Ph.D. DISSERTATION

Privacy and Security in Coded Computation and Cache-aided Information Retrieval

분산 컴퓨팅과 캐시를 접목한 정보 검색에서의
보안 및 프라이버시

Minchul Kim

Feburary 2020

Department of Electrical and Computer Engineering
College of Engineering
Seoul National University

Ph.D. DISSERTATION

Privacy and Security in Coded Computation and Cache-aided Information Retrieval

분산 컴퓨팅과 캐시를 접목한 정보 검색에서의 보안 및 프라이버시

Minchul Kim

Feburary 2020

Department of Electrical and Computer Engineering
College of Engineering
Seoul National University

Privacy and Security in Coded Computation and Cache-aided Information Retrieval

분산 컴퓨팅과 캐시를 접목한 정보 검색에서의 보안 및 프라이버시

지도교수 이 정 우

이 논문을 공학박사 학위논문으로 제출함

2019년 11월

서울대학교 대학원

전기 정보 공학부

김 민 철

김민철의 공학박사 학위 논문을 인준함

2019년 12월

Privacy and Security in Coded Computation and Cache-aided Information Retrieval

Advisor: Jungwoo Lee

Presented to the Graduate School of Seoul National University in Partial Fulfillment of the Requirements for

The Degree of Doctor of Philosophy

November 2019

by

Minchul Kim

Department of Electrical and Computer Engineering
College of Engineering
Seoul National University

This dissertation is approved for

The Degree of Doctor of Philosophy

December 2019

Abstract

As a major format of data changes from the text to the videos, the amount of memory for storing data increases exponentially, as well as the amount of computation for handling the data. As a result, to alleviate these burdens of storage and computations, the distributed systems are actively studied. Meanwhile, since low latency is one of the main objectives in 5G communications, recent techniques such as edge computing or federated learning in machine learning become important. Since the decentralized systems are fundamental characteristics of these techniques, the distributed systems which include the decentralized systems also become important.

In this dissertation, I consider the distributed systems for storage and computation. For the data storage, large-scale data centers collectively store a library of files where the size of each file is tremendous. When a user needs a specific file, it can be downloaded from distributed data centers. In this system, minimizing the amount of download is a significant concern. The user's privacy in this system implies that the user should conceal the index of its desired file against the databases. This kind of problem is referred to as *private information retrieval* (PIR) problem. The goal of PIR problem is to minimize the amount of download from the databases while ensuring the user's privacy.

Meanwhile, for a large amount of computation, the user can divide the whole computation into sub-computations and distribute them to external workers who constitute a distributed system. There can be three cases for the computation. Firstly, the user may own all of the data to be computed and sends both of its data and instructions for the computation to the workers. Secondly, the workers collectively own all of the data and the user just sends instructions for the data selection and computation to the workers. Thirdly, the user and the workers have their own data respectively and the user sends the data and instructions for the data selection and computation to the workers. Since some of the workers can be slow for various reasons, the user may use a coding technique, e.g., an erasure code, to avoid the delaying effect caused by the slow workers. This kind of technique is referred to as *coded computation*. In these systems, speeding up the computation process is a significant concern. In this dissertation, I focus on the third system. In the considered system, the privacy is similar to that of distributed systems for storage. On the other hand, the security implies that the user should conceal the content of its own data against the workers so that the workers do not have any information about the user's own data.

In this dissertation, I consider the user's privacy in distributed systems for storage, and both of the privacy and security in distributed systems for the computation. In case of the distributed systems for storage, since the user does not have its own data, the data security on the user's data cannot be considered. Particularly, I propose some achievable schemes that ensure the privacy and security in these systems.

To begin with, as a new variation of PIR problem, I consider a user's cache that has some pre-stored data of databases' library. I refer to this problem as cache-aided PIR problem. By introducing the user's cache in the PIR problem, the amount of download from the databases is significantly reduced. The achievable scheme is based on the optimal scheme for conventional PIR problem. In the achievable scheme, the pre-store cache was exploited as an side information, which reduces the amount of download, compared to the PIR problem without cache.

Secondly, I consider the master's privacy in coded computation. In the system model, the workers have their own data, as well as the master. The workers' data constitutes a library of several files. The master should compute a function of its own data and a specific file in the library. The master's privacy implies that the workers' should not know which file in the library is desired by the user. I refer to this problem as private coded computation and propose an achievable scheme of private coded computation, namely private polynomial codes. The private polynomial codes are based on the polynomial codes which were proposed in the conventional coded computation system. In the achievable scheme, the workers are grouped for the privacy and asynchronous scheme is considered, which was not considered in the conventional polynomial codes. Due to the asynchronous scheme, the proposed scheme achieves the faster computation time, compared to the modified optimal RPIR scheme.

Lastly, I consider the data security in coded computation, as well as the master's privacy. The system model is similar to that of private coded computation. The data security implies that the master should protect its own data against the workers. I refer to this problem as private secure coded computation and propose an achievable scheme, namely private secure polynomial codes. The private secure polynomial codes are based on the polynomial codes which were proposed in the conventional coded computation system. By modifying the private polynomial codes, the private secure polynomial codes and private secure polynomial codes are compared in terms of computation time and communication load. As a result, the private secure polynomial codes achieves faster computation time for the same communication load.

keywords: Distributed Computing, Private Information Retrieval, Information Theory, Security

student number: 2014-21647

Contents

Abstract i

Contents v

List of Tables viii

List of Figures ix

1 Introduction 1

1.1 Related work 3

1.1.1 Private information retrieval 3

1.1.2 Coded computation 4

1.2 Contributions and Organization 5

2 Cache-aided Private Information Retrieval 8

2.1 Introduction 8

2.2 System model 9

2.3 Main results 12

2.4 Achievable Scheme 17
2.4.1 Cacheless phase 17
2.4.2 Cache-assisted phase 21
2.4.3 Cache-aided PIR 24
2.5 Tightness of achievable scheme 29
2.6 Conclusion and follow-up works 30

3 Private Coded Computation 32

3.1 Introduction 32
3.2 System model 37
3.3 Main results 41
3.4 Private polynomial codes 42
3.4.1 First example 42
3.4.2 Second example 48
3.4.3 General description 52
3.4.4 Privacy proof 56
3.4.5 Performance analysis 59
3.4.6 Special cases 61
3.5 Simulation results 62
3.5.1 Computation time 62
3.5.2 Communication load 68
3.6 Conclusion 69

4 Private Secure Coded Computation **71**

4.1 Introduction 71

4.2 Main results 75

4.3 Private secure polynomial codes 76

4.3.1 Illustrative example 76

4.3.2 General description 80

4.3.3 Performance analysis 83

4.3.4 Privacy and security proof 84

4.4 Simulation results 85

4.4.1 Computation time 86

4.4.2 Communication load 90

4.5 Conclusion 91

5 Conclusion **93**

5.1 Summary 93

5.2 Future directions 94

Abstract (In Korean) **105**

Acknowledgement **107**

List of Tables

3.1 The communication load comparison 69

List of Figures

2.1 The overall process of cache-aided PIR with M messages and N databases. 11

2.2 The performance gap between the proposed scheme and the lower bound for $N = 3$ and $M = 4$ 14

2.3 The performance gap between the proposed scheme and the lower bound for $N = 3$ and $M = 50$ 15

2.4 The performance gap between the proposed scheme and the lower bound for $N = 10$ and $M = 4$ 16

2.5 The process of cacheless download for $N = 2$ and $M = 3$. 18

2.6 The process of cache-assisted phase for $N = 3$ and $M = 3$. 23

3.1 Motivating example of private coded computation. 34

3.2 The overall process of private coded computation. 38

3.3 The overall process of the first example. 43

3.4 The overall process of the second example. 49

3.5 The overall process of proposed scheme 53

3.6 The computation time comparison for $N = 12$, $M = 4$, $\gamma = 1$, and varying μ from 10^{-1} to 10. 65

3.7 The computation time comparison for $N = 12$, $M = 8$, $\gamma = 0.1$, and varying μ. 66

3.8 The computation time comparison for $N = 12$, $M = 4$, $\gamma = 1$, and varying μ. 67

4.1 The overall process of private secure coded computation 72

4.2 Motivating example of private secure coded computation 74

4.3 The overall process of the illustrative example 77

4.4 The overall process of PSPC. 81

4.5 The computation time comparison for $N = 12$, $M = 4$, $\gamma = 0.1$, and varying μ. 87

4.6 The computation time comparison for $N = 12$, $M = 8$, $\gamma = 0.1$, and varying μ. 88

4.7 The computation time comparison for $N = 12$, $M = 4$, $\gamma = 0.0001$, and varying μ. 89

Chapter 1

Introduction

As a major format of data changes from the text to the videos, the amount of memory for storing data increases exponentially, as well as the amount of computation for handling the data. As a result, to alleviate these burdens of storage and computations, the distributed systems are actively studied. Meanwhile, since low latency is one of the main objectives in 5G communications, recent techniques such as edge computing or federated learning in machine learning become important. Since the decentralized systems are fundamental characteristics of these techniques, the distributed systems which include the decentralized systems also become important.

In this dissertation, I consider the distributed systems for storage and computation. For the data storage, large-scale data centers collectively store a library of files where the size of each file is tremendous. When a user needs a specific file, it can be downloaded from distributed data centers. In this system, minimizing the amount of download is a significant concern. The user's privacy in this system implies that the user should conceal the index of its desired file against the databases. This kind of problem is referred to as *private information retrieval* (PIR) problem. The goal of PIR problem is to minimize the amount of download from the databases while ensuring the user's privacy.

Meanwhile, for a large amount of computation, the user can divide the whole computation into sub-computations and distribute them to external workers who constitute a distributed system. There can be three cases for the computation. Firstly, the user may own all of the data to be computed and sends both of its data and instructions for the computation to the workers. Secondly, the workers collectively own all of the data and the user just sends instructions for the data selection and computation to the workers. Thirdly, the user and the workers have their own data respectively and the user sends the data and instructions for the data selection and computation to the workers. Since some of the workers can be slow for various reasons, the user may use a coding technique, e.g., an erasure code, to avoid the delaying effect caused by the slow workers. This kind of technique is referred to as *coded computation*. In these systems, speeding up the computation process is a significant concern. In this dissertation, I focus on the third system. In the considered system, the privacy is similar to that of distributed systems for storage. On the other hand, the security implies that the user should conceal the content of its own data against the workers so that the workers do not have any information about the user's own data.

In this dissertation, I consider the user's privacy in distributed systems for storage, and both of the privacy and security in distributed systems for the computation. In case of the distributed systems for storage, since the user does not have its own data, the data security on the user's data cannot be considered. Particularly, I propose some achievable schemes that ensure the privacy and security in these systems.

1.1 Related work

1.1.1 Private information retrieval

For commercial network services based on cloud storage, where a user downloads files from a number of databases in wireless, the user's privacy is important. Accordingly, diverse approaches for ensuring privacy have been proposed. The private information retrieval (PIR) is one way of protecting user's privacy from the content provider. In a PIR problem, a user desires to privately retrieve one out of M messages. To retrieve the message, the user sends queries and then downloads some combinations of the desired message and the other undesired messages from N databases. After the retrieval, N databases must not know the index of the desired message. The goal of PIR problem is to minimize the amount of download while keeping user's privacy.

Since introduced in [11], the PIR problem has treated 1-bit messages [11]- [13]. In this scenario, since the size of message is small, the amount of queries that a user sends to the databases should be considered as well as the amount of download. On the other hand, if the size of message is much larger than that of queries, the amount of queries can be ignored and only the amount of download cost should be considered.

Recently, under the assumption that the message size is much bigger, information-theoretic capacity of PIR without cache for a replication-based storage system has been derived [14]. At the information theoretic capacity of PIR without cache, the downloaded undesired messages are minimized. The capacity-achieving scheme in [14] is basis for proposed cache-aided PIR scheme.

1.1.2 Coded computation

In a distributed computing system where a master partitions a massive computation into smaller sub-computations and distributes these sub-computations to several workers in order to reduce the runtime to complete the whole computation, some slow workers are bottleneck of the process. These slow workers are called *stragglers* and mitigating the effect of these stragglers is one of the major issues in distributed computing. Recently, a coding technique was introduced for straggler mitigation [19]. In [19], for a matrix-vector multiplication, the matrix is (n, k)-MDS coded and distributed to n workers so that each encoded matrix is assigned to one worker. Each worker multiplies the coded submatrix by a vector and returns the multiplication to the master. After k out of n workers return their multiplications, the master is able to decode the whole computation. Since the computation of the slowest $n - k$ workers is ignored, at most $n - k$ stragglers are mitigated. This kind of approach to distributed computing is referred to as *coded computation*. Several follow-up studies of coded computation were proposed [20]- [38].

In [39], a coded computation scheme for matrix multiplication was proposed, where the minimum number of workers for the master to decode the whole computation does not depend on the number of workers. In [40], a coded computation scheme that sub-blocks a MDS code into small blocks was proposed. In this scheme, several small blocks are assigned to each worker. Each worker processes its assigned blocks sequentially, block-by-block, and transmits the partial per-block results to the master. Since the size of each block is small enough for the stragglers to compute, all of the workers contribute to the computation, thus reducing the computation time. However, after processing their assigned blocks, faster workers stop working and must wait for slower workers to process their blocks. If these faster workers could continue working throughout the coded computation process, the computation time would be further reduced. Let us call a coded computation scheme where all of the workers continue working until the completion of the coded computation process by *asynchronous coded computation*.

1.2 Contributions and Organization

In this dissertation, I propose coding techniques for privacy and security in distributed networks for data storage and data computation.

In Chapter 2, I consider the PIR problem in which a user has pre-stored data in its cache, for a non-colluding replication-based storage system. Since databases do not collude each other, each database cannot know what has been downloaded in other databases. Caching is a technique that reduces the peak traffic during peak-traffic periods, by pre-storing some portion of files during the off-peak periods. In particular, when there are several users that request files from the same database, caching with coding schemes, which is called coded caching, reduces the peak traffic compared to uncoded caching more efficiently [1]- [10]. The caching technique becomes more significant as each file size in the database becomes larger so that the congestion at peak traffic time deteriorates. This situation matches our assumption in PIR problems, where the message size is large. Moreover, many coding techniques in coded caching are similar to the mechanism of the optimal scheme in [14] for the PIR problem. This implies that these two techniques can be combined to achieve synergetic gain on PIR capacity. To apply caching in PIR problem, I assume that the databases cannot identify the cached parts of messages. This assumption is reasonable when the caching takes place much earlier than the actual retrieval and databases do not have long term memory.

In Chapter 3, I consider the master's privacy in coded computation. In the system model, the workers have their own data, as well as the master. The workers' data constitutes a library of several files. The master should compute a function of its own data and a specific file in the library. The master's privacy implies that the workers' should not know which file in the library is desired by the user. I refer to this problem as private coded computation and propose an achievable scheme of private coded computation, namely private polynomial codes.

In Chapter 4, I consider the data security in coded computation, as well as the master's privacy. The system model is similar to that of private coded computation. The data security implies that the master should protect its own data against the workers. I refer to this problem as private secure coded computation and propose an achievable scheme, namely private secure polynomial codes.

In Chapter 5, I summarize my works and indicate the future directions of my works.

Chapter 2

Cache-aided Private Information Retrieval

2.1 Introduction

In this chapter, I consider the *private information retrieval* (PIR) problem in which a user has pre-stored data in its cache, for a non-colluding replication-based storage system. Since databases do not collude each other, each database cannot know what has been downloaded in other databases. To apply caching in PIR problem, I assume that the databases cannot identify the cached parts of messages. This assumption is reasonable when the caching takes place much earlier than the actual retrieval and databases do not have long term memory. I call this new PIR problem as cache-aided PIR. The motivation of caching in the PIR problem comes from the exploitation of cached parts of undesired messages as side information. This exploitation is available only when the databases do not know what has been cached, as I assumed.

2.2 System model

Consider a non-colluding replication-based storage system with N databases $\{\text{DB}_i\}_{i=1}^N$. A user desires one of M messages $\{\mathbf{B}_i\}_{i=1}^M$, which are replicated in each database. These M messages constitute a library $\mathbf{B}$. That is, $\mathbf{B} = \{\mathbf{B}_i\}_{i=1}^M$. The user sends queries to each database to retrieve the desired message. To ensure the user's privacy, the query to each database should be independent of the desired message index D, when the user wants $\mathbf{B}_D$.

I assume that all messages have equal size of F bits, i.e.,

$$H(\mathbf{B}_i) = F, \forall i \in [M].$$

Let F be much larger than the query size so that the amount of query can be ignored. Each message is divided into L fragments. The size of each fragment is F/L bits. I denote the lth fragment of ith message by $\mathbf{B}_{i,l}, l \in [L]$. The user has cache to store some parts of messages in advance. I denote the cached part of each message by $\mathbf{X}_i, i \in [M]$ and overall cache by $\mathbf{X} = \{\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_M\}$. Since the future demand of the user is unknown, I assume that the sizes of the cached part of the messages are the same, which means that

$$H(\mathbf{X}_i) = H(\mathbf{X})/M, \forall i \in [M].$$

I denote the size of cache for each message by

$$\gamma = H(\mathbf{X})/H(\mathbf{B}), 0 \le \gamma \le 1.$$

The query that the user sends to each database DB_i is denoted by Q_i, where $i \in [N]$. Each query Q_i can be expressed by

$$Q_i = [h_{1,1}, h_{1,2}, \ldots, h_{1,l}, h_{2,1}, \ldots, h_{M,L}],$$

where $h_{i,j} \in \mathbb{F}_2$. Note that the coefficient $h_{i,j}$ implies whether the jth fragment of ith message, or $\mathbf{B}_{i,j}$, is requested by the user. I denote these queries by $\mathbf{Q} = \{Q_1, Q_2, \ldots, Q_N\}$. Each databases transmits its response to the user according to the queries. This received data from DB_i is denoted by Y_i, where $i \in [N]$. The answer of each database can be expressed as

$$Y_i = \sum_{i \in [M], j \in [L]} h_{i,j} \mathbf{B}_{i,j}, \tag{2.1}$$

where $h_{i,j}, i \in [M], j \in [L]$ are the elements of Q_i. I denote them by $\mathbf{Y} = \{Y_1, Y_2, \ldots, Y_N\}$.

The retrieval of the desired message $\mathbf{B}_D$ can be information-theoretically expressed as

$$H(\mathbf{B}_D | \mathbf{Q}, \mathbf{Y}, \mathbf{X}) = 0. \ (correctness)$$

The privacy constraint can be expressed as

$$I(Q_i; D) = 0, \forall i \in [N]. \ (privacy)$$

I choose the amount of download for the retrieval as performance measure. I denote the amount of download for fixed cache size γ by $R(\gamma)$. The goal of cache-aided PIR is to minimize $R(\gamma)$, where the minimum amount of the download is denoted by $R^*(\gamma)$. The overall process of cache-aided PIR is described in Fig. 2.1.

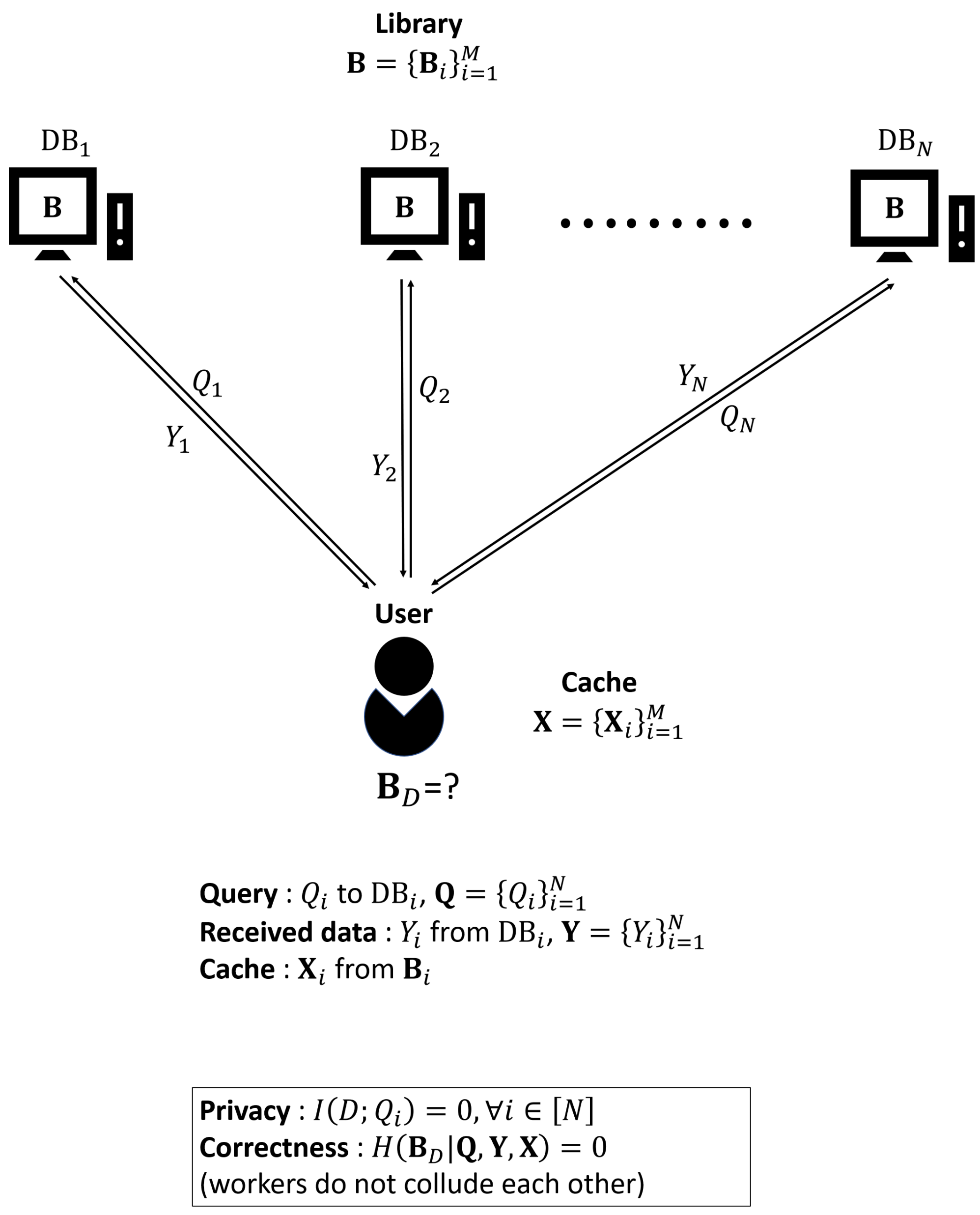

Library
$\mathbf{B} = \{\mathbf{B}_i\}_{i=1}^{M}$
DB_1
DB_2
DB_N
B
B
B
Q_1
Y_1
Q_2
Y_2
Y_N
Q_N
User
Cache
$\mathbf{X} = \{\mathbf{X}_i\}_{i=1}^{M}$
$\mathbf{B}_D$=?
Query : Q_i to DB_i, $\mathbf{Q} = \{Q_i\}_{i=1}^{N}$
Received data : Y_i from DB_i, $\mathbf{Y} = \{Y_i\}_{i=1}^{N}$
Cache : $\mathbf{X}_i$ from $\mathbf{B}_i$
Privacy : $I(D; Q_i) = 0, \forall i \in [N]$
Correctness : $H(\mathbf{B}_D | \mathbf{Q}, \mathbf{Y}, \mathbf{X}) = 0$
(workers do not collude each other)

Figure 2.1: The overall process of cache-aided PIR with M messages and N databases.

