ and $\{\widetilde{\mathbf{B}}_{t,1}^{(u)},\ldots,\widetilde{\mathbf{B}}_{t,d_u}^{(u)}\}$ and computes the following two Lagrange polynomials of degree d_u+z-1

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

where for every $i\in[z+d_u]$, $l_i(x)=\prod_{j\in[d_u+z]\setminus\{i\}}\frac{x-\alpha_{t,j}}{\alpha_{t,i}-\alpha_{t,j}}$ is the Lagrange basis polynomial satisfying $l_i(\alpha_{t,i})=0$ and $l_i(\alpha_{t,j})=1$ for all $j\in[z+d_u]\setminus\{i\}$. Consequently, for all $i=1,\ldots,z$, and $j=z+1,\ldots,z+d_u$ 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 $\widetilde{\mathbf{C}}_{t,1}=\widetilde{\mathbf{A}}_{t,1}\widetilde{\mathbf{B}}_{t,1},\ldots,\widetilde{\mathbf{C}}_{t,d_u}=\widetilde{\mathbf{A}}_{t,d_u}\widetilde{\mathbf{B}}_{t,d_u}$ at $\alpha_{t,z+1},\ldots,\alpha_{t,z+d_u}$, respectively.

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

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 $\mathbf{H}_t^{(1)}(\alpha_{t,i})=\mathbf{H}_t^{(2)}(\alpha_{t,i})=\cdots=\mathbf{H}_t^{(c)}(\alpha_{t,i})=\mathbf{R}_{t,i}$ for all $i=1,\ldots,z$. 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 $\alpha_{t,z+1},\ldots,\alpha_{t,d_u}$ produces $\{\widetilde{\mathbf{C}}_{t,1}^{(u)},\ldots,\widetilde{\mathbf{C}}_{t,d_u}^{(u)}\}$.

At this point, the security check described in Section 4 is performed. If the security check passes, the master feeds the fountain coded matrices $\widetilde{\mathbf{C}}_{t,i}^{(u)}$ into a peeling decoder [37]. The scheme is finished when all the $\mathbf{C}_{i,j}$'s are decoded by the peeling decoder.

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 time of the workers and updates the clustering such that workers with similar response time 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.

4 Adversarial Error Detection

Error Detection in Matrix-Matrix Multiplication Freivalds' algorithm (Algorithm 1) [36] is an efficient way to verify the correctness of a single matrix-matrix multiplication

Algorithm 1: Freivalds' algorithm.

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 $\mathbb{F}_q^l$;
if $\mathbf{X}_1\mathbf{X}_2\boldsymbol{\nu} = \mathbf{X}_3\boldsymbol{\nu}$ **then**
  return *True*;
else
  return *False*;
end

with high probability. Freivalds' 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 probability at least $1 - \frac{1}{q}$ [36, Theorem 3]. Freivalds' algorithm probabilistically verifies the matrix-matrix multiplication with only three matrix-vector multiplications: $\boldsymbol{\nu}' \triangleq \mathbf{X}_2\boldsymbol{\nu}$, $\mathbf{X}_1\boldsymbol{\nu}'$ and $\mathbf{X}_3\boldsymbol{\nu}$. This is in contrast to re-computing $\mathbf{X}_1\mathbf{X}_2$ and deterministically verifying the computation.

Error Detection in Polynomial Multiplication A polynomial multiplication can be verified similarly [36]. 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 [38, 39].

Adversarial Error Detection in SRPM3 We combine the methods for error detection in 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 three polynomial matrices $\mathbf{H}_t^{(u)}(x) \in \mathbb{F}_q^{\frac{r}{m}\times\frac{\ell}{k}}$, $\mathbf{F}_t^{(u)}(x) \in \mathbb{F}_q^{\frac{r}{m}\times s}$ and $\mathbf{G}_t^{(u)}(x) \in \mathbb{F}_q^{s\times\frac{\ell}{k}}$ such that $\mathbf{H}_t^{(u)}(x) \neq \mathbf{F}_t^{(u)}(x)\mathbf{G}_t^{(u)}(x)$, the probability that Algorithm returns* True *when applied to these polynomial matrices is bounded from above by*

$$\Pr\Big(\textit{Algorithm 1 returns } \text{True} | \mathbf{H}_t^{(u)}(x) \neq \mathbf{F}_t^{(u)}(x)\mathbf{G}_t^{(u)}(x)\Big) \leq \frac{\deg(\mathbf{H}_t^{(u)}(x))}{q - \deg(\mathbf{H}_t^{(u)}(x)) - 1}.$$

The probability of SRPM3 not *detecting an error in cluster $u \in [c]$ for every round t is bounded from above as in* (4) *even if all n_u workers collaborate on the attack.*

$$\Pr(\textit{SRPM3 not detecting an error}) \leq \frac{\deg(\mathbf{H}_t^{(u)}(x))}{q - \deg(\mathbf{H}_t^{(u)}(x)) - 1} + \frac{1}{q}. \tag{4}$$

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

The proof of Theorem 1 is omitted and can be found in [1].

Remark 1. If the privacy does not hold, i.e., if more than z workers collude and we cannot guarantee, that $\alpha_1, \ldots, \alpha_{z+d_u}$, $\beta_1, \ldots, \beta_n$ 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. It can be shown 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, see [1] for a detailed discussion.