2.3 Main results

In this section, I state main results and its implications. First, I define several notations to easily show results. I denote the minimum amount of download for a replication-based storage system with M messages and N databases without cache by

$$R_0 = R^*(0) = 1 + \frac{1}{N} + \frac{1}{N^2} + \cdots + \frac{1}{N^{M-1}},$$

which is achieved in [14]. Note that the size of a message is normalized as 1. In addition, I define $\Delta(m, s)$ to briefly show main results and explain the proposed achievable scheme.

$$\Delta(m, s) = \frac{N^m - \sum_{n=0}^{s} \binom{m}{n}(N-1)^n}{(N-1)^{s+1}} N. \tag{2.2}$$

Theorem 1 (achievable scheme)**.** *For a replication-based storage system with M messages and N databases, where a user has cache which stores γ of each message, the achievable amount of download for the retrieval is*

$$R^*(\gamma) \leq$$
$$\begin{cases} \frac{\gamma}{\binom{M-2}{s-1}}\{\Delta(M,s) + \Delta(M-1,s-1)R_0\} + (1-\gamma)R_0, & \Theta(s) \leq \gamma < \Lambda(s), \\ (1-\gamma)\frac{\Delta(M,s)}{\Delta(M-1,s-1)}, & \Lambda(s) \leq \gamma < \Theta(s+1). \end{cases}$$

where $s \in [M-1]$ and

$$\begin{aligned} \Theta(s) &= \frac{\frac{\Delta(M,s-1)}{\Delta(M-1,s-2)} - R_0}{\frac{\Delta(M,s-1)}{\Delta(M-1,s-2)} + \frac{\Delta(M,s)}{\binom{M-2}{s-1}} - R_0\left(1 + \binom{M-2}{s-1}^{-1}\right)}, \\ \Lambda(s) &= \frac{1}{1 + \frac{\Delta(M-1,s-1)}{\binom{M-2}{s-1}}}, \end{aligned}$$

and I set $\Theta(1) = 0$ and $\Theta(M) = \infty$.

Proof. The proof of Theorem 1 will be given in Section 2.4. □

Theorem 2 (lower bound)**.** *For a replication-based storage system with M messages and N databases, where a user has cache which stores γ of each message, the minimum amount of download for the retrieval is lower bounded by*

$$R^*(\gamma) \geq 1 - \gamma. \tag{2.3}$$

Proof. Since the bound is obvious considering the existence of cache, I omit the proof, which is also trivial. □

Remark 1. From Theorem 1 and 2, it is observed that the proposed scheme is optimal, i.e., the performance gap between the achievable scheme and the lower bound is zero, when $\gamma \geq \Lambda(M-1)$. This is because $\Delta(M,s)$ equals to $\Delta(M-1,s-1)$ when $s = M-1$, thus $(1-\gamma)\frac{\Delta(M,s)}{\Delta(M-1,s-1)}$ becomes to $1-\gamma$.

Remark 2. The gaps between the proposed scheme and the lower bound for $(N,M) = (3,4), (3,50), (10,4)$ are depicted in Fig. 2.2, 2.3, and 2.4, respectively. As can be seen in Remark 1, the proposed scheme is optimal for large enough γ. Moreover, as seen in the figures, as N and M becomes larger, the gap is reduced.

Theorem 3 (tightness of achievable scheme)**.** *The achievable bound of cache-aided PIR given in Theorem 1 is within a multiplicative factor of 2 of lower bound of cache-aided PIR given in Theorem 2, i.e.,*

$$\frac{R_{ac}}{R_{lb}} \leq 2, \tag{2.4}$$

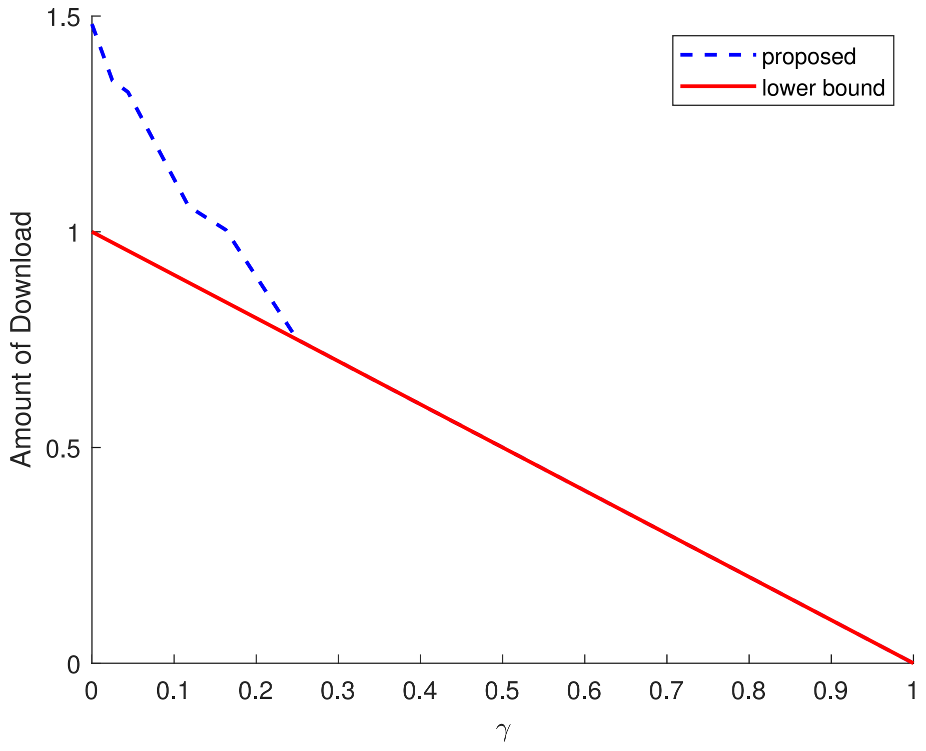


Figure 2.2: The performance gap between the proposed scheme and the lower bound for $N = 3$ and $M = 4$

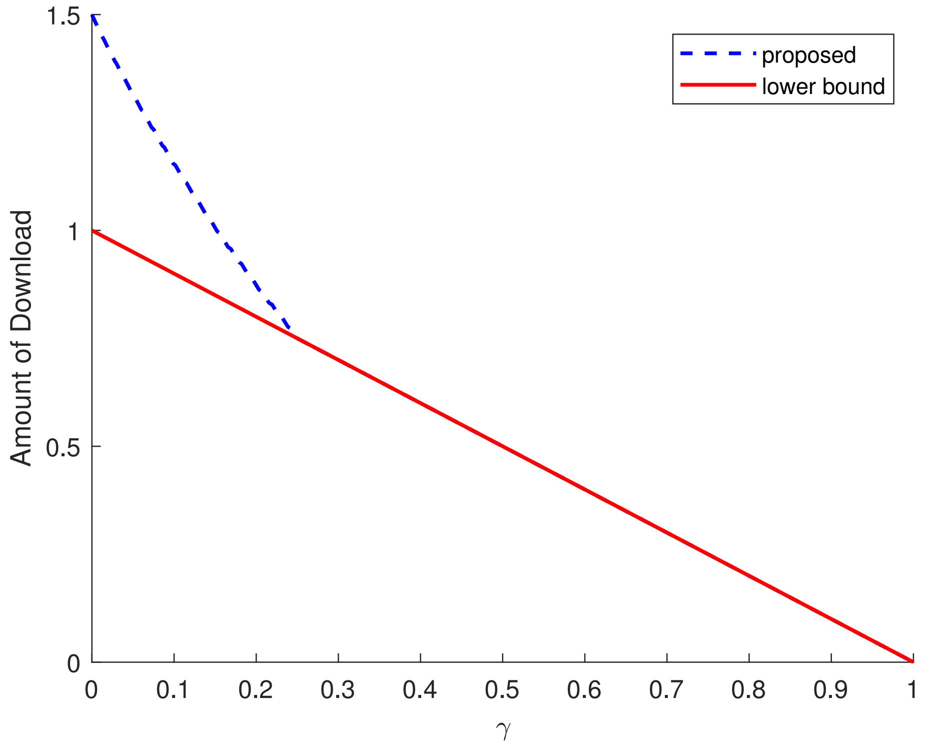


Figure 2.3: The performance gap between the proposed scheme and the lower bound for $N = 3$ and $M = 50$

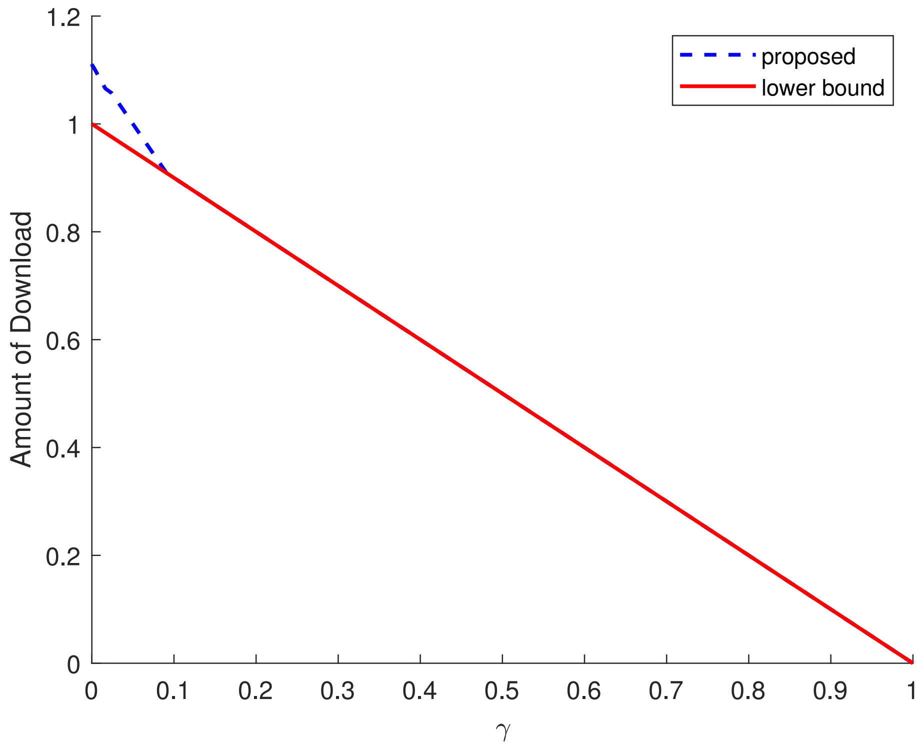


Figure 2.4: The performance gap between the proposed scheme and the lower bound for $N = 10$ and $M = 4$

where R_{lb} is an lower bound of the amount download in cache-aided PIR given in Theorem 2 and R_{ac} is an achievable bound of the amount of download in cache-aided PIR given in Theorem 1.

Proof. The bound is given in Section 2.5 □

2.4 Achievable Scheme

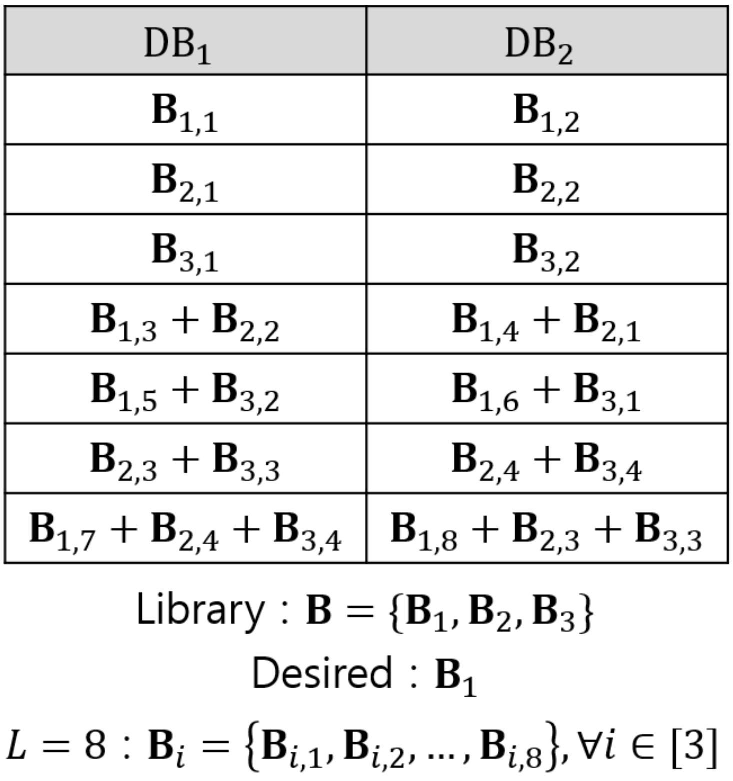
In this section, I propose an achievable scheme of cache-aided PIR. As in [14], the proposed scheme is based on three idea :

- *database symmetry*
- *message symmetry*
- *exploiting the undesired messages as side information*

The proposed scheme consists of two phases. The first phase is cache-assisted phase. The purpose of this phase is to exploit the cached parts of undesired messages as side information. The second phase is cacheless phase. At the second phase, without cache, the user downloads the remaining parts of desired message using the scheme in [14].

2.4.1 Cacheless phase

First, let me explain the second phase, which is just same as the scheme in [14]. As in [14], I set $L = N^M$. For example, I consider a storage system with 2 databases DB_1 and DB_2, and 3 messages $\mathbf{B}_1$, $\mathbf{B}_2$ and $\mathbf{B}_3$, and L should be 8. Let $\mathbf{B}_1$ be the desired



DB_1	DB_2
$\mathbf{B}_{1,1}$	$\mathbf{B}_{1,2}$
$\mathbf{B}_{2,1}$	$\mathbf{B}_{2,2}$
$\mathbf{B}_{3,1}$	$\mathbf{B}_{3,2}$
$\mathbf{B}_{1,3} + \mathbf{B}_{2,2}$	$\mathbf{B}_{1,4} + \mathbf{B}_{2,1}$
$\mathbf{B}_{1,5} + \mathbf{B}_{3,2}$	$\mathbf{B}_{1,6} + \mathbf{B}_{3,1}$
$\mathbf{B}_{2,3} + \mathbf{B}_{3,3}$	$\mathbf{B}_{2,4} + \mathbf{B}_{3,4}$
$\mathbf{B}_{1,7} + \mathbf{B}_{2,4} + \mathbf{B}_{3,4}$	$\mathbf{B}_{1,8} + \mathbf{B}_{2,3} + \mathbf{B}_{3,3}$

Library : $\mathbf{B} = \{\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3\}$

Desired : $\mathbf{B}_1$

$L = 8 : \mathbf{B}_i = \left\{\mathbf{B}_{i,1}, \mathbf{B}_{i,2}, \dots, \mathbf{B}_{i,8}\right\}, \forall i \in [3]$

Figure 2.5: The process of cacheless download for $N = 2$ and $M = 3$.

message. At first, the user sends $Q_1 = \mathbf{e}_1$ to the first database, where $\mathbf{e}_i$ denotes the ith unit vector, $\mathbf{e}_i \in \mathbb{F}_2^{ML\times 1}$. After that, from (2.1), DB$_1$ transmits its answer $Y_1 = \mathbf{B}_{1,1}$. By the database symmetry, DB$_2$ also transmits a fragment of the desired message, $Y_2 = \mathbf{B}_{1,2}$ where $Q_2 = \mathbf{e}_2$. Next, by the message symmetry, the undesired messages $\mathbf{B}_2$ and $\mathbf{B}_3$ should also be downloaded. Therefore, DB$_1$ transmits

$$Y_1 = \mathbf{B}_{2,1}\ (Q_1 = \mathbf{e}_{L+1}),$$

$$Y_1 = \mathbf{B}_{3,1}\ (Q_1 = \mathbf{e}_{2L+1}),$$

and DB$_2$ transmits $\mathbf{B}_{2,2}$ and $\mathbf{B}_{3,2}$, respectively.

After that, since the undesired fragments were downloaded, the user exploits these undesired fragments $\mathbf{B}_{2,1}$, $\mathbf{B}_{2,2}$, $\mathbf{B}_{3,1}$, and $\mathbf{B}_{3,2}$ as side information. Therefore, DB$_1$ transmits

$$Y_1 = \mathbf{B}_{1,3} + \mathbf{B}_{2,2}\ (Q_1 = \mathbf{e}_3 + \mathbf{e}_{L+2}),$$

$$Y_1 = \mathbf{B}_{1,5} + \mathbf{B}_{3,2}\ (Q_1 = \mathbf{e}_5 + \mathbf{e}_{2L+2})$$

Symmetrically, DB$_2$ transmits $\mathbf{B}_{1,4} + \mathbf{B}_{2,1}$ and $\mathbf{B}_{1,6} + \mathbf{B}_{3,1}$. Note that $\mathbf{B}_{2,1}$ and $\mathbf{B}_{3,1}$ are not exploited in DB$_1$. Since $\mathbf{B}_{2,1}$ and $\mathbf{B}_{3,1}$ were downloaded, re-download of $\mathbf{B}_{2,1}$ and $\mathbf{B}_{3,1}$ as side information enables DB$_1$ to identify that they are not desired by the user.

At the next step, the message symmetry is reconsidered because of the message asymmetry caused by the previous exploitation. Therefore, equations that are comprised of only undesired fragments should be downloaded. Specifically, DB$_1$ transmits the equation

$$Y_1 = \mathbf{B}_{2,3} + \mathbf{B}_{3,3} \, (Q_1 = \mathbf{e}_{L+3} + \mathbf{e}_{2L+3}),$$

and DB$_2$ transmits

$$Y_2 = \mathbf{B}_{2,4} + \mathbf{B}_{3,4} \, (Q_2 = \mathbf{e}_{L+4} + \mathbf{e}_{2L+4}).$$

The additional equations consisting of only undesired fragments due to message symmetrization are exploited as side information. Therefore, DB$_1$ transmits

$$Y_1 = \mathbf{B}_{1,7} + \mathbf{B}_{2,4} + \mathbf{B}_{3,4} \, (Q_1 = \mathbf{e}_7 + \mathbf{e}_{L+4} + \mathbf{e}_{2L+4}),$$

and DB$_2$ transmits

$$Y_2 = \mathbf{B}_{1,8} + \mathbf{B}_{2,3} + \mathbf{B}_{3,3} \, (Q_2 = \mathbf{e}_8 + \mathbf{e}_{L+3} + \mathbf{e}_{2L+3}).$$

Since all of the 3 messages are summed into same equations, no more message asymmetry occurs. Therefore, the messages were downloaded symmetrically and the user's privacy is protected. Since all of $L = 8$ fragments of the desired messages were downloaded, the retrieval is completed without further download. The overall process is depicted in Fig. 2.5.

The general algorithm of cacheless phase is described in Algorithm 1 for each database. The overall process across databases is done by applying this algorithm to each database.

In summary, after the first download, the cacheless phase continues iteratively between the message symmetry and the exploitation. This iteration ends at the exploitation step where all the other $M - 1$ undesired messages are added to the new desired

Algorithm 1 General algorithm of cacheless phase

```
for $i = 0$ to $M - 1$ do
    if $i = 0$ then
        Step 1. Download one desired fragment
    else
        Step 2. Symmetrize the messages
        Step 3. Exploit the undesired fragments downloaded from the other $N - 1$
        databases in previous symmetrization as side information
    end if
end for
```

message fragments. In this step, since all of the M message fragments are summed into one equation, no more message asymmetry occurs. Since I set L as N^M, the retrieval is completed.

2.4.2 Cache-assisted phase

In cache-assisted phase, the cached parts of undesired messages can be exploited as side information at the beginning of the phase. That is, the cache-assisted phase starts with Step 3 in a general algorithm of cacheless phase and continues iteratively. Note that after the iteration is over, the cache-assisted phase ends without repetition since there remains no more cached parts of undesired messages to be exploited.

In the initial exploitation, the number of undesired fragments added to the new desired fragment should be considered. I denote this number by s, whose range is from

1 to $M-1$. Recall that the exploitation step in cacheless phase starts with $s=0$, and ends with $s=M-1$. Therefore, s implicitly means the starting position of iteration. In other words, the cache-assisted phase skips the first s iterations out of $K-1$ iterations compared to the cacheless phase. As a result, the number of iteration in cache-assisted phase is $M-1-s$. I denote the amount of download of the proposed scheme for given γ and s by $R(\gamma, s)$.

The example of cache-assisted phase is given in Fig. 2.6, where $N=M=3$ and $s=1$. I also consider that $\gamma = 2/L$. That is, in the user's cache, 2 fragments of each message are stored. I set L as 53, which is matched to the cache-assisted algorithm and the cacheless algorithm afterward. The desired message is assumed to be $\mathbf{B}_1$. At first, the cached undesired fragments $\mathbf{B}_{2,1}$, $\mathbf{B}_{2,2}$, $\mathbf{B}_{3,1}$, and $\mathbf{B}_{3,2}$ are exploited as side information in all databases. Since $s=1$, each fragment is added to the new desired fragment, respectively. Thus DB$_1$ transmits

$$\mathbf{B}_{1,3} + \mathbf{B}_{2,1} \; (Q_1 = \mathbf{e}_{L+1}),$$

$$\mathbf{B}_{1,6} + \mathbf{B}_{2,2} \; (Q_1 = \mathbf{e}_{L+2}),$$

$$\mathbf{B}_{1,9} + \mathbf{B}_{3,1} \; (Q_1 = \mathbf{e}_{2L+1}),$$

$$\mathbf{B}_{1,12} + \mathbf{B}_{3,2} \; (Q_1 = \mathbf{e}_{2L+2}).$$

The databases DB$_2$ and DB$_3$ transmit symmetrically. The next step is exploitation of previous equations. As explained in the cacheless phase, previous equations of DB$_1$ cannot be exploited for itself, which are the same for DB$_2$ and DB$_3$. After the exploitation, since all of the 3 messages are summed into each equation, no more message

User's cache

$\{\mathbf{B}_{1,1}, \mathbf{B}_{1,2}, \mathbf{B}_{2,1}, \mathbf{B}_{2,2}, \mathbf{B}_{3,1}, \mathbf{B}_{3,2}\}$

	DB_1	DB_2	DB_3
Exploitation	$\mathbf{B}_{1,3} + \mathbf{B}_{2,1}$	$\mathbf{B}_{1,4} + \mathbf{B}_{2,1}$	$\mathbf{B}_{1,5} + \mathbf{B}_{2,1}$
	$\mathbf{B}_{1,6} + \mathbf{B}_{2,2}$	$\mathbf{B}_{1,7} + \mathbf{B}_{2,2}$	$\mathbf{B}_{1,8} + \mathbf{B}_{2,2}$
	$\mathbf{B}_{1,9} + \mathbf{B}_{3,1}$	$\mathbf{B}_{1,10} + \mathbf{B}_{3,1}$	$\mathbf{B}_{1,11} + \mathbf{B}_{3,1}$
	$\mathbf{B}_{1,12} + \mathbf{B}_{3,2}$	$\mathbf{B}_{1,13} + \mathbf{B}_{3,2}$	$\mathbf{B}_{1,14} + \mathbf{B}_{3,2}$
Message Symmetry	$\mathbf{B}_{2,3} + \mathbf{B}_{3,3}$	$\mathbf{B}_{2,4} + \mathbf{B}_{3,4}$	$\mathbf{B}_{2,5} + \mathbf{B}_{3,5}$
	$\mathbf{B}_{2,6} + \mathbf{B}_{3,6}$	$\mathbf{B}_{2,7} + \mathbf{B}_{3,7}$	$\mathbf{B}_{2,8} + \mathbf{B}_{3,8}$
Exploitation	$\mathbf{B}_{1,15} + \mathbf{B}_{2,4} + \mathbf{B}_{3,4}$	$\mathbf{B}_{1,16} + \mathbf{B}_{2,3} + \mathbf{B}_{3,3}$	$\mathbf{B}_{1,17} + \mathbf{B}_{2,3} + \mathbf{B}_{3,3}$
	$\mathbf{B}_{1,18} + \mathbf{B}_{2,5} + \mathbf{B}_{3,5}$	$\mathbf{B}_{1,19} + \mathbf{B}_{2,6} + \mathbf{B}_{3,6}$	$\mathbf{B}_{1,20} + \mathbf{B}_{2,6} + \mathbf{B}_{3,6}$
	$\mathbf{B}_{1,21} + \mathbf{B}_{2,7} + \mathbf{B}_{3,7}$	$\mathbf{B}_{1,22} + \mathbf{B}_{2,5} + \mathbf{B}_{3,5}$	$\mathbf{B}_{1,23} + \mathbf{B}_{2,4} + \mathbf{B}_{3,4}$
	$\mathbf{B}_{1,24} + \mathbf{B}_{2,8} + \mathbf{B}_{3,8}$	$\mathbf{B}_{1,25} + \mathbf{B}_{2,8} + \mathbf{B}_{3,8}$	$\mathbf{B}_{1,26} + \mathbf{B}_{2,7} + \mathbf{B}_{3,7}$

Library : $\mathbf{B} = \{\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3\}$

Desired : $\mathbf{B}_1$

$L = 26 : \mathbf{B}_i = \{\mathbf{B}_{i,1}, \mathbf{B}_{i,1}, \ldots, \mathbf{B}_{i,26}\}, \forall i \in [3]$

Figure 2.6: The process of cache-assisted phase for $N = 3$ and $M = 3$.

asymmetry occurs. Therefore, the messages were downloaded symmetrically. Thus, the user's privacy is protected as in the cacheless phase. Since all of the cached undesired fragments are exploited, the whole process of cache-assisted phase is over, and the remaining parts of $\mathbf{B}_1$ are downloaded in the cacheless phase. Since the number of downloaded fragments of desired message in cacheless phase is N^M, which is 27 in the above example, the value of L is adjusted to $26 + 27 = 53$.

The general algorithm of cache-assisted phase is described in Algorithm 2 for each database, as in the cacheless phase. Due to the database symmetry, the overall process across databases is done by applying this algorithm to each database. Note that the message symmetry step and exploitation step is basically same as cacheless phase, which is same as [14]. Since the user's privacy is protected for the algorithm in [14], it's obvious that the privacy is protected for proposed algorithm as well.

2.4.3 Cache-aided PIR

Until now, I have assumed the retrieval continues to the cacheless phase. However, if γ is large enough, the retrieval can be done without the cacheless phase. In this case, not all of the cached parts of undesired messages are exploited as side information. The threshold value of γ depends on the value of s, and I denote this threshold by $\Lambda(s)$. If $\gamma < \Lambda(s)$, the cacheless phase follows the cache-assisted phase. On the other hand, if $\gamma \geq \Lambda(s)$, the retrieval is done with cache-assisted phase only. Therefore, for given s, there are two different functions for $R(\gamma, s)$. One is for $\gamma < \Lambda(s)$ and the other is for

Algorithm 2 General algorithm of cache-assisted phase

```
for $i = s$ to $M - 1$ do
    if $i = s$ then
        Step 1. Exploit the cached undesired fragments as side information by adding
        $s$ undesired fragments to the new desired fragment.
    else
        Step 2. Symmetrize the messages
        Step 3. Exploit the undesired fragments downloaded from the other $N - 1$
        databases in previous symmetrization as side information
    end if
end for
```

$\gamma \geq \Lambda(s)$.

Now, I derive $R(\gamma, s)$. For fixed s, there are $\binom{M-1}{s}$ kinds of combination of undesired messages to be exploited as side information. For a specific undesired message, which is cached as much as γ, the number of kinds of combination is $\binom{M-2}{s-1}$. If I assume that each combination is exploited only once, the size of each fragment should be $\gamma/\binom{M-2}{s-1}$ and therefore L should be $F\binom{M-2}{s-1}/\gamma$, which is dependent on γ. Consequently, the amount of downloaded fragments of desired message in initial exploitation of cache-assisted phase is $\gamma\binom{M-1}{s}/\binom{M-2}{s-1}$.

The message symmetry process should be followed with $\binom{M-1}{s+1}$ combinations of undesired messages, whose amount is $\gamma\binom{M-1}{s+1}/\binom{M-2}{s-1}$ so that the amount of combinations consisting of $s+1$ elements is $\gamma\binom{M}{s+1}/\binom{M-2}{s-1}$. Note that in the message symmetry process, only undesired messages are downloaded. Later process is iterated just as same as the PIR without cache. For computing the amount of download, note that the number of elements in each combination is incremented by 1 and the number of downloaded fragments are multiplied by $N-1$ at every exploitation step except the initial exploitation. Therefore, if I denote $V(s)$ as the amount of fragments from all of N databases including the desired message in cache-assisted phase for given s, and $T(s)$ as the total amount of fragments from N databases in cache-assisted phase for given s, I get

$$\begin{aligned} V(s) &= \frac{\gamma N}{\binom{M-2}{s-1}} \sum_{n=0}^{M-1-s} (N-1)^n \binom{M-1}{s+n} \\ &= \frac{\gamma}{\binom{M-2}{s-1}} \Delta(M-1, s-1), \end{aligned} \tag{2.5}$$

$$\begin{aligned} T(s) &= \frac{\gamma N}{\binom{M-2}{s-1}} \sum_{n=0}^{M-1-s} (N-1)^n \binom{M}{s+1+n} \\ &= \frac{\gamma}{\binom{M-2}{s-1}} \Delta(M, s). \end{aligned} \tag{2.6}$$

Since the remaining parts of the desired message after the cache-assisted phase is $1-\gamma-V(s)$, the threshold value $\Lambda(s)$ should satisfy the equation, $1-\Lambda(s)-V(s)=0$.

[1]The size of a fragment should be adjusted so that the retrieval can be done after $M-1$ iterations in cacheless phase regardless of the number of repetitions. If the retrieval ends in the middle of the iterations, the user's privacy cannot be protected.

Algorithm 3 General algorithm of cache-aided PIR

```
for $i = 1$ to $M - 1$ do
    if $\Theta(i) \leq \gamma < \Lambda(i)$ then
        Step 1. Cache-assisted phase for given $s = i$
        Step 2. Cacheless phase for remainder[1]
    else if $\Lambda(i) \leq m < \Theta(i+1)$ then
        Step 1. Cache-assisted phase for given $s = i$
    end if
end for
```

For $\gamma < \Lambda(s)$, the amount of download from the cache-assisted phase is $T(s)$ and that of the cacheless phase is $R_0(1 - \gamma - V(s))$. Therefore the total amount of download is $T(s) + R_0(1 - \gamma - V(s))$. Since γ is normalized size of cached $\mathbf{B}_D$, 1 denotes the size of $\mathbf{B}_D$. For $\gamma \geq \Lambda(s)$, the amount of download from the cache-assisted phase is given by

$$\frac{1-\gamma}{V(s)/T(s)}.$$

Therefore, $R(\gamma, s)$ is given as

$$R(\gamma, s) = \begin{cases} T(s) + R_0(1 - \gamma - V(s)), & \gamma < \Lambda(s), \\ (1-\gamma)\frac{T(s)}{V(s)}, & \gamma \geq \Lambda(s). \end{cases} \tag{2.7}$$

Now, I can compute $R(\gamma, s)$ for every s and γ. If I denote the amount of download of the proposed scheme for given γ by $R_{ac}(\gamma)$, $R_{ac}(\gamma)$ is the minimum among

$R(\gamma,1), R(\gamma,2), \cdots, R(\gamma,M-1)$. That is,

$$R_{ac}(\gamma) = \min(R(\gamma,1), R(\gamma,2), \cdots, R(\gamma,M-1)).$$

The value of s that minimizes $R(\gamma,s)$ grows gradually as γ increases. Specifically, for $0 \le \gamma < \Lambda(1)$, $R(\gamma,s)$ is minimized when $s=1$ and s increments by 1 as γ grows. When γ becomes greater than γ_{M-1}, the value of s is kept at $M-1$. In addition, there is a point where $R(\gamma,s-1)$ becomes larger than $R(\gamma,s)$ for every section in $\Lambda(s-1) \le \gamma < \Lambda(s)$. If I denote this point by $\Theta(s)$, then for $\Lambda(s) \le \gamma < \Theta(s)$, $s-1$ is the argument of minimum. However, for $\Theta(s) \le \gamma < \Lambda(s)$, s becomes the argument of minimum. This $\Theta(s)$ can be derived from

$$(1-\Theta(s))\frac{T(s-1)}{V(s-1)} = T(s) + 1 - \Theta(s) - V(s)R_0.$$

Since the above equation holds when $\gamma = \Theta(s)$, the following holds.

$$\begin{aligned} V(s) &= \frac{\Theta(s)}{\binom{M-2}{s-1}}\Delta(M-1,s-1), \\ T(s) &= \frac{\Theta(s)}{\binom{M-2}{s-1}}\Delta(M,s), \\ V(s-1) &= \frac{\Theta(s)}{\binom{M-2}{s-2}}\Delta(M-1,s-2), \\ T(s-1) &= \frac{\Theta(s)}{\binom{M-2}{s-2}}\Delta(M,s-1). \end{aligned}$$

As a result, the above equation becomes

$$\begin{aligned} &(1-\Theta(s))\frac{\Delta(M-1,s-1)}{\Delta(M-1,s-2)} \\ &= \frac{\Theta(s)}{\binom{M-2}{s-1}}\Delta(M,s) + 1 - \Theta(s) - R_0\frac{\Theta(s)}{\binom{M-2}{s-1}}\Delta(M-1,s-1). \end{aligned}$$

The above equation is equivalent to

$$\frac{\Delta(M-1,s-1)}{\Delta(M-1,s-2)}-1$$
$$=\Theta(s)\{\frac{\Delta(M-1,s-1)}{\Delta(M-1,s-2)}+\frac{\Delta(M,s)}{\binom{M-2}{s-1}}-1-R_0\frac{\Delta(M-1,s-1)}{\binom{M-2}{s-1}}\}.$$

Therefore, the parameter $\Theta(s)$ is given as follows.

$$\Theta(s)=\frac{\frac{\Delta(M,s-1)}{\Delta(M-1,s-2)}-R_0}{\frac{\Delta(M,s-1)}{\Delta(M-1,s-2)}+\frac{\Delta(M,s)}{\binom{M-2}{s-1}}-R_0\left(1+\binom{M-2}{s-1}^{-1}\right)}.$$

Consequently, the amount of download of the scheme can be summarized as in Theorem 1 and the general algorithm of cache-aided PIR is summarized in Algorithm 3.

2.5 Tightness of achievable scheme

In this section, I show that the performance of achievable scheme is within a constant multiplicative factor of 2 of lower bound of the amount of download in cache-aided PIR, which is given in Theorem 2. Specifically, I prove that the amount of download of achievable scheme for any s, which is given in (2.7), is within a constant multiplicative factor of 2 of lower bound of the amount of download in cache-aided PIR given in (2.3). For $\gamma > \Lambda(s)$, the ratio of lower bound to the achievable scheme is given by

$$\frac{T(s)}{V(s)}=\frac{V(s)+T(s)-V(s)}{V(s)}.$$

Note that $T(s)-V(s)$ implies the amount of downloaded undesired messages, which is strictly smaller than $V(s)$. Considering cache-aided PIR algorithm explained in Section 2.4.2, the undesired messages are downloaded when message symmetry is needed. Since all of these downloaded undesired messages are exploited as side information in each database, the amount of downloaded desired messages is strictly larger. Therefore, the following holds.

$$\frac{V(s)+T(s)-V(s)}{V(s)} \leq 2.$$

For $\gamma \leq \Lambda(s)$, the ratio of lower bound to the achievable scheme is given by

$$R_0 + \frac{\gamma}{1-\gamma}(\frac{T(s)}{\gamma} + \frac{R_0 V(s)}{\gamma}).$$

Note that the term $\frac{T(s)}{\gamma} + \frac{R_0 V_s}{\gamma}$ is independent to γ, considering (2.5) and (2.6). For $\frac{T(s)}{\gamma} + \frac{R_0 V_s}{\gamma} < 0$, the ratio is maximum at $\gamma = 0$, which equals to

$$R_0 = 1 + 1/N + \cdots + 1/N^{M-1}.$$

For fixed N, R_0 is maximum at $M = \infty$, which equals to $(1-1/N)^{-1}$. Therefore, maximum of R_0 is 2 when $N = 2$ and $M = \infty$

For $\frac{T(s)}{\gamma} + \frac{R_0 V_s}{\gamma} > 0$, the ratio is maximum at $\gamma = \Lambda(s)$, which equals to $T(s)/V(s)$. This results from $1-\Lambda(s)-V(s) = 0$. As shown before, $T(s)/V(s) < 2$. Consequently, the amount of download of achievable scheme is within a constant multiplicative factor of 2 of lower bound of the amount of download cache-aided PIR.

2.6 Conclusion and follow-up works

In this chapter, I analyzed a cache-aided PIR problem with replication-based storage systems so that more efficient retrieval is possible. Specifically, I proposed an achievable scheme for cache-aided PIR which is proven to be optimal for a large enough cache size. However, the gap between the achievable scheme and the lower bound exists for a small cache size, thus future work could be to propose an optimal scheme of the cache-aided PIR problem for an arbitrary cache size.

After my work, some follow-up works were proposed. First, there are works directly related to the cache-aided PIR. In [15], unlike my assumption, the databases are assumed to know what has been cached by the user. In this work, it was proven that the performance of cache-aided PIR in this case is as same as the PIR without cache, except the reduced amount of download due to the cached desired message itself. In [16], improved achievable scheme and lower bound of cache-aided PIR were proposed, whose scheme is based on mine and uses memory-sharing. Second, there are works related to the heterogeneous caching. In [17]- [18], the user caches whole M messages out of $M-1$ undesired messages, excluding the others.

Chapter 3

Private Coded Computation

3.1 Introduction

In this chapter, for a matrix multiplication, a new variation of coded computation that ensures the master's privacy from the workers, which is referred to as *private coded computation*, is proposed. In this problem, the master should multiply two matrices $\mathbf{A}$ and $\mathbf{B}_D$ where the matrix $\mathbf{B}_D$ is an element in a library $\mathbf{B}$ that consists of M matrices $\{\mathbf{B}_k\}_{k=1}^{M}$. This library $\mathbf{B}$ is exclusively shared by external workers whereas the matrix $\mathbf{A}$ is exclusively owned by the master. The master encodes its matrix $\mathbf{A}$ with an encoding function g and sends encoded data to the workers. The master also sends queries that request each worker to encode the library $\mathbf{B}$ with its encoding function h and to multiply $g(\mathbf{A})$ and $h(\mathbf{B})$. After the master recovers the matrix multiplication $\mathbf{A}\mathbf{B}_D$ from the returned multiplications, the workers must not be able to identify that $\mathbf{B}_D$ is desired by the master, which would imply that the master's privacy on $\mathbf{B}_D$

is ensured. I propose an achievable scheme of private coded computation, which is referred to as *private polynomial codes* (PPC).

As a motivating example of the private coded computation, I may consider a user who employs an artificial intelligence (AI) assistant, e.g. Google Assistant or Siri, with its mobile phone/device. I assume that the user requests a recommendation from an AI assistant of an item which is included in one of several categories, e.g. movies, games, restaurants, and so on. The user stores its preference data and wants a recommendation of an item in a specific category. When requested, the assistant encodes the user's preference data and sends encoded data to distributed workers for computing a function of preference data and that of desired category, from which the item is recommended. I assume that the user deletes the recommendation service usage record right after the recommendation so that the AI assistant does not identify the user's recommendation service usage pattern.

Generally, the user will use this recommendation service according to its life style. Therefore, if the workers obtain the user's recommendation service usage records and identify the desired category every time, the user's privacy on its life style is infringed by the workers. I remark that this privacy infringement is related to the categories, not the user's preference data. That is, encrypting the user's preference data cannot ensure the user's privacy on the life style. In order to ensure the user's privacy, the workers should not know that a particular category is desired by a user, which motivates the private coded computation. This motivating example is depicted in Fig. 3.1.

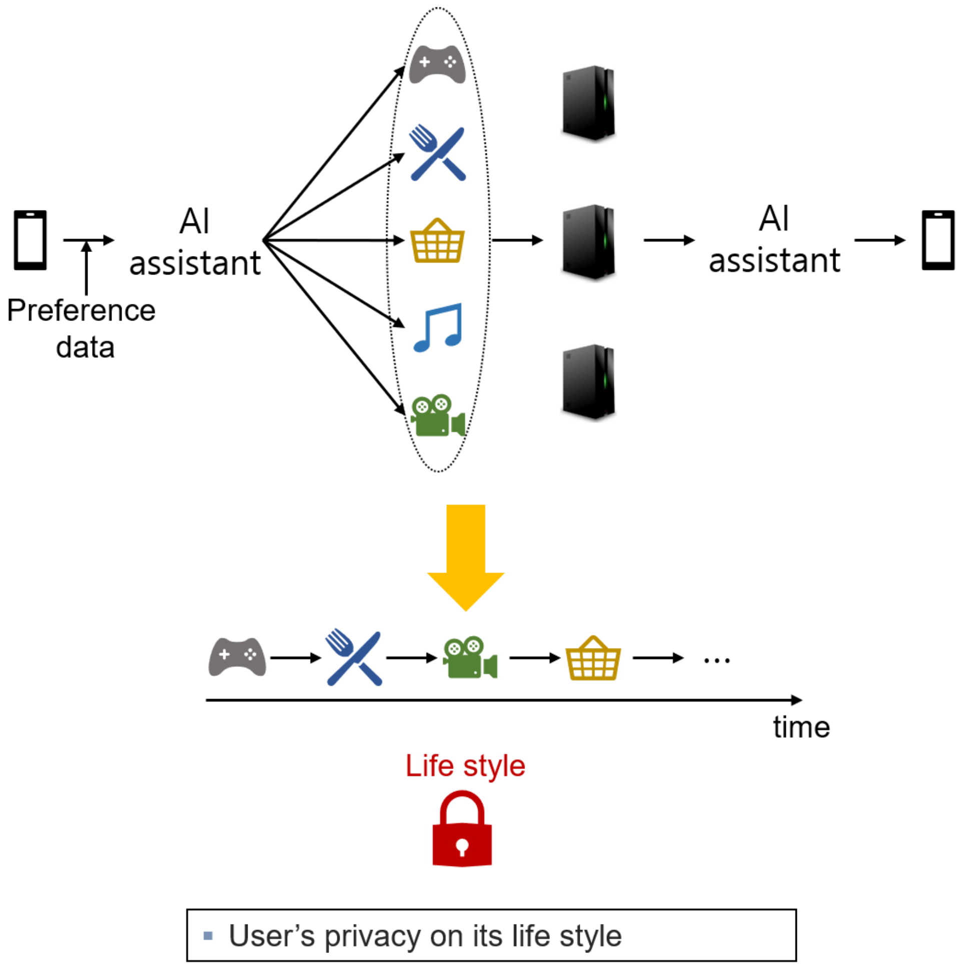


Figure 3.1: Motivating example of private coded computation.

Related works : The private coded computation which is dealt with in this chapter only considers the master's privacy, but not the data security on $\mathbf{A}$. On the other hand, *secure coded computation* [41] - [45] only considers the data security, but not the master's privacy. Generally, in the secure coded computation for matrix multiplication, the master has two matrices $\mathbf{A}$ and $\mathbf{B}$, and wants to compute $\mathbf{AB}$ without revealing any information about both $\mathbf{A}$ and $\mathbf{B}$ to the workers. Secure coded computation was studied mainly in computer science and machine learning fields [41] - [45]. Since the workers do not share a library unlike the private coded computation, the master's privacy on a specific element in the library cannot be considered in the secure coded computation. Note that, in *Lagrange coded computing* [44], the master's privacy was considered for protecting the content of master's data against colluding workers. That is, the master's privacy of [44] is different from that of private coded computation, where the master does not need to protect its data $\mathbf{A}$. In the secure coded computation for matrix multiplication, *gap additive secure polynomial* (GASP) codes [45] achieved improved download rate compared to the previous work.

The master's privacy on desired data $\mathbf{B}_D$ in the library $\mathbf{B}$ was previously considered in *private information retrieval* (PIR) problem [46]. In the conventional PIR problem, the master should privately download a specific element in the data library. Recently, an optimal PIR scheme that minimizes the amount of download was proposed [47]. After this work, several follow-up works that considered variations of PIR problem were proposed [48] - [54]. In particular, *robust PIR* (RPIR) was proposed where some of the databases may not respond to the master [53]. In [53], an optimal RPIR scheme that minimizes the amount of download was proposed. I compare my proposed scheme with this optimal RPIR scheme in terms of computation time and communication load which will be defined in Section 3.2.

In [54], *X-secure T-private information retrieval* (XSTPIR) was also studied. In XSTPIR, the data library is encoded and stored in databases. To obtain information about the stored data, up to X databases may collude. However, for data security, the colluding databases should not obtain any information about the stored data. Furthermore, the master should conceal the index D of desired data $\mathbf{B}_D$ against T colluding workers. In [54], the asymptotic capacity of XSTPIR was characterized, and an achievable scheme based on cross subspace alignment was proposed.

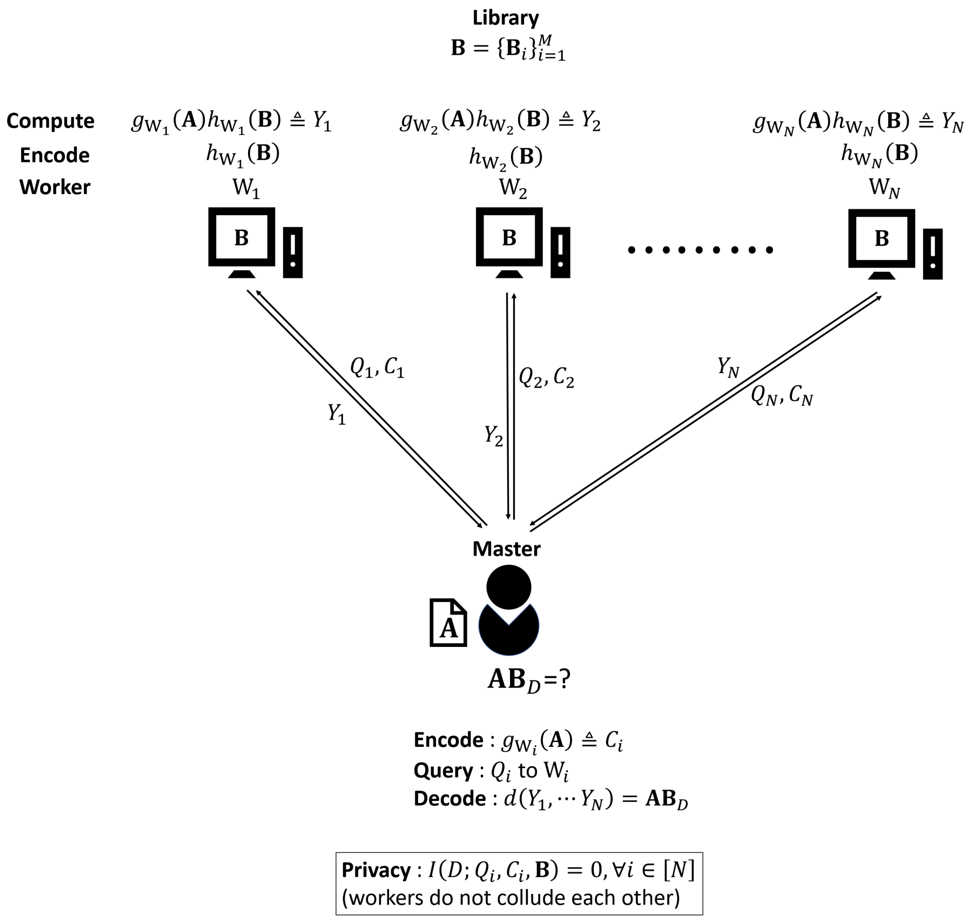
As a variation of PIR, *private computation* (PC) was proposed. In PC, the master wants to privately compute a function of the library $\mathbf{B}$. For example, the function could be the linear combinations of the elements in the library [56] - [59]. In these works, the coefficients of linear combinations should be concealed. As a nonlinear function for PC, polynomial computation was considered in [60].