Remark 2. In the proposed version of SRPM3, 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 $n_1 \geq 2d_u + 2z + b_1 - 1$. If 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 [35] is that the scheme in [35] verifies each individual worker, hence adding computational complexity to the scheme.

5 Simulation Results

Rate of Missed Detection We simulated two possible attack strategies that can be run by the malicious workers. For each strategy, we measured the number of times the master does not detect the errors introduced by the workers, cf. Figure 1. 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. We consider the following attack strategies.

1. *Single rank–*1 *error:* A single worker adds a random rank–1 matrix to the result. From the analysis of Algorithm 1, rank–1 matrices are the hardest to detect [1].
2. *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.

We plot in addition the upper bound on the probability of missed detections given in Theorem 1. 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.

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 (the largest prime less than 2^{62}).

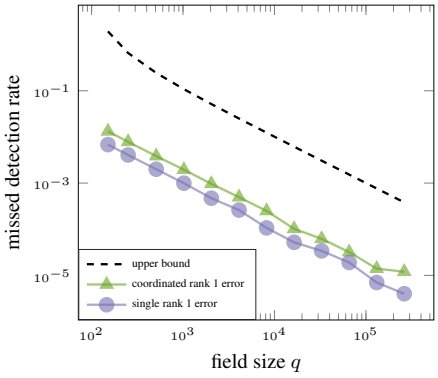

Fig. 1: Rate of missed detections in the first cluster consisting of $n_u = 100$ workers of a round with $r = s = l = 10$ in one million rounds.

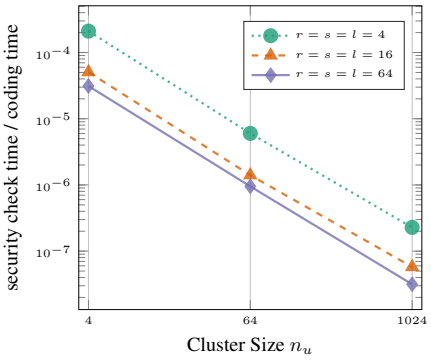

Fig. 2: Ratio of CPU time spent for the security check to CPU time spent for encoding/decoding the Lagrange polynomials for one cluster.

Computational complexity of encoding and decoding with an FFT based algorithm for interpolation and evaluation is $\mathcal{O}\left(\left(\frac{rs}{m} + \frac{s\ell}{k} + \frac{r\ell}{mk}\right) n_1 \log^2 n_1\right)$ as described in [40, Chapter 11], whereas the complexity of the security check is $\mathcal{O}\left(\frac{rs}{m} + \frac{s\ell}{k} + \frac{r\ell}{mk}\right)$. Simulation results validate our theoretical findings 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 when increasing n_u. Note that the field size can be reduced at the expense of repeating the security check multiple times [1].

6 Conclusion

We considered a heterogeneous and time-varying setting of secure and private distributed matrix-matrix multiplication. We introduced SRPM3, a new scheme that allows a master to offload matrix-matrix multiplications to malicious, curious and heterogeneous workers. 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 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.

References

1. C. Hofmeister, R. Bitar, M. Xhemrishi, and A. Wachter-Zeh, "Secure private and adaptive matrix multiplication beyond the singleton bound," *arXiv preprint arXiv:2108.05742*, 2021.

2. J. Dean and L. A. Barroso, "The tail at scale," *Communications of the ACM*, vol. 56, no. 2, pp. 74–80, 2013.
3. 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.
4. 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.
5. 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.
6. 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.
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. 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.
11. 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.
12. 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.
13. 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.
14. 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.
15. S. Li and S. Avestimehr, "Coded computing," *Foundations and Trends® in Communications and Information Theory*, vol. 17, no. 1, 2020.
16. 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.
17. 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.
18. 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.
19. R. Bitar, M. Xhemrishi, and A. Wachter-Zeh, "Adaptive private distributed matrix multiplication," pp. 1–1, 2022.
20. 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.

21. 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.
22. 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.
23. B. Hasircioglu, J. Gomez-Vilardebo, and D. Gunduz, "Bivariate polynomial codes for secure distributed matrix multiplication," *arXiv preprint arXiv:2106.07731*, 2021.
24. 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.
25. 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.
26. 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.
27. 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.
28. 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.
29. 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.
30. 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.
31. S. Li, R. Bitar, S. Jaggi, and Y. Zhang, "Network coding with myopic adversaries," in *IEEE International Symposium on Information Theory (ISIT)*, 2021.
32. 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.
33. 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.
34. 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.
35. 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.
36. R. Freivalds, "Fast probabilistic algorithms," in *Mathematical Foundations of Computer Science 1979* (J. Bečvář, ed.), (Berlin, Heidelberg), pp. 57–69, Springer Berlin Heidelberg, 1979.
37. D. J. MacKay, "Fountain codes," *IEEE Proceedings-Communications*, vol. 152, no. 6, pp. 1062–1068, 2005.
38. J. T. Schwartz, "Fast probabilistic algorithms for verification of polynomial identities," *J. ACM*, vol. 27, p. 701–717, Oct. 1980.
39. 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.
40. J. v. Zur Gathen, *Modern computer algebra*. Cambridge [u.a.]: Cambridge Univ. Press, 3. ed. ed., 2013.