If the function in PC is a linear combination of elements in $\mathbf{B}$ whose coefficients are all zero except one, the PC problem reduces to an ordinary PIR problem. That is, the PIR problem is a special case of PC problem. Moreover, since a matrix multiplication is a collection of linear combinations, PC schemes for linear combination can be applied to private coded computation. That is, by consecutively applying PC schemes and assigning each element of $\mathbf{A}$ as a coefficient of the desired matrix $\mathbf{B}_D$ for each iteration, the master can privately recover $\mathbf{A}\mathbf{B}_D$. Moreover, the master's privacy considered in PC problem is stronger than that of private coded computation. That is, for a linear combination of elements in $\mathbf{B}$, in PC problem, all of the coefficients of linear combination should be concealed, whereas the coefficient for the desired matrix should be concealed in private coded computation. As a result, private coded computation is another special case of PC problem. Therefore, I compare my proposed scheme with some PC schemes in terms of computation time and communication load. Specifically, I choose the PC schemes in [54] and [60]. Note that the scheme in [54] can be extended to PC problem.

3.2 System model

In this section, I describe a system model of private coded computation. There is a master which has its own matrix $\mathbf{A}$ in a vector space $\mathbb{V}_1$ over a field $\mathbb{F}$. There are also N external workers $\{\mathrm{W}_i\}_{i=1}^N$ which share a library $\mathbf{B}$ which consists of M different matrices $\{\mathbf{B}_k\}_{k=1}^M$. Each matrix $\mathbf{B}_k$ is an element in a vector space $\mathbb{V}_2$ over the same field $\mathbb{F}$. The master would like to compute $\mathbf{A}\mathbf{B}_D$, where $D \in [M]$. I assume that the workers are identical and do not collude with each other. That is, their computation and communication capabilities are identical, and each worker does not know the information exchanged between the master and the other workers.

The overall process is depicted in Fig. 3.2. First, the master encodes its own matrix



Library
$\mathbf{B} = \{\mathbf{B}_i\}_{i=1}^{M}$
Compute
Encode
Worker
$g_{\mathrm{W}_1}(\mathbf{A})h_{\mathrm{W}_1}(\mathbf{B}) \triangleq Y_1$
$h_{\mathrm{W}_1}(\mathbf{B})$
W_1
$g_{\mathrm{W}_2}(\mathbf{A})h_{\mathrm{W}_2}(\mathbf{B}) \triangleq Y_2$
$h_{\mathrm{W}_2}(\mathbf{B})$
W_2
$g_{\mathrm{W}_N}(\mathbf{A})h_{\mathrm{W}_N}(\mathbf{B}) \triangleq Y_N$
$h_{\mathrm{W}_N}(\mathbf{B})$
W_N
B
B
B
Q_1, C_1
Y_1
Q_2, C_2
Y_2
Y_N
Q_N, C_N
Master
A
$\mathbf{AB}_D$=?
Encode : $g_{\mathrm{W}_i}(\mathbf{A}) \triangleq C_i$
Query : Q_i to W_i
Decode : $d(Y_1, \cdots Y_N) = \mathbf{AB}_D$
Privacy : $I(D; Q_i, C_i, \mathbf{B}) = 0, \forall i \in [N]$
(workers do not collude each other)

Figure 3.2: The overall process of private coded computation.

$\mathbf{A}$ and sends encoded $\mathbf{A}$ to each worker with queries for the workers to encode the library $\mathbf{B}$. That is, $\mathbf{A}$ is encoded at the master for each worker whereas $\mathbf{B}$ is encoded at each worker. The workers encode $\mathbf{B}$ according to the queries and multiply the encoded $\mathbf{A}$ and $\mathbf{B}$. If several multiplications are assigned to each worker, the workers sequentially compute the multiplications and return each multiplication upon finishing it. When a sufficient number of multiplications are returned, the master obtains $\mathbf{AB}_D$ by decoding.

The encoding improves resiliency to stragglers. That is, by encoding, the whole multiplication $\mathbf{AB}_D$ is divided into smaller multiplications, and the master assigns one or more multiplications to each worker. I assume that the same number of multiplications is assigned to each worker and denote this number by L. I also denote the encoding function at the master for the worker W_i by g_{W_i}, where $g_{\mathrm{W}_i} : \mathbb{V}_1 \to \mathbb{U}_1^{\delta_1}$ for a vector space $\mathbb{U}_1$ over the same field $\mathbb{F}$. The queries that the master sends to the worker W_i are denoted by Q_i. Similar to g_{W_i}, let h_{W_i} denote the encoding function of the worker W_i, where $h_{\mathrm{W}_i} : \mathbb{V}_2^M \to \mathbb{U}_2^{\delta_2}$ for a vector space $\mathbb{U}_2$ over the same field $\mathbb{F}$. Note that $\delta_1\delta_2$ equals to L. Let me denote the multiplications returned from the worker W_i by Y_i, with which the master is able to recover the whole multiplication $\mathbf{AB}_D$. Accordingly, if I denote the decoding function by d, it should satisfy the following condition which is given by

$$d(Y_1, Y_2, \cdots, Y_N) = \mathbf{AB}_D.$$

The master's privacy is ensured when none of the workers identifies the index D

of the desired matrix $\mathbf{B}_D$ after the master recovers the whole multiplication. Since the privacy I consider is information-theoretic privacy, the privacy constraint for each worker W_i is given by

$$I(D; Q_i, C_i, \mathbf{B}) = 0, \tag{3.1}$$

where $C_i = g_{\mathrm{W}_i}(\mathbf{A})$.

I consider two performance metrics in private coded computation: computation time and communication load. I define these terms as follows.

Definition 1. In private coded computation, *communication load U* is defined by the amount of communication from the master to all of the workers.

That is, the communication load equals to the amount of encoded $\mathbf{A}$ transmitted to all of the workers.

For the computation time, I assume that the workers simultaneously start working.

Definition 2. In private coded computation, *computation time T* is defined by the time taken for the workers to return the sufficient number of multiplications to the master so that the master is able to recover the whole multiplication $\mathbf{AB}_D$.

In Definition 1, the communication from the workers to the master is excluded. This is because the time taken for the workers to return the multiplications to the master is included in computation time as defined in Definition 2.

3.3 Main results

In this section, I state main results, which are achievable scheme and privacy proof.

Theorem 1 (achievable scheme)**.** *In an achievable scheme for a private coded computation system with a library* $\mathbf{B}$ *of* M *matrices* $\{\mathbf{B}\}_{i=1}^{M}$ *and* N *workers* $\{W\}_{i=1}^{N}$*, where a master has its own matrix* $\mathbf{A}$*, the master's encoding of* $\mathbf{A}$ *for* W_i *is given by*

$$\left\{\tilde{\mathbf{A}}(x_p)\right\}_{p=1}^{L} = \left\{\sum_{l=0}^{m-1} \mathbf{A}_l x_p^l\right\}_{p=1}^{L},$$

the encoding of $\mathbf{B}$ *by the worker* W_i *in a group* G_t *is given by*

$$\tilde{\mathbf{B}}(y_t) = \tilde{\mathbf{B}}_D(y_t) + \sum_{k\in[M]\setminus D} \tilde{\mathbf{B}}_k(y_{j_k}) = \sum_{l=1}^{n-1} \mathbf{B}_{D,l} y_t^l + \sum_{k\in[M]\setminus D} \sum_{l=1}^{n-1} \mathbf{B}_{k,l} y_{j_k}^l,$$

and the computations returned from the worker W_i *in the group* G_t *are given by*

$$\left\{\tilde{\mathbf{A}}(x_p)\tilde{\mathbf{B}}(y_t)\right\}_{p=1}^{L},$$

where

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_0 \\ \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_{m-1} \end{bmatrix},$$

$$\mathbf{B}_k = \begin{bmatrix} \mathbf{B}_{k,1} & \cdots & \mathbf{B}_{k,n-1} \end{bmatrix}, k \in [M].$$

Theorem 2 (privacy proof)**.** *For the achievable scheme given in Theorem 1, the following privacy constraint for each worker* W_i *is satisfied.*

$$I(D; Q_i, C_i, \mathbf{B}) = 0, \forall i \in [N],$$

where D, Q_i, and C_i denote the desired index, the queries for W_i, and encoded $\mathbf{A}$ sent to W_i, respectively.

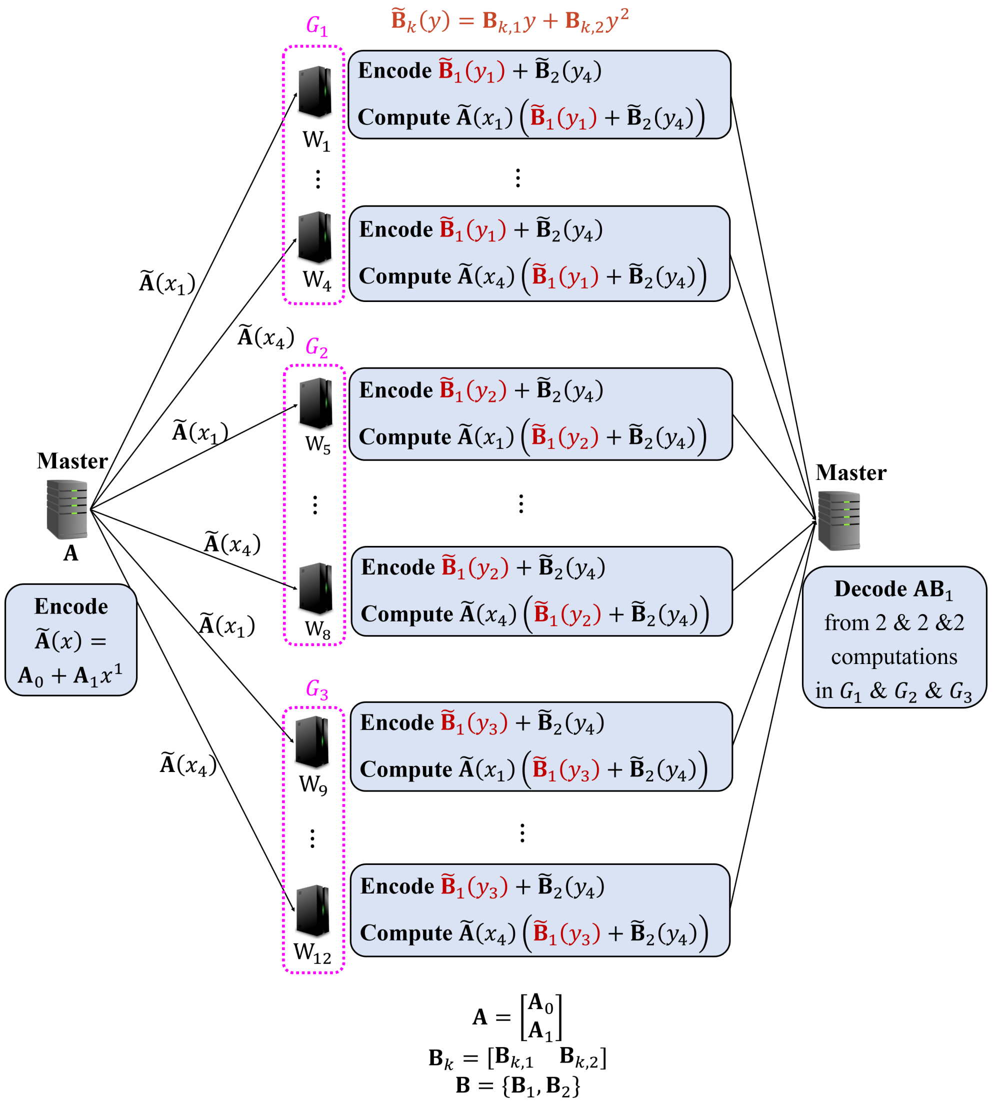
3.4 Private polynomial codes

In this section, I propose an achievable scheme for private coded computation, namely private polynomial codes. I provide two illustrative examples and describe the general method. I also show that the master's privacy is ensured. Next, I explain two special cases of my proposed scheme, which are referred to as *private one-shot polynomial codes* (POPC) and *private asynchronous polynomial codes* (PAPC). Note that the term 'private asynchronous polynomial codes' stems from the asynchronous coded computation which was explained in Section 3.1. Finally, I characterize the computation time and communication load of my proposed scheme.

3.4.1 First example

The master has a matrix $\mathbf{A} \in \mathbb{F}_q^{r\times s}$ for sufficiently large finite field $\mathbb{F}_q$. Specifically, the field size should be at least $\mathbb{F}_{2^3}$, which will be explained later. There are 12 non-colluding workers $\{\mathrm{W}_n\}_{n=1}^{12}$ who share a library $\mathbf{B}$ of two matrices $\mathbf{B}_1, \mathbf{B}_2 \in \mathbb{F}_q^{s\times t}$. The master wants to compute $\mathbf{A}\mathbf{B}_1$ using $\{\mathrm{W}_n\}_{n=1}^{12}$ while hiding the fact that the master desires $\mathbf{B}_1$ from the workers.

The overall process of the first example is depicted in Fig. 3.3. The matrix $\mathbf{A}$ is partitioned into two submatrices $\mathbf{A}_0, \mathbf{A}_1 \in \mathbb{F}_q^{r/2\times s}$ and each of $\mathbf{B}_1, \mathbf{B}_2$ are also



G_1
$\widetilde{\mathbf{B}}_k(y) = \mathbf{B}_{k,1}y + \mathbf{B}_{k,2}y^2$
Encode $\widetilde{\mathbf{B}}_1(y_1) + \widetilde{\mathbf{B}}_2(y_4)$
Compute $\widetilde{\mathbf{A}}(x_1)\left(\widetilde{\mathbf{B}}_1(y_1) + \widetilde{\mathbf{B}}_2(y_4)\right)$
W$_1$
Encode $\widetilde{\mathbf{B}}_1(y_1) + \widetilde{\mathbf{B}}_2(y_4)$
Compute $\widetilde{\mathbf{A}}(x_4)\left(\widetilde{\mathbf{B}}_1(y_1) + \widetilde{\mathbf{B}}_2(y_4)\right)$
W$_4$
$\widetilde{\mathbf{A}}(x_1)$
$\widetilde{\mathbf{A}}(x_4)$
G_2
Encode $\widetilde{\mathbf{B}}_1(y_2) + \widetilde{\mathbf{B}}_2(y_4)$
Compute $\widetilde{\mathbf{A}}(x_1)\left(\widetilde{\mathbf{B}}_1(y_2) + \widetilde{\mathbf{B}}_2(y_4)\right)$
W$_5$
$\widetilde{\mathbf{A}}(x_1)$
Master
A
Master
Encode $\widetilde{\mathbf{B}}_1(y_2) + \widetilde{\mathbf{B}}_2(y_4)$
Compute $\widetilde{\mathbf{A}}(x_4)\left(\widetilde{\mathbf{B}}_1(y_2) + \widetilde{\mathbf{B}}_2(y_4)\right)$
W$_8$
$\widetilde{\mathbf{A}}(x_4)$
$\widetilde{\mathbf{A}}(x_1)$
Encode $\widetilde{\mathbf{A}}(x) = \mathbf{A}_0 + \mathbf{A}_1x^1$
Decode $\mathbf{AB}_1$ from 2 & 2 &2 computations in G_1 & G_2 & G_3
G_3
Encode $\widetilde{\mathbf{B}}_1(y_3) + \widetilde{\mathbf{B}}_2(y_4)$
Compute $\widetilde{\mathbf{A}}(x_1)\left(\widetilde{\mathbf{B}}_1(y_3) + \widetilde{\mathbf{B}}_2(y_4)\right)$
W$_9$
$\widetilde{\mathbf{A}}(x_4)$
Encode $\widetilde{\mathbf{B}}_1(y_3) + \widetilde{\mathbf{B}}_2(y_4)$
Compute $\widetilde{\mathbf{A}}(x_4)\left(\widetilde{\mathbf{B}}_1(y_3) + \widetilde{\mathbf{B}}_2(y_4)\right)$
W$_{12}$
$\mathbf{A} = \begin{bmatrix}\mathbf{A}_0 \\ \mathbf{A}_1\end{bmatrix}$
$\mathbf{B}_k = [\mathbf{B}_{k,1} \quad \mathbf{B}_{k,2}]$
$\mathbf{B} = \{\mathbf{B}_1, \mathbf{B}_2\}$

Figure 3.3: The overall process of the first example.

partitioned into two submatrices $\mathbf{B}_{k,1}, \mathbf{B}_{k,2} \in \mathbb{F}_q^{s\times t/2}, k \in [2]$, so that

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_0 \\ \\ \mathbf{A}_1 \end{bmatrix},$$

$$\mathbf{B}_k = \begin{bmatrix} \mathbf{B}_{k,1} & \mathbf{B}_{k,2} \end{bmatrix},$$

$$\mathbf{A}\mathbf{B}_1 = \begin{bmatrix} \mathbf{A}_0\mathbf{B}_{1,1} & \mathbf{A}_0\mathbf{B}_{1,2} \\ \\ \mathbf{A}_1\mathbf{B}_{1,1} & \mathbf{A}_1\mathbf{B}_{1,2} \end{bmatrix}.$$

The encoding for each $\mathbf{A}$, $\mathbf{B}_1$ and $\mathbf{B}_2$ is as follows.

$$\tilde{\mathbf{A}}(x) = \mathbf{A}_0 + \mathbf{A}_1 x,$$

$$\tilde{\mathbf{B}}_k(y) = \mathbf{B}_{k,1} y + \mathbf{B}_{k,2} y^2,$$

where $k \in [2]$ and $x, y \in \mathbb{F}_q$ denote the variables of polynomials $\tilde{\mathbf{A}}(x)$ and $\tilde{\mathbf{B}}_k(y)$, respectively. Note that the master's own matrix $\mathbf{A}$ and the library $\mathbf{B}$ are encoded separately.

The master randomly chooses 4 distinct points $\{x_p\}_{p=1}^4$ and another 4 distinct points $\{y_p\}_{p=1}^4$. Since the master should choose 8 distinct points altogether, the field size should be at least $\mathbb{F}_{2^3}$. After that, the master divides 12 workers into three equal-sized groups, which are denoted by G_1, G_2, and G_3. Without loss of generality, I assume that the workers $\{\mathrm{W}_i\}_{i=1}^4$, $\{\mathrm{W}_i\}_{i=5}^8$, and $\{\mathrm{W}_i\}_{i=9}^{12}$ are grouped into G_1, G_2, and G_3, respectively.

After that, the master computes 4 evaluations $\{\tilde{\mathbf{A}}(x_p)\}_{p=1}^4$ and sends $\tilde{\mathbf{A}}(x_i)$ to 3 workers W_i, W_{i+4}, and W_{i+8}. Note that these workers belong to distinct groups.

The master also sends queries for the worker in the group G_t to compute the encoded library which is given by

$$\tilde{\mathbf{B}}_1(y_t) + \tilde{\mathbf{B}}_2(y_4)$$

$$= \mathbf{B}_{1,1}y_t + \mathbf{B}_{1,2}y_t^2 + \mathbf{B}_{2,1}y_4 + \mathbf{B}_{2,2}y_4^2.$$

That is, each group G_t evaluates $\tilde{\mathbf{B}}_1(y)$ at distinct point y_t. Consider a polynomial $\tilde{\mathbf{B}}_1(y) + \tilde{\mathbf{B}}_2(y_4)$ in y, whose degree is 2. Since the point y_4 is identical across the groups, the term $\mathbf{B}_{2,1}y_4 + \mathbf{B}_{2,2}y_4^2$ is a constant term and there are 3 evaluations of this polynomial across the groups.

Nevertheless, the workers cannot notice this asymmetry that $\tilde{\mathbf{B}}_1(y)$ is evaluated at three points whereas $\tilde{\mathbf{B}}_2(y)$ is evaluated at only one point. That is, each worker in the group G_t sees the points y_t and y_4 as two identical random points queried by the master. This is because I assumed that the workers do not communicate with each other. Therefore, the workers cannot know that $\mathbf{B}_1$ is the desired matrix and the master's privacy on $\mathbf{B}_1$ is ensured.

After encoding the library, the worker W_i, W_{i+4} or W_{i+8} in G_t computes a multiplication

$$\tilde{\mathbf{A}}(x_i)(\tilde{\mathbf{B}}_1(y_t) + \tilde{\mathbf{B}}_2(y_4)).$$

Since there are 4 workers in each G_t, up to 4 multiplications are returned from G_t to the master.

The decoding is two-fold : 3 interpolations of polynomials in x which are followed by two interpolations of polynomials in y. The first 3 interpolations in x are done group-wise. That is, when the master receives two multiplications from the fastest two workers in G_t, the master interpolates the polynomial in x which is given by

$$\tilde{\mathbf{A}}(x)(\tilde{\mathbf{B}}_1(y_t) + \tilde{\mathbf{B}}_2(y_4)),$$

where the degree is 1 and the term $\tilde{\mathbf{B}}_1(y_t) + \tilde{\mathbf{B}}_2(y_4)$ is constant within the group G_t.

As a result, from each group G_t, the master obtains two coefficients

$$\begin{gathered}\mathbf{A}_0(\tilde{\mathbf{B}}_1(y_t) + \tilde{\mathbf{B}}_2(y_4)) \\ = \mathbf{A}_0(\mathbf{B}_{1,1}y_t + \mathbf{B}_{1,2}y_t^2 + \mathbf{B}_{2,1}y_4 + \mathbf{B}_{2,2}y_4^2), \\ \mathbf{A}_1(\tilde{\mathbf{B}}_1(y_t) + \tilde{\mathbf{B}}_2(y_4)) \\ = \mathbf{A}_1(\mathbf{B}_{1,1}y_t + \mathbf{B}_{1,2}y_t^2 + \mathbf{B}_{2,1}y_4 + \mathbf{B}_{2,2}y_4^2).\end{gathered}$$

Consider two polynomials in y which are given by

$$\begin{gathered}\mathbf{A}_0\mathbf{B}_{1,1}y + \mathbf{A}_0\mathbf{B}_{1,2}y^2 + \mathbf{A}_0(\mathbf{B}_{2,1}y_4 + \mathbf{B}_{2,2}y_4^2), \\ \mathbf{A}_1\mathbf{B}_{1,1}y + \mathbf{A}_1\mathbf{B}_{1,2}y^2 + \mathbf{A}_1(\mathbf{B}_{2,1}y_4 + \mathbf{B}_{2,2}y_4^2),\end{gathered}$$

where the degree of each polynomial is 2.

Since the 6 coefficients obtained from G_1, G_2 and G_3 are 3 evaluations of each of two polynomials in y, the master is able to interpolate these two polynomials in y. As a result, the master obtains $\mathbf{AB}_1$ from the coefficients $\mathbf{A}_0\mathbf{B}_{1,1}$, $\mathbf{A}_0\mathbf{B}_{1,2}$, $\mathbf{A}_1\mathbf{B}_{1,1}$, and $\mathbf{A}_1\mathbf{B}_{1,2}$.

Remark 1. *In the first example, the stragglers are mitigated within the group, not across the groups. That is, in each group where there are 4 workers, the computation of the two slowest workers are ignored.*

Remark 2. *As explained, the interpolations of polynomials in x are done group-wise, and the workers are assumed not to communicate with each other. Therefore, the identical set of evaluations $\{\tilde{\mathbf{A}}(x_p)\}_{p=1}^{4}$ is allowed to be transmitted to every group.*

Remark 3. *Let me assume that the master desires $\mathbf{B}_2$ instead of $\mathbf{B}_1$. Still, the master randomly chooses four distinct points $\{y_p\}_{p=1}^{4}$ and divides the workers into three equal-sized groups. However, at this time, the master sends queries to group G_t for the workers in G_t to compute*

$$\tilde{\mathbf{B}}_2(y_t) + \tilde{\mathbf{B}}_1(y_4)$$
$$= \mathbf{B}_{2,1}y_t + \mathbf{B}_{2,2}y_t^2 + \mathbf{B}_{1,1}y_4 + \mathbf{B}_{1,2}y_4^2.$$

That is, $\tilde{\mathbf{B}}_2(y)$ is evaluated at three points y_1, y_2, and y_3, whereas $\tilde{\mathbf{B}}_1(y)$ is evaluated at only one point y_4. Since the four distinct points are randomly chosen and the workers do not communicate with each other, they still see y_t and y_4 as identical random points, which indicates that the master's privacy on $\mathbf{B}_2$ in ensured.

Remark 4. *In PPC, the workers are always grouped according to the point with which they evaluate the desired matrix. That is, for a given set of evaluating points for undesired matrices, each worker evaluates the desired matrix with only one point. If not, the master's privacy on the desired matrix cannot be ensured.*

For example, in the first example, let me assume that there is a worker who computes two evaluations

$$\tilde{\mathbf{B}}_1(y_a) + \tilde{\mathbf{B}}_2(y_4),$$

$$\tilde{\mathbf{B}}_1(y_b) + \tilde{\mathbf{B}}_2(y_4).$$

As a result, this worker immediately recognizes that $\mathbf{B}_1$ *is more evaluated than* $\mathbf{B}_2$*, thus implying that the master's privacy on* $\mathbf{B}_1$ *is infringed.*

Of course, for privacy, the master may request this worker to compute two more evaluations

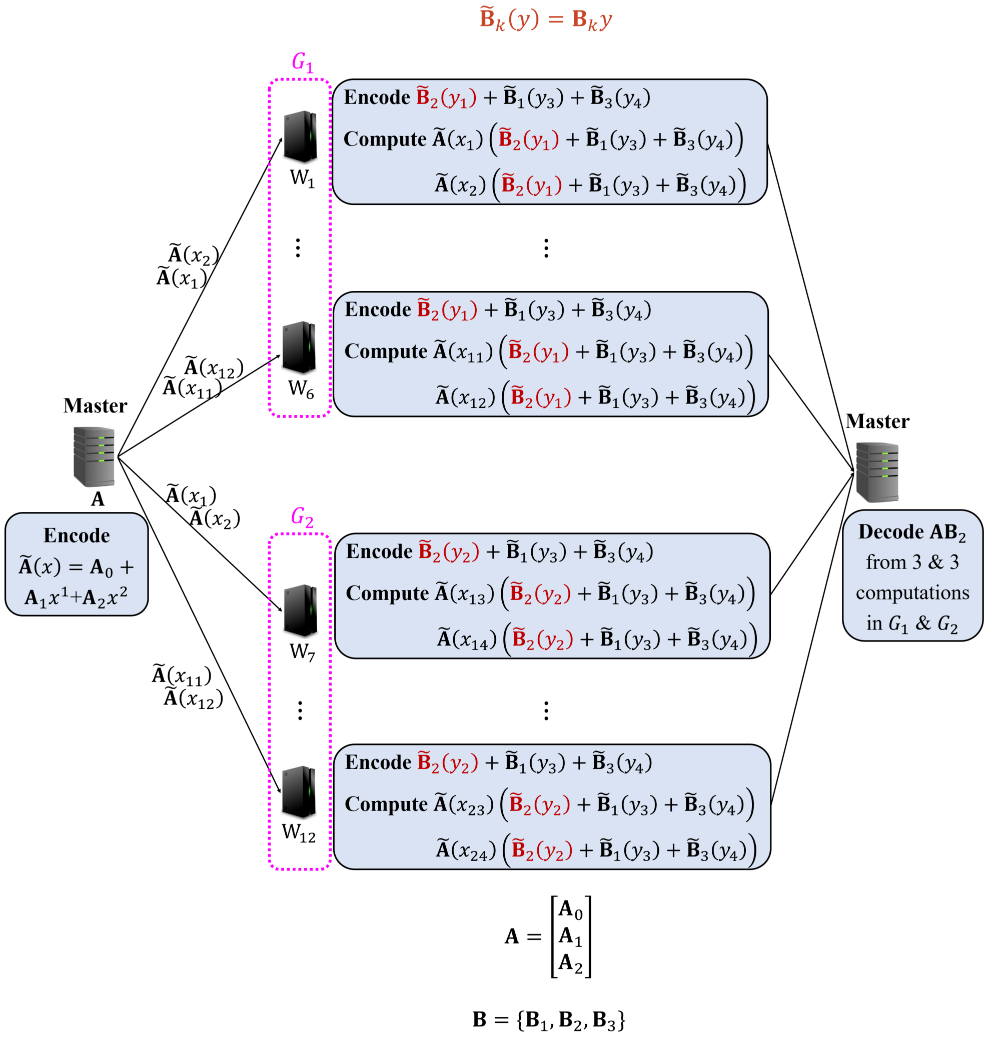
$$\tilde{\mathbf{B}}_1(y_a) + \tilde{\mathbf{B}}_2(y_5),$$

$$\tilde{\mathbf{B}}_1(y_b) + \tilde{\mathbf{B}}_2(y_5).$$

However, the polynomials $\tilde{\mathbf{B}}_1(y) + \tilde{\mathbf{B}}_2(y_4)$ *and* $\tilde{\mathbf{B}}_1(y) + \tilde{\mathbf{B}}_2(y_5)$ *are distinct since* y_4 *and* y_5 *are distinct. Therefore, this worker evaluated* $\tilde{\mathbf{B}}_1(y)$ *at only one point for each polynomial.*

3.4.2 Second example

I now explain the second example where there are 12 workers. The overall process is depicted in Fig. 3.4. This time, there are three matrices $\mathbf{B}_1$, $\mathbf{B}_2$ and $\mathbf{B}_3$ in the library $\mathbf{B}$. The matrix $\mathbf{A}$ has three submatrices so that $\mathbf{A} = \begin{bmatrix} \mathbf{A}_0{}^T & \mathbf{A}_1{}^T & \mathbf{A}_2^T \end{bmatrix}^T$, whereas each $\mathbf{B}_k$ is not partitioned into submatrices. I assume the master wants $\mathbf{B}_2$. The encoded



$\widetilde{\mathbf{B}}_k(y) = \mathbf{B}_k y$
G_1
G_2
Master
A
Encode $\widetilde{\mathbf{A}}(x) = \mathbf{A}_0 + \mathbf{A}_1 x^1 + \mathbf{A}_2 x^2$
$\widetilde{\mathbf{A}}(x_2)$
$\widetilde{\mathbf{A}}(x_1)$
$\widetilde{\mathbf{A}}(x_{12})$
$\widetilde{\mathbf{A}}(x_{11})$
$\widetilde{\mathbf{A}}(x_1)$
$\widetilde{\mathbf{A}}(x_2)$
$\widetilde{\mathbf{A}}(x_{11})$
$\widetilde{\mathbf{A}}(x_{12})$
W$_1$
W$_6$
W$_7$
W$_{12}$
Encode $\widetilde{\mathbf{B}}_2(y_1) + \widetilde{\mathbf{B}}_1(y_3) + \widetilde{\mathbf{B}}_3(y_4)$
Compute $\widetilde{\mathbf{A}}(x_1)\left(\widetilde{\mathbf{B}}_2(y_1) + \widetilde{\mathbf{B}}_1(y_3) + \widetilde{\mathbf{B}}_3(y_4)\right)$
$\widetilde{\mathbf{A}}(x_2)\left(\widetilde{\mathbf{B}}_2(y_1) + \widetilde{\mathbf{B}}_1(y_3) + \widetilde{\mathbf{B}}_3(y_4)\right)$
Encode $\widetilde{\mathbf{B}}_2(y_1) + \widetilde{\mathbf{B}}_1(y_3) + \widetilde{\mathbf{B}}_3(y_4)$
Compute $\widetilde{\mathbf{A}}(x_{11})\left(\widetilde{\mathbf{B}}_2(y_1) + \widetilde{\mathbf{B}}_1(y_3) + \widetilde{\mathbf{B}}_3(y_4)\right)$
$\widetilde{\mathbf{A}}(x_{12})\left(\widetilde{\mathbf{B}}_2(y_1) + \widetilde{\mathbf{B}}_1(y_3) + \widetilde{\mathbf{B}}_3(y_4)\right)$
Encode $\widetilde{\mathbf{B}}_2(y_2) + \widetilde{\mathbf{B}}_1(y_3) + \widetilde{\mathbf{B}}_3(y_4)$
Compute $\widetilde{\mathbf{A}}(x_{13})\left(\widetilde{\mathbf{B}}_2(y_2) + \widetilde{\mathbf{B}}_1(y_3) + \widetilde{\mathbf{B}}_3(y_4)\right)$
$\widetilde{\mathbf{A}}(x_{14})\left(\widetilde{\mathbf{B}}_2(y_2) + \widetilde{\mathbf{B}}_1(y_3) + \widetilde{\mathbf{B}}_3(y_4)\right)$
Encode $\widetilde{\mathbf{B}}_2(y_2) + \widetilde{\mathbf{B}}_1(y_3) + \widetilde{\mathbf{B}}_3(y_4)$
Compute $\widetilde{\mathbf{A}}(x_{23})\left(\widetilde{\mathbf{B}}_2(y_2) + \widetilde{\mathbf{B}}_1(y_3) + \widetilde{\mathbf{B}}_3(y_4)\right)$
$\widetilde{\mathbf{A}}(x_{24})\left(\widetilde{\mathbf{B}}_2(y_2) + \widetilde{\mathbf{B}}_1(y_3) + \widetilde{\mathbf{B}}_3(y_4)\right)$
Master
Decode $\mathbf{AB}_2$ from 3 & 3 computations in G_1 & G_2
$\mathbf{A} = \begin{bmatrix}\mathbf{A}_0\\ \mathbf{A}_1\\ \mathbf{A}_2\end{bmatrix}$
$\mathbf{B} = \{\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3\}$

Figure 3.4: The overall process of the second example.

matrices $\tilde{\mathbf{A}}(x)$ and $\tilde{\mathbf{B}}_k(y)$ are given as follows.

$$\tilde{\mathbf{A}}(x) = \mathbf{A}_0 + \mathbf{A}_1 x + \mathbf{A}_1 x^2,$$

$$\tilde{\mathbf{B}}_k(y) = \mathbf{B}_k y.$$

The master randomly chooses 12 distinct points $\{x_l\}_{l=1}^{12}$ and 4 distinct points $\{y_p\}_{p=1}^{4}$. Since total 16 points are necessary for encoding, the field size should be at least $\mathbb{F}_{2^4}$. The master divides the workers into 2 groups and without loss of generality, I assume that the workers $\{\mathrm{W}_1, \mathrm{W}_2, \cdots, \mathrm{W}_6\}$ are G_1 and the others are G_2. After that, the master computes 12 evaluations $\{\tilde{\mathbf{A}}(x_l)\}_{l=1}^{12}$ and sends two evaluations $\tilde{\mathbf{A}}(x_{2i-1})$ and $\tilde{\mathbf{A}}(x_{2i})$ to each of two workers W_i and W_{i+6}. Note that the workers W_i and W_{i+6} belong to different groups. The master also sends queries for the workers in G_t to compute

$$\tilde{\mathbf{B}}_2(y_t) + \tilde{\mathbf{B}}_1(y_3) + \tilde{\mathbf{B}}_3(y_4)$$

$$= \mathbf{B}_2 y_t + \mathbf{B}_1 y_3 + \mathbf{B}_3 y_4.$$

Consider a polynomial $\tilde{\mathbf{B}}_2(y) + \tilde{\mathbf{B}}_1(y_3) + \tilde{\mathbf{B}}_3(y_4)$ in y, whose degree is 1. Since $\mathbf{B}_1$ and $\mathbf{B}_3$ are undesired matrices, the points y_3 and y_4 are identical across the groups, which makes the term $\mathbf{B}_1 y_3 + \mathbf{B}_3 y_4$ as constant term in the polynomial. Still, the workers see the points y_t, y_3, and y_4 as identically random points because they do not communicate with each other. Therefore, the master's privacy on $\mathbf{B}_2$ is ensured.

After encoding the library, the worker W_i or W_{i+6} in G_t sequentially computes two multiplications

$$\tilde{\mathbf{A}}(x_{2i-1})(\tilde{\mathbf{B}}_2(y_t) + \tilde{\mathbf{B}}_1(y_3) + \tilde{\mathbf{B}}_3(y_4)),$$

$$\tilde{\mathbf{A}}(x_{2i})(\tilde{\mathbf{B}}_2(y_t) + \tilde{\mathbf{B}}_1(y_3) + \tilde{\mathbf{B}}_3(y_4)).$$

That is, the workers return each multiplication to the master upon finishing it. Since there are 6 workers in G_t, up to 12 multiplications are sequentially returned from G_t.

The decoding is two-fold : two interpolations of polynomials in x which are followed by 3 interpolations of polynomials in y. The first two interpolations in x are done group-wise. That is, when the master receives 3 multiplications from G_t, the master interpolates the polynomial in x which is given by

$$\tilde{\mathbf{A}}(x)(\tilde{\mathbf{B}}_2(y_t) + \tilde{\mathbf{B}}_1(y_3) + \tilde{\mathbf{B}}_3(y_4)),$$

whose degree is 2 and the encoded library is identical within the group G_t.

After each interpolation of polynomial in x, the master obtains 3 coefficients

$$\mathbf{A}_0(\tilde{\mathbf{B}}_2(y_t) + \tilde{\mathbf{B}}_1(y_3) + \tilde{\mathbf{B}}_3(y_4))$$

$$= \mathbf{A}_0(\mathbf{B}_2 y_t + \mathbf{B}_1 y_3 + \mathbf{B}_3 y_4),$$

$$\mathbf{A}_1(\tilde{\mathbf{B}}_2(y_t) + \tilde{\mathbf{B}}_1(y_3) + \tilde{\mathbf{B}}_3(y_4))$$

$$= \mathbf{A}_1(\mathbf{B}_2 y_t + \mathbf{B}_1 y_3 + \mathbf{B}_3 y_4),$$

$$\mathbf{A}_2(\tilde{\mathbf{B}}_2(y_t) + \tilde{\mathbf{B}}_1(y_3) + \tilde{\mathbf{B}}_3(y_4))$$

$$= \mathbf{A}_2(\mathbf{B}_2 y_t + \mathbf{B}_1 y_3 + \mathbf{B}_3 y_4).$$

Consider 3 polynomials in y which are given by

$$\mathbf{A}_0\mathbf{B}_2y + (\mathbf{A}_0\mathbf{B}_1y_3 + \mathbf{A}_0\mathbf{B}_3y_4),$$

$$\mathbf{A}_1\mathbf{B}_2y + (\mathbf{A}_1\mathbf{B}_1y_3 + \mathbf{A}_1\mathbf{B}_3y_4),$$

$$\mathbf{A}_2\mathbf{B}_2y + (\mathbf{A}_2\mathbf{B}_1y_3 + \mathbf{A}_2\mathbf{B}_3y_4),$$

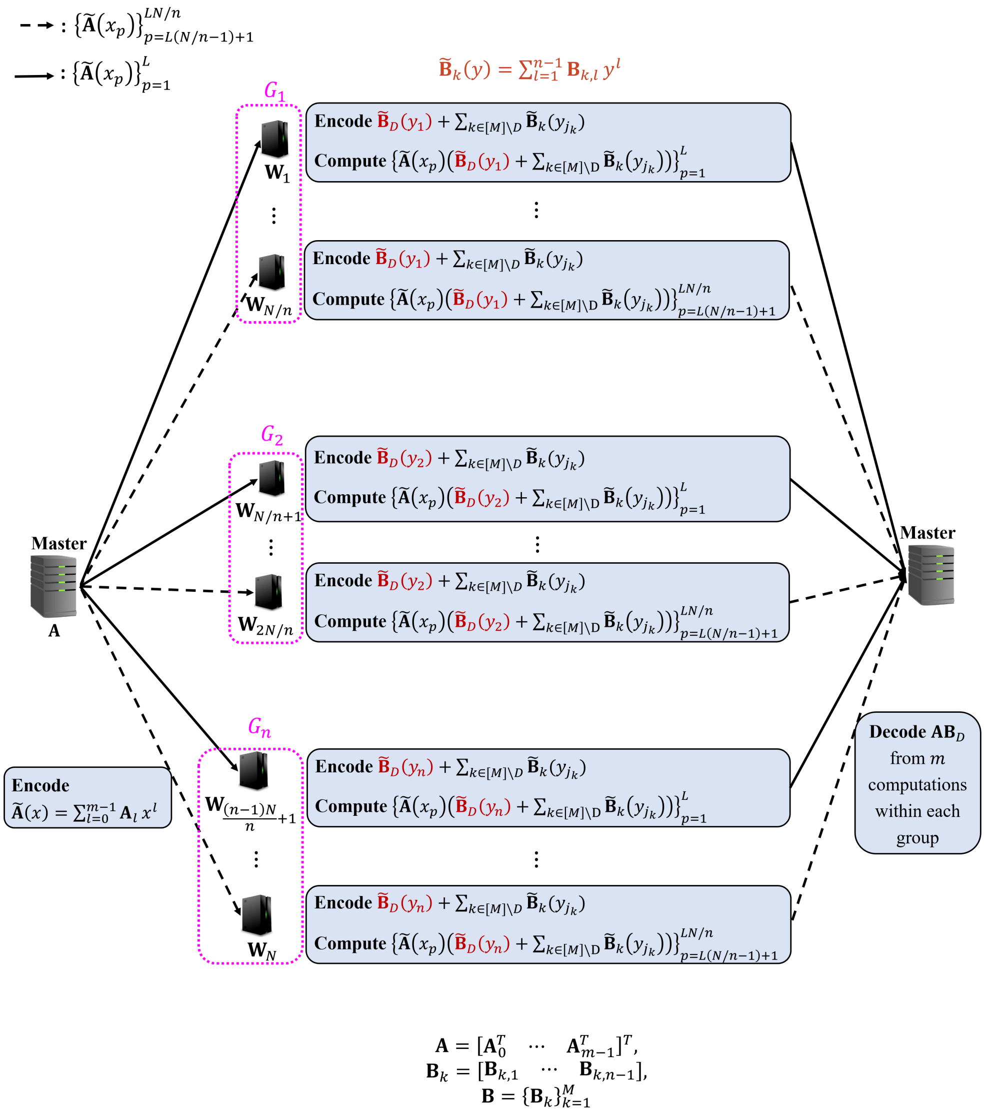
each of whose degree is 1.

Since the 6 coefficients obtained from G_1 and G_2 are two evaluations of each of 3 polynomials in y, the master is able to interpolate these 3 polynomials in y. As a result, the master obtains $\mathbf{AB}_2$ from the coefficients $\mathbf{A}_0\mathbf{B}_2$, $\mathbf{A}_1\mathbf{B}_2$, and $\mathbf{A}_2\mathbf{B}_2$.

Remark 5. *In the first example, each worker computes only one multiplications, which implies that $L = 1$. On the other hand, L equals to 2 in the second example. If I set $L = 1$ in the second example, the results of the slowest 3 workers among 6 workers within a group would always be ignored. However, since $L = 2$, there is a chance that the fastest worker returns two multiplications while the second fastest worker returns a multiplication. In this case, the effects of the slowest 4 workers are mitigated within a group.*

3.4.3 General description

In this section, I describe the general process of PPC. The overall process is depicted in Fig. 4.3.2. There are N non-colluding workers $\{\mathrm{W}_n\}_{n=1}^N$ and each worker has a library $\mathbf{B}$ of M matrices $\{\mathbf{B}_k\}_{k=1}^M$ where each $\mathbf{B}_k \in \mathbb{F}_q^{s\times t}$ for sufficiently large finite



$- - \rightarrow : \{\tilde{\mathbf{A}}(x_p)\}_{p=L(N/n-1)+1}^{LN/n}$
$\longrightarrow : \{\tilde{\mathbf{A}}(x_p)\}_{p=1}^{L}$
$\tilde{\mathbf{B}}_k(y) = \sum_{l=1}^{n-1} \mathbf{B}_{k,l}\, y^l$
G_1
$\mathbf{W}_1$
$\mathbf{W}_{N/n}$
Encode $\tilde{\mathbf{B}}_D(y_1) + \sum_{k\in[M]\setminus D} \tilde{\mathbf{B}}_k(y_{j_k})$
Compute $\{\tilde{\mathbf{A}}(x_p)(\tilde{\mathbf{B}}_D(y_1) + \sum_{k\in[M]\setminus D} \tilde{\mathbf{B}}_k(y_{j_k}))\}_{p=1}^{L}$
Encode $\tilde{\mathbf{B}}_D(y_1) + \sum_{k\in[M]\setminus D} \tilde{\mathbf{B}}_k(y_{j_k})$
Compute $\{\tilde{\mathbf{A}}(x_p)(\tilde{\mathbf{B}}_D(y_1) + \sum_{k\in[M]\setminus D} \tilde{\mathbf{B}}_k(y_{j_k}))\}_{p=L(N/n-1)+1}^{LN/n}$
G_2
$\mathbf{W}_{N/n+1}$
$\mathbf{W}_{2N/n}$
Encode $\tilde{\mathbf{B}}_D(y_2) + \sum_{k\in[M]\setminus D} \tilde{\mathbf{B}}_k(y_{j_k})$
Compute $\{\tilde{\mathbf{A}}(x_p)(\tilde{\mathbf{B}}_D(y_2) + \sum_{k\in[M]\setminus D} \tilde{\mathbf{B}}_k(y_{j_k}))\}_{p=1}^{L}$
Encode $\tilde{\mathbf{B}}_D(y_2) + \sum_{k\in[M]\setminus D} \tilde{\mathbf{B}}_k(y_{j_k})$
Compute $\{\tilde{\mathbf{A}}(x_p)(\tilde{\mathbf{B}}_D(y_2) + \sum_{k\in[M]\setminus D} \tilde{\mathbf{B}}_k(y_{j_k}))\}_{p=L(N/n-1)+1}^{LN/n}$
Master
A
Master
G_n
$\mathbf{W}_{\frac{(n-1)N}{n}+1}$
$\mathbf{W}_N$
Encode $\tilde{\mathbf{B}}_D(y_n) + \sum_{k\in[M]\setminus D} \tilde{\mathbf{B}}_k(y_{j_k})$
Compute $\{\tilde{\mathbf{A}}(x_p)(\tilde{\mathbf{B}}_D(y_n) + \sum_{k\in[M]\setminus D} \tilde{\mathbf{B}}_k(y_{j_k}))\}_{p=1}^{L}$
Encode $\tilde{\mathbf{B}}_D(y_n) + \sum_{k\in[M]\setminus D} \tilde{\mathbf{B}}_k(y_{j_k})$
Compute $\{\tilde{\mathbf{A}}(x_p)(\tilde{\mathbf{B}}_D(y_n) + \sum_{k\in[M]\setminus D} \tilde{\mathbf{B}}_k(y_{j_k}))\}_{p=L(N/n-1)+1}^{LN/n}$
Encode $\tilde{\mathbf{A}}(x) = \sum_{l=0}^{m-1} \mathbf{A}_l\, x^l$
Decode $\mathbf{AB}_D$ from m computations within each group
$\mathbf{A} = [\mathbf{A}_0^T \ \cdots \ \mathbf{A}_{m-1}^T]^T,$
$\mathbf{B}_k = [\mathbf{B}_{k,1} \ \cdots \ \mathbf{B}_{k,n-1}],$
$\mathbf{B} = \{\mathbf{B}_k\}_{k=1}^{M}$

Figure 3.5: The overall process of proposed scheme

field $\mathbb{F}_q$. The master has a matrix $\mathbf{A} \in \mathbb{F}_q^{r \times s}$ and desires to multiply $\mathbf{A}$ by a matrix $\mathbf{B}_D$ in the library $\mathbf{B}$ while concealing the index D from all of the workers. Matrices $\mathbf{A}$ and each $\mathbf{B}_k$ are partitioned into m submatrices $\{\mathbf{A}_k\}_{k=0}^{m-1} \in \mathbb{F}_q^{r/m \times s}$ and $n-1$ submatrices $\{\mathbf{B}_{k,l}\}_{l=1}^{n-1} \in \mathbb{F}_q^{s \times t/(n-1)}$, respectively, so that

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_0 \\ \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_{m-1} \end{bmatrix},$$

$$\mathbf{B}_k = \begin{bmatrix} \mathbf{B}_{k,1} & \cdots & \mathbf{B}_{k,n-1} \end{bmatrix},$$

$$\mathbf{A}\mathbf{B}_D = \begin{bmatrix} \mathbf{A}_0\mathbf{B}_{D,1} & \mathbf{A}_0\mathbf{B}_{D,2} & \cdots & \mathbf{A}_0\mathbf{B}_{D,n-1} \\ \mathbf{A}_1\mathbf{B}_{D,1} & \mathbf{A}_1\mathbf{B}_{D,2} & \cdots & \mathbf{A}_1\mathbf{B}_{D,n-1} \\ \vdots & \vdots & \cdots & \vdots \\ \mathbf{A}_{m-1}\mathbf{B}_{D,1} & \mathbf{A}_{m-1}\mathbf{B}_{D,2} & \cdots & \mathbf{A}_{m-1}\mathbf{B}_{D,n-1} \end{bmatrix}.$$

The encoding for each $\mathbf{A}$ and $\{\mathbf{B}_k\}_{k=1}^{M}$ is given as follows.

$$\tilde{\mathbf{A}}(x) = \sum_{l=0}^{m-1} \mathbf{A}_l x^l, \tag{3.2}$$

$$\tilde{\mathbf{B}}_k(y) = \sum_{l=1}^{n-1} \mathbf{B}_{k,l} y^l, \tag{3.3}$$

where $k \in [M]$ and $x, y \in \mathbb{F}_q$ for sufficiently large finite field $\mathbb{F}_q$.

As explained in Section 3.2, the parameter L denotes the number of evaluations of $\tilde{\mathbf{A}}(x)$ that the master sends to each worker. Accordingly, the master randomly chooses LN/n distinct points $\{x_p\}_{p=1}^{LN/n}$, n distinct points $\{y_p\}_{p=1}^{n}$, and $M-1$ distinct points

$\{y_{j_k} | k \in [M] \backslash D\}$. After that, the master divides the workers into n equal-sized groups $\{G_t\}_{t=1}^{n}$. Without loss of generality, I assume that the group G_t has N/n workers $\{\mathrm{W}_i\}_{i=(t-1)N/n+1}^{tN/n}$.

The master computes LN/n evaluations $\{\tilde{\mathbf{A}}(x_p)\}_{p=1}^{LN/n}$ and sends L evaluations $\{\tilde{\mathbf{A}}(x_p)\}_{p=L(i-1)+1}^{Li}$ to each of n workers $\{\mathrm{W}_i, \mathrm{W}_{i+N/n}, \mathrm{W}_{i+2N/n}, \cdots, \mathrm{W}_{i+(n-1)N/n}\}$ where $i \in [N/n]$. Note that the worker $\mathrm{W}_{i+(t-1)N/n}$ belongs to G_t. The master also sends queries to the workers in G_t so that they compute encoded library as follows.

$$\tilde{\mathbf{B}}_D(y_t) + \sum_{k \in [M] \backslash D} \tilde{\mathbf{B}}_k(y_{j_k}).$$

Note that the term $\sum_{k \in [M] \backslash D} \tilde{\mathbf{B}}_k(y_{j_k})$ is constant across the groups.

After computing the encoded library, each worker $\mathrm{W}_{i+(t-1)N/n}$ sequentially computes L multiplications

$$\left\{ \tilde{\mathbf{A}}(x_p)(\tilde{\mathbf{B}}_D(y_t) + \sum_{k \in [M] \backslash D} \tilde{\mathbf{B}}_k(y_{j_k})) \right\}_{p=L(i-1)+1}^{Li}.$$

The worker returns each multiplication to the master upon finishing it.

The decoding is two-fold : n interpolations of polynomials in x which are followed by m interpolations of polynomials in y. The first n interpolations in x are done group-wise. That is, when the master receives m multiplications from G_t, the master interpolates the polynomial in x which is given by

$$\tilde{\mathbf{A}}(x)(\tilde{\mathbf{B}}_D(y_t) + \sum_{k \in [M] \backslash D} \tilde{\mathbf{B}}_k(y_{j_k})),$$

whose degree is $m - 1$.

Note that the encoded library is identical within the group G_t. Recalling the polynomial $\tilde{\mathbf{A}}(x)$ which was given in (3.3), the master obtains m coefficients

$$\left\{\mathbf{A}_l(\tilde{\mathbf{B}}_D(y_t) + \sum_{k\in[M]\setminus D} \tilde{\mathbf{B}}_k(y_{j_k}))\right\}_{l=0}^{m-1}.$$

Consider m polynomials in y which are given by

$$\left\{\mathbf{A}_l\tilde{\mathbf{B}}_D(y) + (\sum_{k\in[M]\setminus D} \mathbf{A}_l\tilde{\mathbf{B}}_k(y_{j_k}))\right\}_{l=0}^{m-1},$$

each of whose degree is $n-1$.

Since the mn coefficients obtained from $\{G_t\}_{t=1}^{n}$ are n evaluations of each of m polynomials in y, the master is able to interpolate these m polynomials in y. As a result, recalling the polynomial $\tilde{\mathbf{B}}_D(y)$ which was given in (3.3), the master obtains $\mathbf{A}\mathbf{B}_D$ from the coefficients

$$\{\mathbf{A}_l\mathbf{B}_{D,r}\}_{(l,r)=(0,1)}^{(m,n-1)}.$$

Remark 6. *Recalling that each group G_t receives LN/n evaluations from the master, LN/n should be larger than m so that the master interpolates the polynomial in x from m multiplications returned from G_t. That is,*

$$L\frac{N}{n} \geq m. \tag{3.4}$$

3.4.4 Privacy proof

In this section, I show that the master's privacy is ensured in PPC. In particular, I show that the privacy constraint for each worker W_i in group G_t is satisfied, which was given by (3.1). By a chain rule, I write the privacy constraint as follows.

$$I(D; Q_i, C_i, \mathbf{B}) = I(D; Q_i) + I(D; \mathbf{B}|Q_i) + I(D; C_i|Q_i, \mathbf{B})$$

Recall that Q_i and C_i denote the queries for worker W_i and the evaluations of $\tilde{\mathbf{A}}(x)$ sent to W_i, respectively. Therefore, it is sufficient to show that the index D is independent of queries, library, and evaluations of $\tilde{\mathbf{A}}(x)$.

First, I show that the evaluations of $\tilde{\mathbf{A}}(x)$ are independent of D. The matrices $\mathbf{A}$ and those in the library $\mathbf{B}$ are independent. Moreover, they are encoded separately as polynomials in x and y, respectively. Therefore, the evaluations of $\tilde{\mathbf{A}}(x)$ are independent of D, thus implying that

$$I(D; C_i|Q_i, \mathbf{B}) = 0.$$

Next, I show that the library $\mathbf{B}$ is independent of D. The library $\mathbf{B}$ is exclusively shared by the workers and the master does not have it. That is, the master determines D without knowing any information about $\mathbf{B}$. Therefore, the library $\mathbf{B}$ is independent of D, thus implying that

$$I(D; \mathbf{B}|Q_i) = 0.$$

Finally, I show that the queries are independent of D. The queries Q_i are fourfold:

1. $Q_{i,p}$: for partitioning each matrix in the library into submatrices

2. $Q_{i,e}$: for evaluating each $\tilde{\mathbf{B}}_k(y)$

3. $Q_{i,s}$: for summing the evaluations

4. $Q_{i,c}$: for multiplying the encoded library and C_i

All of submatrices $\{\mathbf{B}_{k,l}\}_{(k,l)=(1,1)}^{(M,n-1)}$ are elements in $\mathbb{F}_q^{s\times t/(n-1)}$. That is, they are of equal size. Therefore, $Q_{i,p}$ are independent of D, thus implying that

$$I(D;Q_{i,p})=0.$$

Recall that the points y_t and $\{y_{j_k}|k\in[M]\setminus D\}$ are distinct and randomly chosen. Therefore, $Q_{i,e}$ are independent of D, thus implying that

$$I(D;Q_{i,e})=0.$$

All of the evaluations of $\{\tilde{\mathbf{B}}_k(y)\}_{k=1}^{M}$ are symmetrically summed into one equation

$$\tilde{\mathbf{B}}_D(y_t)+\sum_{k\in[M]\setminus D}\tilde{\mathbf{B}}_k(y_{j_k}).$$

Therefore, $Q_{i,s}$ are independent of D, thus implying that

$$I(D;Q_{i,s})=0.$$

As explained, C_i and the term

$$\tilde{\mathbf{B}}_D(y_t)+\sum_{k\in[M]\setminus D}\tilde{\mathbf{B}}_k(y_{j_k})$$

are independent of D. Therefore, $Q_{i,c}$ are also independent of D, thus implying that

$$I(D;Q_{i,c})=0.$$

That is, the queries are also independent of D, thus implying that

$$I(D;Q_i)=0.$$

As a result, the privacy constraint is satisfied for the worker W_i. Since the privacy constraint is satisfied for every worker, the master's privacy is ensured in PPC. $\square$

3.4.5 Performance analysis

In this section, I characterize the computation time and communication load of PPC. As defined in Section 3.2, the computation time in private coded computation is the time taken for the workers to return enough number of multiplications so that the master is able to recover the whole computation $\mathbf{AB}_D$. Recalling that the master should receive m multiplications from each group, the computation time of PPC equals the time taken for the slowest group to return m multiplications.

I assume that the straggling model of each worker follows a shifted exponential distribution as in [19] and [61]. In [61], the probability $P_s(t)$ for each worker to return exactly s multiplications by time t was given by

$$P_s(t) = \begin{cases} 0, & t < s\gamma, \\ 1 - e^{-\mu(\frac{t}{s}-\gamma)}, & s\gamma \le t < (s+1)\gamma, \\ e^{-\mu(\frac{t}{s+1}-\gamma)} - e^{-\mu(\frac{t}{s}-\gamma)}, & (s+1)\gamma \le t, \end{cases}$$

where γ and μ denote the shift and straggling parameter, respectively.

According to the analysis in [61], I now derive the expected time consumed for a group G_a to return m multiplications to the master. I denote this time by a random variable T_a. Since each worker in G_a returns up to L multiplications, there are $L+1$ groups of workers in G_a according to the number of returned multiplications by time t. I denote the number of workers in G_a who return s multiplications by time t by $N_s(t)$, where $\sum_{s=0}^{L} N_s(t) = N/n$. Accordingly, if I denote the number of multiplications returned from G_a by time t by $M(t)$,

$$M(t) = \sum_{s=1}^{L} sN_s(t).$$

For a vector $\mathbf{N}(t) = \{N_0(t), \cdots, N_L(t)\}$, the probability of particular $\mathbf{N}(t)$ is given by

$$Pr(\mathbf{N}(t)) = \prod_{s=0}^{L} P_s(t)^{N_s} \binom{N/n - \sum_{j<s} N_j}{N_s}.$$

As a result, the probability $Pr(T_a \leq t)$ is given by

$$Pr(T_a \leq t) = \sum_{\mathbf{N}(t):M(t)\geq m} Pr(\mathbf{N}(t)),$$

and the expected time $E[T_a]$ is given as follows.

$$\begin{aligned} E[T_a] &= \int_0^\infty Pr(T_a > t)dt \\ &= \int_0^\infty \left[1 - \sum_{\mathbf{N}(t):M(t)\geq m} Pr(\mathbf{N}(t))\right] dt. \end{aligned}$$

Consequently, the expected computation time $E[T]$ of PPC is given as follows.

$$E[T] = \max(E[T_1], \cdots, E[T_n]). \tag{3.5}$$

I now characterize the communication load of PPC. Let $|\mathbf{A}|$ denote the amount of communication load for the master to transmit a matrix $\mathbf{A}$ to a worker. As shown in (3.3), the size of $\tilde{\mathbf{A}}(x)$ equals $1/m$ of $\mathbf{A}$, which implies that the communication load for each evaluation of $\tilde{\mathbf{A}}(x)$ equals $\frac{|\mathbf{A}|}{m}$. Since the master sends L evaluations of $\tilde{\mathbf{A}}(x)$ to each of N workers, the communication load of PPC equals $\frac{LN}{m}|\mathbf{A}|$.

3.4.6 Special cases

In this section, I propose special cases of POPC and PAPC.

In POPC, the master sends only one evaluation of $\tilde{\mathbf{A}}(x)$ to each worker. That is, $L = 1$. The first example in Section 3.4.1 corresponds to POPC.

In PAPC, the master divides its own matrix $\mathbf{A}$ into much smaller partitions, which implies that m becomes much larger in turn. If m is sufficiently large, even the slow workers are able to contribute to the master, rather than ignored. As m becomes larger, the lower bound of L also becomes larger, since $L \geq \frac{mn}{N}$ from (3.4). As L becomes larger, faster workers are able to compute and return more multiplications to the master. That is, faster workers are exploited more efficiently in PAPC as compared to POPC where the faster workers are not exploited after returning only one multiplication. Therefore, by properly designing the parameters m and L, all of the workers in each group are able to continue working until they return m multiplications to the master.

As a result, the computation time of PAPC is faster than that of POPC. On the other hand, as L becomes larger, the communication load of PAPC is larger than that of POPC. In fact, since $L = 1$ in POPC, the communication load is minimized at POPC. I will show these results by simulation in the next section.

3.5 Simulation results

In this section, in terms of the computation time and communication load defined in Section 3.2, I compare POPC and PAPC with the optimal RPIR scheme in [53], and PC schemes in [54] and [60]. For a fair comparison, I assume that the workers do not collude with each other in RPIR scheme, and PC schemes in [54] and [60], which is the same as in POPC and PAPC.

The optimal RPIR scheme in [53] is directly applicable to private coded computation. The master encodes the library $\mathbf{B} = \{\mathbf{B}_k\}_{k=1}^{M}$ into $\{\tilde{\mathbf{B}}_k\}_{k=1}^{M}$ with MDS code and each $\tilde{\mathbf{B}}_k$ is multiplied by $\mathbf{A}$. That is, the master should transmit all of the matrix $\mathbf{A}$ to each worker. Encoding $\mathbf{A}$ does not affect the computation time and the communication load since every encoded matrix $\tilde{\mathbf{A}}$ should be multiplied across the workers to exploit the encoded undesired matrices $\{\tilde{\mathbf{B}}_k | k \in [M] \setminus D\}$ as side information.

For the scheme in [54], I consider the PC version of the scheme, which was explained in [54]. For the scheme in [60], I assume the desired function is a linear combination. I consecutively apply these schemes so that the coefficient of desired matrix $\mathbf{B}_D$ corresponds to an element in $\mathbf{A}$ for each iteration.

3.5.1 Computation time

As assumed in Section 4.3.3, time distribution of each worker follows shifted exponential distribution. When computing $\mathbf{A}\mathbf{B}_D$ without considering the master's privacy as in [19], the computation time equals the time taken for the Kth fastest worker to compute and return $1/K$ of whole computation, where the slowest $N-K$ stragglers are mitigated and $K \in [N]$. Therefore, the computation time is the Kth statistic of N independent exponential random variables. If I denote a sum $\sum_{n=1}^{N}\frac{1}{n}$ by H_N, $H_N \simeq \log N$ for large N. Since the expected value of the Kth statistic of N independent exponential random variable is given by $\frac{H_N - H_{N-K}}{\mu}$, the expected computation time of the conventional coded computation is given by

$$E[T_{conv}] = \frac{1}{K}(\gamma + \frac{1}{\mu}\text{log}\frac{N}{N-K}),$$

where γ and μ are the shift and the straggling parameters, respectively, as denoted in Section 4.3.3.

Similar to the conventional coded computation, in the RPIR scheme, when the effects of at most $N-K$ stragglers are mitigated, the computation time is determined by the Kth fastest worker. In order to recover $\mathbf{AB}_D$ under the privacy constraint, the RPIR scheme requires $1+\frac{1}{K}+\cdots+\frac{1}{K^{M-1}}$ times more computation than that required by directly computing $\mathbf{AB}_D$ without considering the master's privacy. Therefore, the expected computation time of RPIR scheme which is denoted by $E[T_{RPIR}]$ takes $(1+\frac{1}{K}+\cdots+\frac{1}{K^{M-1}})$ times longer than $E[T_{conv}]$, which is given by

$$\begin{aligned} E[T_{RPIR}] &= (1+\frac{1}{K}+\cdots+\frac{1}{K^{M-1}})\frac{1}{K}(\gamma + \frac{1}{\mu}\text{log}\frac{N}{N-K}) \\ &= (\frac{1}{K}+\cdots+\frac{1}{K^{M}})(\gamma + \frac{1}{\mu}\text{log}\frac{N}{N-K}). \end{aligned}$$

Similarly, the expected computation times of schemes in [54] and [60] are given by

$$E[T_{PC1}] = \frac{1 - \frac{1}{(2K-1)^{M(K-1)}}}{K-1}(\gamma + \frac{1}{\mu}\log K),$$

$$E[T_{PC2}] = \frac{1}{1 - \frac{M+K}{N}}\frac{1}{K}(\gamma + \frac{1}{\mu}\log\frac{N}{N-K}),$$

where $E[T_{PC1}]$ and $E[T_{PC2}]$ correspond to the expected computation time of the scheme in [54] and [60], respectively.

I compare the computation time between three schemes for three sets of parameters $(N, M, \gamma) = (12, 4, 0.1), (12, 8, 0.1), (12, 4, 1)$. I vary μ from 10^{-1} to 10 for each set of parameters. I assume the number of groups in both of POPC and the PAPC is two, thus implying that $n = 2$. For POPC, mn should be less than N from (3.4). Therefore, for $n = 2$, $m \in [6]$ in POPC. I choose the best m that minimizes the computation time of POPC. For PAPC, I set $L = m = 100$ so that every worker in each group G_t continues working until m multiplication are returned from G_t. For the optimal RPIR scheme and PC schemes in [54] and [60], similar to POPC, I choose the best K that minimizes the computation time, where $K \in [N]$.

I employ the Monte-Carlo method and the comparison results are given in Fig. 3.6, 3.7, and 3.8. As shown in the figures, PAPC outperforms the other schemes when γ is relatively small. As explained, this is because the workers are exploited more efficiently in PAPC compared to the other schemes when γ is relatively small. On the other hand, the gap between the five schemes decreases as μ increases. This is reasonable because the delaying effect by slow workers becomes negligible as μ increases. Although POPC achieves the worst computation time for small μ, it outperforms the

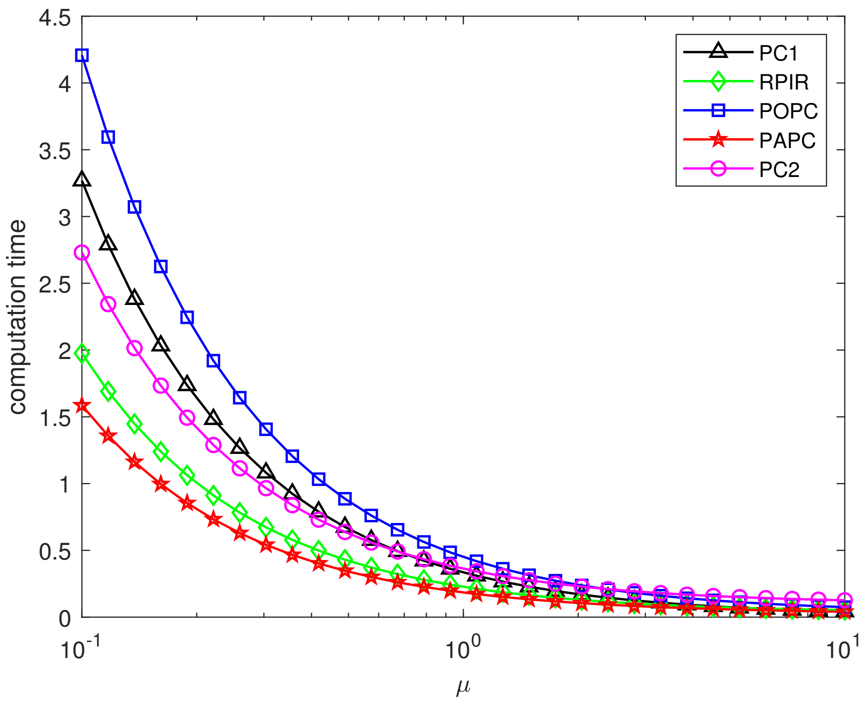


Figure 3.6: The computation time comparison for $N = 12$, $M = 4$, $\gamma = 1$, and varying μ from 10^{-1} to 10.

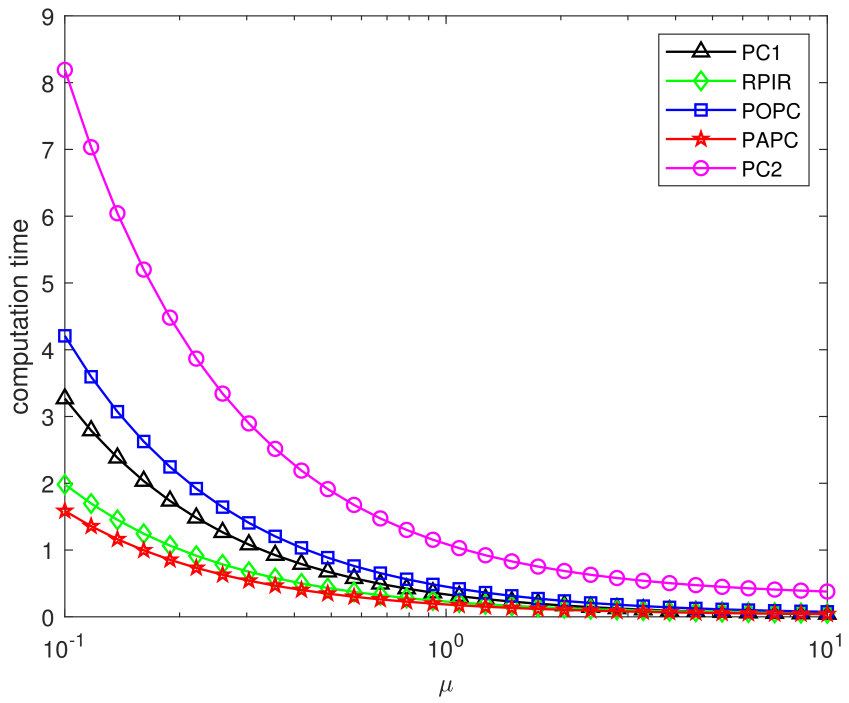


Figure 3.7: The computation time comparison for $N = 12$, $M = 8$, $\gamma = 0.1$, and varying μ.

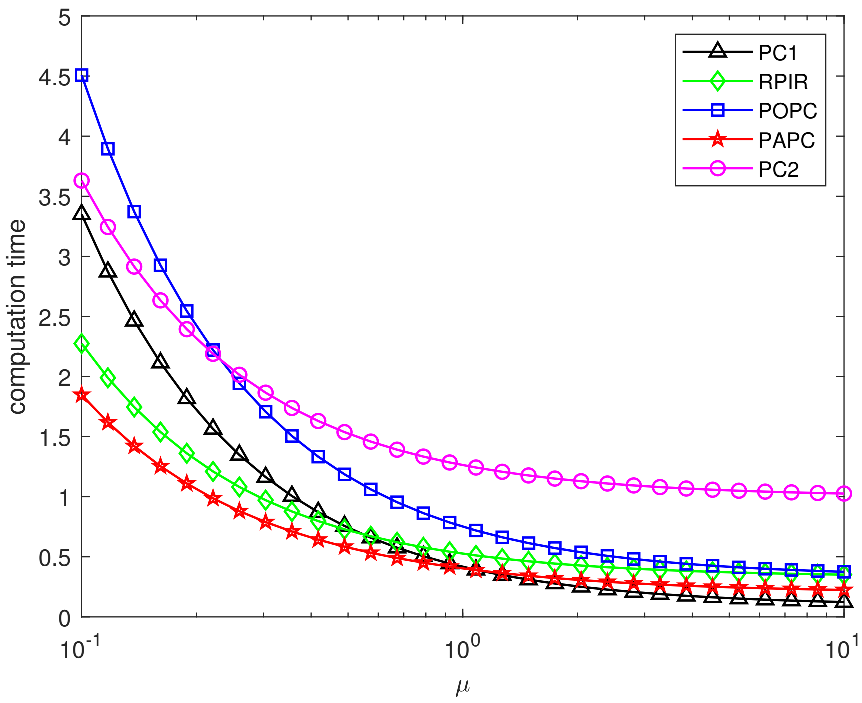


Figure 3.8: The computation time comparison for $N = 12$, $M = 4$, $\gamma = 1$, and varying μ.

PC scheme in [60] as μ becomes larger.

For the PC schemes in [54] and [60], their computation times are worse than the optimal RPIR scheme and PAPC when γ is relatively small. This is because the master's privacy in [54] and [60] is stronger than private coded computation problem, as explained in Section 3.1.

3.5.2 Communication load

For the communication load, I generally characterize the communication load of each scheme and compare them. The communication loads of the five schemes are summarized in Table. 3.1. As characterized in Section 4.3.3, the communication load of PPC equals $\frac{LN}{m}|\mathbf{A}|$. Although L is fixed to 1 in POPC, it should be designed in PAPC so that every worker in a group continues working until m multiplications are returned to the master. Considering the extreme case where the fastest worker in each group solely returns m multiplications before other workers in the same group return a multiplication, L is at most m in PAPC. As a result, the communication load in PAPC is upper bounded by $N|\mathbf{A}|$ which is m times larger than that of POPC. Note that m is the parameter in POPC and upper bounded by N/n from (3.4).

Let me characterize the communication load of the optimal RPIR scheme and PC schemes in [54] and [60]. As explained, in the optimal RPIR scheme, the whole content of $\mathbf{A}$ should be transmitted to each worker regardless of whether the master encodes $\mathbf{A}$ or not. Therefore, the communication load of the optimal RPIR scheme equals $N|\mathbf{A}|$

Table 3.1: The communication load comparison

Communication load	
U_{RPIR}	$N\|\mathbf{A}\|$
U_{PC1}	$N\|\mathbf{A}\|$
U_{PC2}	$N\|\mathbf{A}\|$
U_{POPC}	$\frac{N}{m}\|\mathbf{A}\|$
U_{PAPC}	$\frac{LN}{m}\|\mathbf{A}\|$

which is the same as the upper bound of PAPC. For the PC schemes in [54] and [60], since the encoding $\mathbf{A}$ at the master side is not considered, the whole content of $\mathbf{A}$ should be transmitted to each worker, same as the optimal RPIR scheme. As a result, if I denote the communication load of each scheme by U_{RPIR}, U_{PC1}, U_{PC2}, U_{POPC}, U_{PAPC}, respectively, the following holds in general:

$$U_{RPIR} = U_{PC1} = U_{PC2} \geq U_{PAPC} > U_{POPC}.$$

3.6 Conclusion

In this chapter, I considered private coded computation as a variation of coded computation that ensures the master's privacy. As an achievable scheme for private coded computation, I proposed PPC based on the scheme in [39]. As special cases of PPC, I characterized POPC and PAPC. While PAPC achieved faster computation time, POPC achieved smaller communication load. I compared POPC and PAPC with the optimal RPIR scheme and PC schemes in [54] and [60], and verified that the proposed schemes outperform the conventional schemes either in terms of computation time or communication load. In future work, I may use different codes in order to improve the performance. For example, a scheme in [62] which is a generalized version of scheme in [39], can be applied to PPC.

Chapter 4

Private Secure Coded Computation

4.1 Introduction

In this chapter, I present *private secure coded computation* that ensures both the master's privacy and data security against the workers, as a new variation of coded computation. I consider the security in secure coded computation and the master's privacy in private coded computation. Specifically, I consider a coded computation. The overall process of private secure coded computation is depicted in Fig. 4.1. There are N non-colluding workers $\{\mathrm{W}_i\}_{i=1}^{N}$ who share a library $\mathbf{B}$ of M matrices $\mathbf{B}_1, \cdots, \mathbf{B}_M$. The master must compute $\mathbf{A}\mathbf{B}_D$. Let g_{W_i} and h_{W_i} denote encoding functions of the master for and the worker W_i itself, respectively. The master encodes its private matrix $\mathbf{A}$ into $g_{\mathrm{W}_i}(\mathbf{A}) = C_i$, and sends to W_i. The master also sends the queries Q_i, and the worker W_i encodes $\mathbf{B}$ into $h_{\mathrm{W}_i}(\mathbf{B})$ and computes $g_{\mathrm{W}_i}(\mathbf{A})h_{\mathrm{W}_i}(\mathbf{B}) = Y_i$ according to Q_i. The master recovers $\mathbf{A}\mathbf{B}_D$ from $Y_1, \cdots, Y_N$ with a decoding function d. After the master

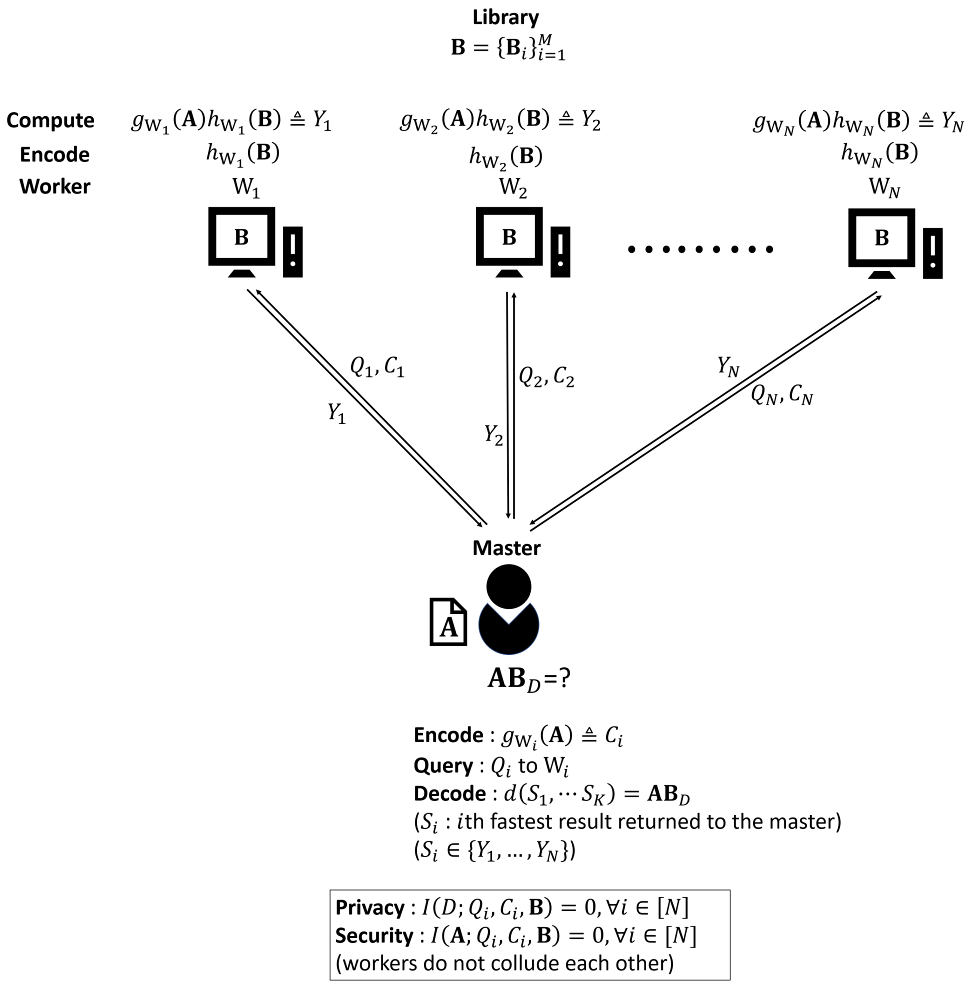

Library
$\mathbf{B} = \{\mathbf{B}_i\}_{i=1}^{M}$
Compute
Encode
Worker
$g_{W_1}(\mathbf{A})h_{W_1}(\mathbf{B}) \triangleq Y_1$
$h_{W_1}(\mathbf{B})$
W_1
$g_{W_2}(\mathbf{A})h_{W_2}(\mathbf{B}) \triangleq Y_2$
$h_{W_2}(\mathbf{B})$
W_2
$g_{W_N}(\mathbf{A})h_{W_N}(\mathbf{B}) \triangleq Y_N$
$h_{W_N}(\mathbf{B})$
W_N
B
Q_1, C_1
Y_1
Q_2, C_2
Y_2
Y_N
Q_N, C_N
Master
A
$\mathbf{AB}_D$=?
Encode : $g_{W_i}(\mathbf{A}) \triangleq C_i$
Query : Q_i to W_i
Decode : $d(S_1, \cdots S_K) = \mathbf{AB}_D$
(S_i : ith fastest result returned to the master)
($S_i \in \{Y_1, \ldots, Y_N\}$)
Privacy : $I(D; Q_i, C_i, \mathbf{B}) = 0, \forall i \in [N]$
Security : $I(\mathbf{A}; Q_i, C_i, \mathbf{B}) = 0, \forall i \in [N]$
(workers do not collude each other)

Figure 4.1: The overall process of private secure coded computation

recovers $\mathbf{AB}_D$, the workers must not identify the index D of desired matrix $\mathbf{B}_D$, as well as they should not obtain any information about $\mathbf{A}$. These privacy and security constraints are expressed as

$$I(D; Q_i, C_i, \mathbf{B}) = 0, \text{(privacy)} \tag{4.1}$$

$$I(\mathbf{A}; Q_i, C_i, \mathbf{B}) = 0. \text{ (security)} \tag{4.2}$$

I present a motivating example of the private secure coded computation. Similar to the motivating example for private coded computation, there is a mobile user who uses artificial intelligence (AI) assistant. The assistant provides a recommendation service and there are M categories that the user can choose, e.g. movies, games, restaurants, and so on. The data matrices for M categories are denoted by $\{\mathbf{B}_k\}_{k=1}^{M}$ and they comprise the library $\mathbf{B}$. I assume that the library is not owned by the user but the external workers which are controlled by the assistant. With its preference parameter matrix $\mathbf{A}$, the user orders the assistant to recommend an item in a specific category $\mathbf{B}_D$ where $D \in [M]$. Note that the user encrypts $\mathbf{A}$ so that the assistant acquires no information of $\mathbf{A}$, thus implying that the data security on $\mathbf{A}$ is ensured. When ordered, the assistant instructs the external workers to compute a multiplication $\mathbf{AB}_D$ and decides its recommendation based on $\mathbf{AB}_D$. For the user's privacy, the assistant does not reveal the index of the aimed matrix D to the workers and the workers also should not identify D from the assistant's instructions.

As explained before, the user usually uses the recommendation service according to its life style. Therefore, if the workers can identify D and record the timeline, they

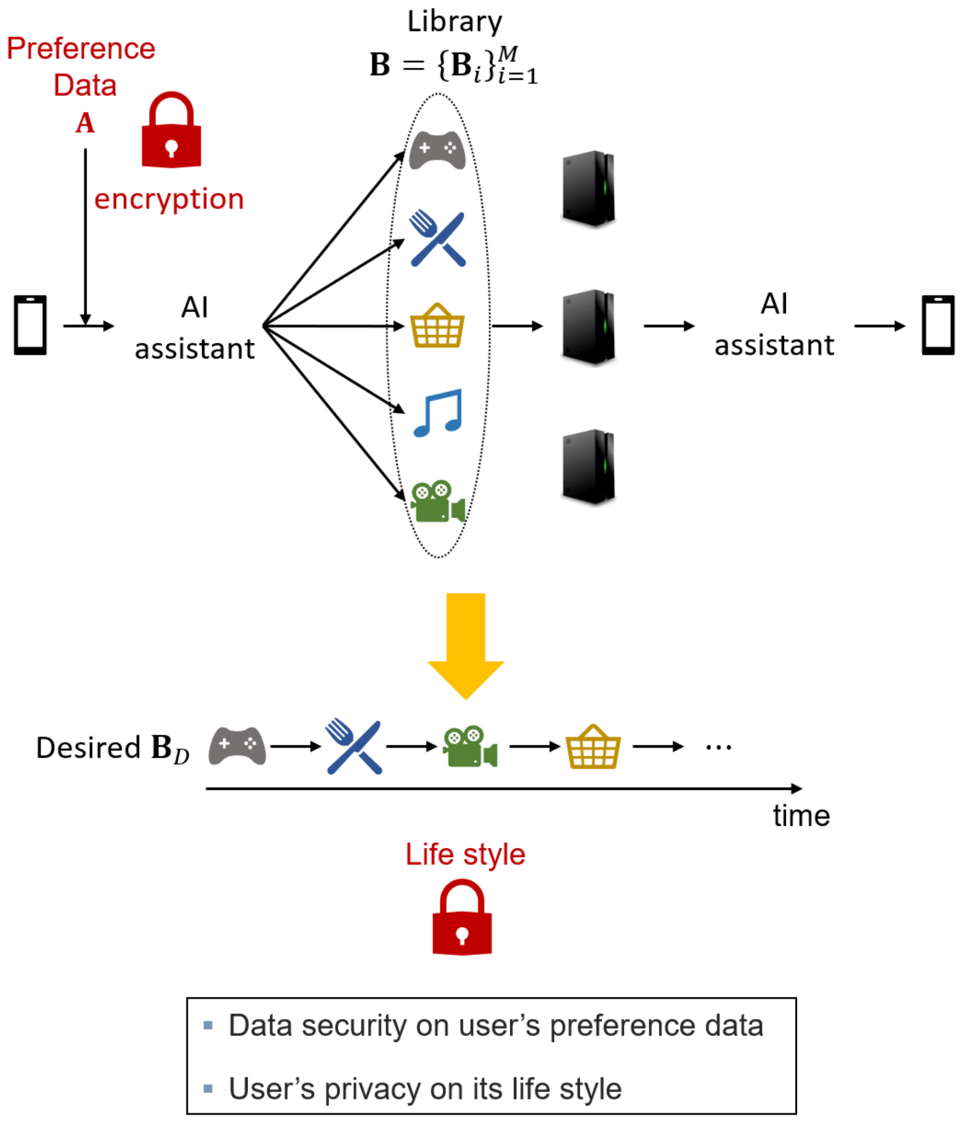

Preference Data $\mathbf{A}$
encryption
Library $\mathbf{B} = \{\mathbf{B}_i\}_{i=1}^{M}$
AI assistant
AI assistant
Desired $\mathbf{B}_D$
…
time
Life style
Data security on user's preference data
User's privacy on its life style

Figure 4.2: Motivating example of private secure coded computation

acquire the information about the user's life style and invade the user's privacy on its life style. This motivating example is depicted in Fig. 4.2.

As an achievable scheme, I propose *private secure polynomial codes* (PSPC), based on *polynomial codes* [39]. I compare the proposed scheme with uncoded scheme, *private polynomial codes* (PPC) in [55], and the optimal scheme of robust *private information retrieval* (RPIR) [53], with respect to computation time and communication load which are defined in previous chapter.

4.2 Main results

In this section, I state main results, which are achievable scheme and proofs for privacy and security.

Theorem 1 (achievable scheme)**.** *In an achievable scheme for a private secure coded computation system with a library $\mathbf{B}$ of M matrices $\{\mathbf{B}\}_{i=1}^{M}$ and N workers $\{W\}_{i=1}^{N}$, where a master has its own matrix $\mathbf{A}$ and a random matrix $\mathbf{R}$, the master's encoding of $\mathbf{A}$ for W_i is given by*

$$\tilde{\mathbf{A}}(x_i) = \sum_{l=0}^{m-1} \mathbf{A}_l x_i^l + \mathbf{R}x_i^m,$$

the encoding of $\mathbf{B}$ by the worker W_i is given by

$$\tilde{\mathbf{B}}(x_i) = \tilde{\mathbf{B}}_D(x_i) + \sum_{k\in[M]\setminus D} \tilde{\mathbf{B}}_k(x_{j_k}) = \sum_{l=1}^{n-1} \mathbf{B}_{D,l} x_i^l + \sum_{k\in[M]\setminus D} \sum_{l=1}^{n-1} \mathbf{B}_{k,l} x_{j_k}^l,$$

and the computation returned from the worker W_i is given by

$$\tilde{\mathbf{A}}(x_i)\tilde{\mathbf{B}}(x_i),$$

where

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_0 \\ \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_{m-1} \end{bmatrix},$$

$$\mathbf{B}_k = \begin{bmatrix} \mathbf{B}_{k,1} & \cdots & \mathbf{B}_{k,n-1} \end{bmatrix}, k \in [M].$$

Theorem 2 (privacy and security proof)**.** *For the achievable scheme given in Theorem 1, the following privacy and security constraints for each worker* W_i *are satisfied.*

$$I(D; Q_i, C_i, \mathbf{B}) = 0, \forall i \in [N],$$

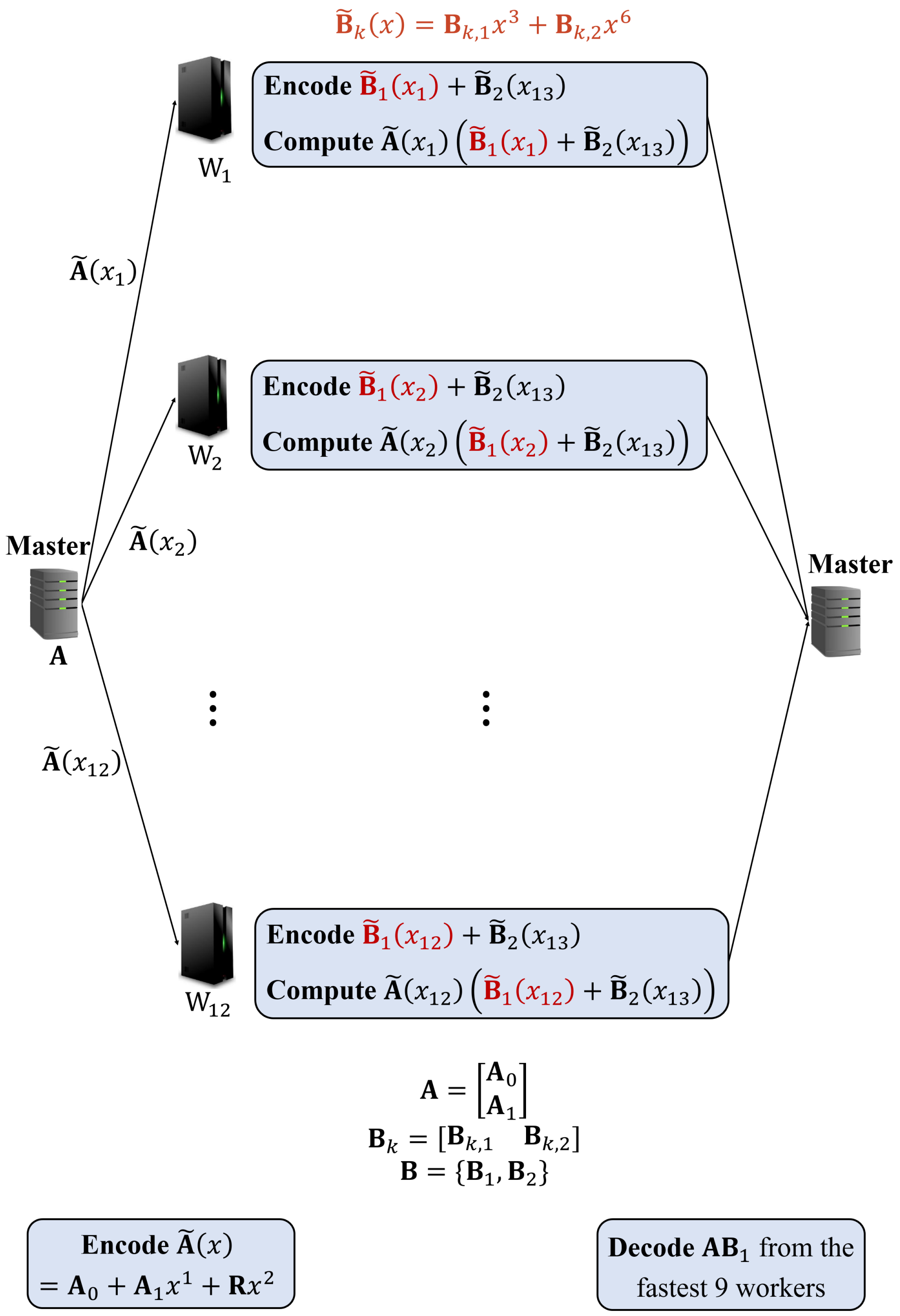
$$I(\mathbf{A}; Q_i, C_i, \mathbf{B}) = 0, \forall i \in [N],$$

where D*,* Q_i*, and* C_i *denote the desired index, the queries for* W_i*, and encoded* $\mathbf{A}$ *sent to* W_i*, respectively.*

4.3 Private secure polynomial codes

4.3.1 Illustrative example

The overall process is depicted in Fig. 4.3. A master's private matrix is denoted by $\mathbf{A} \in \mathbb{F}_q^{r\times s}$ for sufficiently large finite field $\mathbb{F}_q$. There are 12 non-colluding workers $\{\mathrm{W}_i\}_{i=1}^{12}$ who do not communicate with each other and share a library $\mathbf{B}$ of two matrices $\mathbf{B}_1, \mathbf{B}_2 \in \mathbb{F}_q^{s\times t}$. The master needs to compute $\mathbf{AB}_1$. For the matrices $\mathbf{A}_0, \mathbf{A}_1 \in$



$\widetilde{\mathbf{B}}_k(x) = \mathbf{B}_{k,1}x^3 + \mathbf{B}_{k,2}x^6$
Encode $\widetilde{\mathbf{B}}_1(x_1) + \widetilde{\mathbf{B}}_2(x_{13})$
Compute $\widetilde{\mathbf{A}}(x_1)\left(\widetilde{\mathbf{B}}_1(x_1) + \widetilde{\mathbf{B}}_2(x_{13})\right)$
W_1
$\widetilde{\mathbf{A}}(x_1)$
Encode $\widetilde{\mathbf{B}}_1(x_2) + \widetilde{\mathbf{B}}_2(x_{13})$
Compute $\widetilde{\mathbf{A}}(x_2)\left(\widetilde{\mathbf{B}}_1(x_2) + \widetilde{\mathbf{B}}_2(x_{13})\right)$
W_2
Master
$\widetilde{\mathbf{A}}(x_2)$
Master
$\mathbf{A}$
$\widetilde{\mathbf{A}}(x_{12})$
Encode $\widetilde{\mathbf{B}}_1(x_{12}) + \widetilde{\mathbf{B}}_2(x_{13})$
Compute $\widetilde{\mathbf{A}}(x_{12})\left(\widetilde{\mathbf{B}}_1(x_{12}) + \widetilde{\mathbf{B}}_2(x_{13})\right)$
W_{12}
$\mathbf{A} = \begin{bmatrix}\mathbf{A}_0 \\ \mathbf{A}_1\end{bmatrix}$
$\mathbf{B}_k = [\mathbf{B}_{k,1} \quad \mathbf{B}_{k,2}]$
$\mathbf{B} = \{\mathbf{B}_1, \mathbf{B}_2\}$
Encode $\widetilde{\mathbf{A}}(x) = \mathbf{A}_0 + \mathbf{A}_1 x^1 + \mathbf{R}x^2$
Decode $\mathbf{AB}_1$ from the fastest 9 workers

Figure 4.3: The overall process of the illustrative example

$\mathbb{F}_q^{r/2 \times s}$ and $\mathbf{B}_{k,1}, \mathbf{B}_{k,2} \in \mathbb{F}_q^{s \times t/2}$, $k \in [2]$, I can express

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_0 \\ \\ \mathbf{A}_1 \end{bmatrix},$$

$$\mathbf{B}_k = \begin{bmatrix} \mathbf{B}_{k,1} & \mathbf{B}_{k,2} \end{bmatrix}.$$

At first, the master randomly chooses 13 distinct points $\{x_i\}_{i=1}^{13}$ in $\mathbb{F}_q$. For each worker W_i, the master computes and transmits $\tilde{\mathbf{A}}(x_i)$ which is given by

$$\tilde{\mathbf{A}}(x_i) = \mathbf{A}_0 + \mathbf{A}_1 x_i + \mathbf{R} x_i^2,$$

where $\mathbf{R} \in \mathbb{F}_q^{r/2 \times s}$ is a random matrix for ensuring data security on $\mathbf{A}$.

With $\tilde{\mathbf{A}}(x_i)$, the master also transmits queries Q_i that order W_i to encode the library $\mathbf{B}$ into

$$\tilde{\mathbf{B}}(x_i) = \tilde{\mathbf{B}}_1(x_i) + \tilde{\mathbf{B}}_2(x_{13}),$$

where $\tilde{\mathbf{B}}_k(x)$ is given by

$$\tilde{\mathbf{B}}_k(x) = \mathbf{B}_{k,1} x^3 + \mathbf{B}_{k,2} x^6, \forall k \in [2].$$

Each worker W_i computes and returns a multiplication which is given by

$$\begin{aligned} &\tilde{\mathbf{A}}(x_i)\tilde{\mathbf{B}}(x_i) \\ &= (\mathbf{A}_0 + \mathbf{A}_1 x_i + \mathbf{R} x_i^2) \times \\ &(\mathbf{B}_{1,1} x_i^3 + \mathbf{B}_{1,2} x_i^6 + \mathbf{B}_{2,1} x_{13}^3 + \mathbf{B}_{2,2} x_{13}^6) \\ &= \sum_{l=0}^{8} \mathbf{Z}_l x_i^l, \end{aligned}$$

where $\{\mathbf{Z}_l\}_{l=0}^{8}$ are given by

$$\mathbf{Z}_0 = \mathbf{A}_0(\mathbf{B}_{2,1}x_{13}^3 + \mathbf{B}_{2,2}x_{13}^6),$$

$$\mathbf{Z}_1 = \mathbf{A}_1(\mathbf{B}_{2,1}x_{13}^3 + \mathbf{B}_{2,2}x_{13}^6),$$

$$\mathbf{Z}_2 = \mathbf{R}(\mathbf{B}_{2,1}x_{13}^3 + \mathbf{B}_{2,2}x_{13}^6),$$

$$\mathbf{Z}_3 = \mathbf{A}_0\mathbf{B}_{1,1},$$

$$\mathbf{Z}_4 = \mathbf{A}_1\mathbf{B}_{1,1},$$

$$\mathbf{Z}_5 = \mathbf{R}\mathbf{B}_{1,1},$$

$$\mathbf{Z}_6 = \mathbf{A}_0\mathbf{B}_{1,2},$$

$$\mathbf{Z}_7 = \mathbf{A}_1\mathbf{B}_{1,2},$$

$$\mathbf{Z}_8 = \mathbf{R}\mathbf{B}_{1,2}.$$

Since the degree of polynomial $\tilde{\mathbf{A}}(x)\tilde{\mathbf{B}}(x)$ is 8 and 13 points $\{x_i\}_{i=1}^{13}$ are distinct from each other, the master can interpolate the polynomial after the 9 fastest workers return their results and the multiplication $\mathbf{A}\mathbf{B}_1$ can be recovered from the coefficients $\mathbf{Z}_3$, $\mathbf{Z}_4$, $\mathbf{Z}_6$, and $\mathbf{Z}_7$.

Remark 1. In PSPC, I may consider some adversarial workers who return erroneous results, as well as the stragglers. For example, in the above example, I assume that up to one worker is an adversarial. In this case, the master needs to employ Reed-Solomon decoding instead of polynomial interpolation. Moreover, the master needs two more multiplications for decoding, which delays the process. In general, if there are E adversarial workers, the master has to receive $2E$ more multiplications from the workers.

4.3.2 General description

The overall process of PSPC is depicted in Fig. 4.4. There are N non-colluding workers $\{\mathrm{W}_i\}_{i=1}^{N}$ who do not communicate with each other and share a library $\mathbf{B}$ of M matrices $\{\mathbf{B}_k\}_{k=1}^{M}$ where each $\mathbf{B}_k \in \mathbb{F}_q^{s\times t}$ for sufficiently large finite field $\mathbb{F}_q$. For its private matrix $\mathbf{A} \in \mathbb{F}_q^{r\times s}$, the master needs to compute $\mathbf{A}\mathbf{B}_D$, where $D \in [M]$. For the matrices $\{\mathbf{A}_l\}_{l=0}^{m-1} \in \mathbb{F}_q^{r/m\times s}$ and $\{\mathbf{B}_{k,l}\}_{l=1}^{n} \in \mathbb{F}_q^{s\times t/n}$, $k \in [M]$, I can express

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_0 \\ \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_{m-1} \end{bmatrix},$$

$$\mathbf{B}_k = \begin{bmatrix} \mathbf{B}_{k,1} \cdots \mathbf{B}_{k,n} \end{bmatrix}.$$

The master randomly chooses $N + M - 1$ distinct points $\{x_i\}_{i=1}^{N}$ and $\{x_{j_k} | k \in [M] \setminus D\}$ in $\mathbb{F}_q$. For each worker W_i, the master computes and transmits $\tilde{\mathbf{A}}(x_i)$ which is given by

$$\tilde{\mathbf{A}}(x_i) = \sum_{l=0}^{m-1} \mathbf{A}_l x_i^l + \mathbf{R} x_i^m, \tag{4.3}$$

where $\mathbf{R} \in \mathbb{F}_q^{r/m\times s}$ is a random matrix for ensuring data security on $\mathbf{A}$.

With $\tilde{\mathbf{A}}(x_i)$, the master also transmits queries Q_i that order W_i to encode the

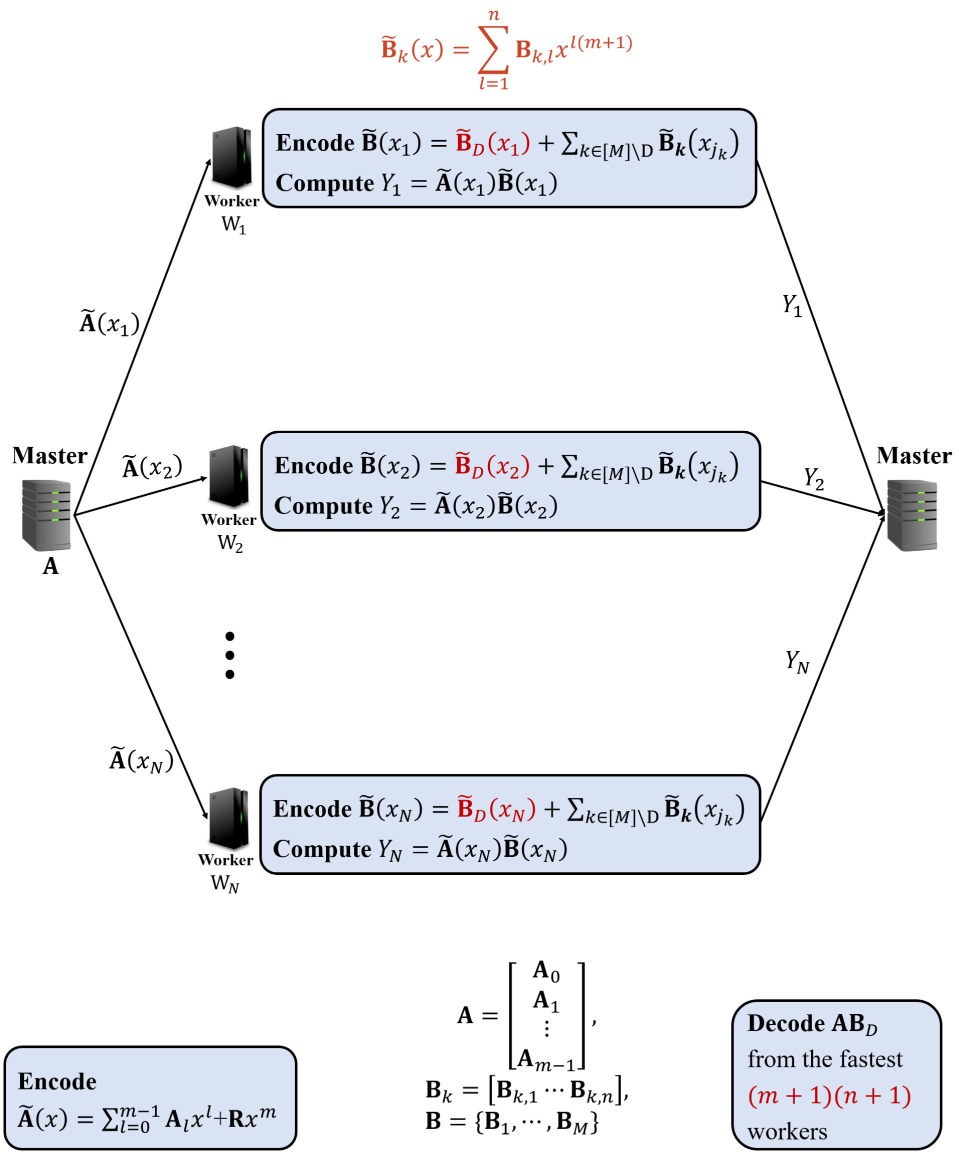

$\widetilde{\mathbf{B}}_k(x) = \sum_{l=1}^{n} \mathbf{B}_{k,l} x^{l(m+1)}$
Encode $\widetilde{\mathbf{B}}(x_1) = \widetilde{\mathbf{B}}_D(x_1) + \sum_{k\in[M]\setminus D} \widetilde{\mathbf{B}}_k(x_{j_k})$
Compute $Y_1 = \widetilde{\mathbf{A}}(x_1)\widetilde{\mathbf{B}}(x_1)$
Worker W_1
$\widetilde{\mathbf{A}}(x_1)$
Y_1
Master
$\mathbf{A}$
$\widetilde{\mathbf{A}}(x_2)$
Encode $\widetilde{\mathbf{B}}(x_2) = \widetilde{\mathbf{B}}_D(x_2) + \sum_{k\in[M]\setminus D} \widetilde{\mathbf{B}}_k(x_{j_k})$
Compute $Y_2 = \widetilde{\mathbf{A}}(x_2)\widetilde{\mathbf{B}}(x_2)$
Worker W_2
Y_2
Master
Y_N
$\widetilde{\mathbf{A}}(x_N)$
Encode $\widetilde{\mathbf{B}}(x_N) = \widetilde{\mathbf{B}}_D(x_N) + \sum_{k\in[M]\setminus D} \widetilde{\mathbf{B}}_k(x_{j_k})$
Compute $Y_N = \widetilde{\mathbf{A}}(x_N)\widetilde{\mathbf{B}}(x_N)$
Worker W_N
$\mathbf{A} = \begin{bmatrix} \mathbf{A}_0 \\ \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_{m-1} \end{bmatrix}$,
$\mathbf{B}_k = [\mathbf{B}_{k,1} \cdots \mathbf{B}_{k,n}]$,
$\mathbf{B} = \{\mathbf{B}_1, \cdots, \mathbf{B}_M\}$
Encode
$\widetilde{\mathbf{A}}(x) = \sum_{l=0}^{m-1} \mathbf{A}_l x^l + \mathbf{R}x^m$
Decode $\mathbf{AB}_D$
from the fastest
$(m+1)(n+1)$
workers

Figure 4.4: The overall process of PSPC.

library $\mathbf{B}$ into

$$\tilde{\mathbf{B}}(x_i) = \tilde{\mathbf{B}}_D(x_i) + \sum_{k\in[M]\backslash D} \tilde{\mathbf{B}}_k(x_{j_k}), \tag{4.4}$$

where $\tilde{\mathbf{B}}_k(x)$ is given by

$$\tilde{\mathbf{B}}_k(x) = \sum_{l=1}^{n} \mathbf{B}_{k,l} x^{l(m+1)}, \forall k \in [M]. \tag{4.5}$$

Each worker W_i computes and returns a multiplication which is given by

$$\begin{aligned}
&\tilde{\mathbf{A}}(x_i)\tilde{\mathbf{B}}(x_i) \\
&= \tilde{\mathbf{A}}(x_i)\tilde{\mathbf{B}}_D(x_i) + \tilde{\mathbf{A}}(x_i)\sum_{k\in[M]\backslash D}\tilde{\mathbf{B}}_k(x_{j_k}) \\
&= \sum_{l=0}^{m-1}\sum_{p=1}^{n}\mathbf{A}_l\mathbf{B}_{D,p}x^{l+p(m+1)} + \sum_{p=1}^{n}\mathbf{R}\mathbf{B}_{D,p}x^{pm+m+p} \\
&+ \sum_{l=0}^{m-1}\sum_{k\in[M]\backslash D}\mathbf{A}_l\tilde{\mathbf{B}}_k(x_{j_k})x^l + \sum_{k\in[M]\backslash D}\mathbf{R}\tilde{\mathbf{B}}_k(x_{j_k})x^m \\
&= \sum_{l=0}^{mn+m+n}\mathbf{Z}_l x^l,
\end{aligned}$$

where $\{\mathbf{Z}_l\}_{l=0}^{mn+m+n}$ are given by

$$\mathbf{Z}_l = \sum_{k\in[M]\backslash D}\mathbf{A}_l\tilde{\mathbf{B}}_k(x_{j_k}) \ \ \forall l \in [0:m-1],$$

$$\mathbf{Z}_l = \sum_{k\in[M]\backslash D}\mathbf{R}\tilde{\mathbf{B}}_k(x_{j_k}) \ \ \forall l = m,$$

$$\mathbf{Z}_l = \mathbf{R}\mathbf{B}_{D,l} \ \ \forall l = m + p(m+1), p \in [n],$$

$$\mathbf{Z}_l = \mathbf{A}_{l-p(m+1)}\mathbf{B}_{D,p} \ \ \forall l \in [p(m+1) : p(m+1)+m-1],$$

$$p \in [n].$$

Since the degree of polynomial $\tilde{\mathbf{A}}(x)\tilde{\mathbf{B}}(x)$ is $mn + m + n$ and the points $\{x_i\}_{i=1}^N$ are distinct from each other, the master can interpolate the polynomial after the $(m + 1)(n + 1)$ fastest workers return their results and the multiplication $\mathbf{AB}_D$ can be recovered from the coefficients $\{\mathbf{Z}_l | l \in [p(m + 1) : p(m + 1) + m - 1], p \in [n]\}$.

Remark 2. While the conventional polynomial codes in [39] uses the term

$$\tilde{\mathbf{B}}_k(x) = \sum_{l=0}^{n-1} \mathbf{B}_{k,l} x^{l(m+1)},$$

there is no $\mathbf{B}_{k,0}x^0$ term in PSPC as seen in (4.5). This is because the coefficient of x^0 in polynomial $\tilde{\mathbf{B}}(x)$ should be the undesired term $\sum_{k\in[M]\setminus D} \tilde{\mathbf{B}}_k(x_{j_k})$ in (4.4). If there is $\mathbf{B}_{k,0}x^0$ for each $\mathbf{B}_k$, the desired submatrix $\mathbf{B}_{D,0}$ cannot be obtained by decoding since the undesired term $\sum_{k\in[M]\setminus D} \tilde{\mathbf{B}}_k(x_{j_k})$ is combined with $\mathbf{B}_{D,0}$. That is, in PSPC, I sacrifice the term $\mathbf{B}_{D,0}$ for the master's privacy, which can be seen as privacy cost.

4.3.3 Performance analysis

In this section, I derive the computation time and the communication load of PSPC according to Definition 1 and 2. I assume that the computation time distribution of each worker is independent of each other, and follows the shifted-exponential distribution as in [19]. The computation time T_{PSPC} is given by

$$T_{PSPC} = \frac{1}{(m+1)n}(\gamma + \frac{1}{\mu}\log\frac{N}{N-(m+1)(n+1)}), \tag{4.6}$$

where μ and γ denote the straggling parameter and the shift parameter, respectively.

For the communication load U_{PSPC}, let $|\mathbf{A}|$ denote the communication load for transmitting $\mathbf{A}$. Since the amount of encoded $\mathbf{A}$ for each worker W_i is given by (4.3), I have

$$U_{PSPC} = N|\mathbf{A}|/m. \tag{4.7}$$

Remark 3. Recalling that the master needs $(m+1)(n+1)$ multiplications from N workers, $(m+1)(n+1) \leq N$. Therefore, for reducing U_{PSPC}, the master maximizes m under $m \leq N/(n+1) - 1$. On the other hand, since the parameters μ and γ also affect T_{PSPC}, it is difficult to generally characterize the effects of m and n on T_{PSPC}.

4.3.4 Privacy and security proof

I prove that (3.1) and (3.2) are satisfied for every worker. By the chain rule, I can write (3.1) as follows.

$$I(D; Q_i, C_i, \mathbf{B}) = I(D; Q_i) + I(D; \mathbf{B}|Q_i) + I(D; C_i|Q_i, \mathbf{B})$$

First, let me consider C_i. If the worker W_i recognizes that $\tilde{\mathbf{A}}(x)$ is evaluated at x_i, it will identify the desired index D since $\tilde{\mathbf{A}}(x)$ and $\tilde{\mathbf{B}}_D(x)$ is evaluated at the same point. However, recalling that C_i is encrypted by the random matrix $\mathbf{R}$, the worker W_i cannot infer x_i from C_i, thus implying that $I(D; C_i|Q_i, \mathbf{B}) = 0$. Next, I consider $\mathbf{B}$. Recalling that the master determines D without obtaining any information of $\mathbf{B}$, $\mathbf{B}$ is independent of D, thus implying that $I(D; \mathbf{B}|Q_i^D) = 0$.

Finally, I consider Q_i. The queries Q_i are fourfold: partitioning each $\mathbf{B}_k$ into n

submatrices, evaluating each $\tilde{\mathbf{B}}_k(x)$, summing them into one equation (4.4), and multiplying the equation (4.4) by C_i. Since the others are fixed operations and do not carry any information, I only consider the evaluation operation. As assumed, the master randomly chooses M distinct points x_i and $\{x_{j_k}|k \in [M] \setminus D\}$ for the worker W_i who does not communicate with other workers. As a result, the worker W_i observes the points x_i and $\{x_{j_k}|k \in [M] \setminus D\}$ as indistinguishable random points, thus implying that $I(D; Q_i) = 0$.

I now prove (2). I can write (2) as follows.

$$I(\mathbf{A}; Q_i, C_i, \mathbf{B}) = I(\mathbf{A}; Q_i) + I(\mathbf{A}; \mathbf{B}|Q_i) + I(\mathbf{A}; C_i|Q_i, \mathbf{B})$$

Since C_i is encrypted, $H(\mathbf{A}|C_i) = H(\mathbf{A})$, which implies that $I(\mathbf{A}; C_i|Q_i, \mathbf{B}) = 0$. Recalling that $\mathbf{A}$ and $\mathbf{B}$ are independent, $I(\mathbf{A}; \mathbf{B}|Q_i) = 0$. The queries Q_i for encoding $\mathbf{B}$ are independent of $\mathbf{A}$, which implies that $I(\mathbf{A}; Q_i) = 0$.□

4.4 Simulation results

I compare PSPC, uncoded scheme, private polynomial codes (PPC), and optimal RPIR scheme in [53] with respect to the computation time and communication load. In the uncoded scheme, the master orders each worker to compute M multiplications $\{(\mathbf{A} + \mathbf{R})\mathbf{B}_k\}_{k=1}^{M}$, where $\mathbf{R}$ is a random matrix. After the fastest worker returns M multiplications, the master obtains $\mathbf{AB}_D$.

For PPC, I choose two special cases-*private one-shot polynomial codes* (POPC) and *private asynchronous polynomial codes* (PAPC). In POPC, each worker returns only one matrix multiplication results, same as PSPC. On the other hand, in PAPC, several smaller multiplication tasks are assigned to each worker, and by assigning a sufficient number of multiplications to each worker, every worker continues working throughout the process. For ensuring the data security in POPC and PAPC, the random matrix $\mathbf{R}$ is added when encoding $\mathbf{A}$, similar to PSPC.

The optimal RPIR scheme in [53] minimizes the amount of download when the master does not have $\mathbf{A}$, and merely downloads $\mathbf{B}_D$ while concealing the index D against the workers. I modifies the optimal RPIR scheme for coded computation by considering $\mathbf{A}$ at the master side. Similar to PSPC, for the data security, the master sends $\mathbf{A} + \mathbf{R}$ in the modified optimal RPIR scheme.

4.4.1 Computation time

As in Section 4.3.3, I assume that the computation time distribution of each worker is independent of each other, and follows the shifted-exponential distribution as in [19]. For uncoded scheme, the computation time is given by

$$T_{uncoded} = M(\gamma + \frac{1}{\mu}\log\frac{N}{N-1}).$$

I compare the computation time for given N, M, γ and varying μ. That is, there are three sets of parameters $(N, M, \gamma) = (12, 4, 0.1), (12, 8, 0.1), (12, 4, 0.0001)$, and μ is varied from 10^{-1} to 10 for each set of parameters. Since μ is a straggling parameter, larger μ implies that the effect of stragglers becomes negligible. Let K denote the minimum number of returned multiplications to recover $\mathbf{AB}_D$. For fair comparison,

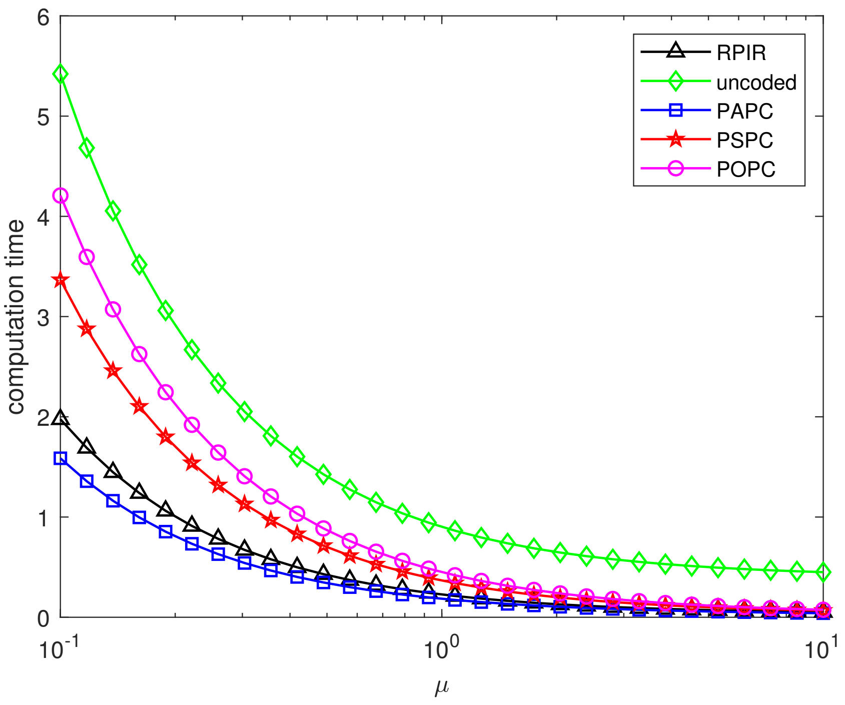


Figure 4.5: The computation time comparison for $N = 12$, $M = 4$, $\gamma = 0.1$, and varying μ.

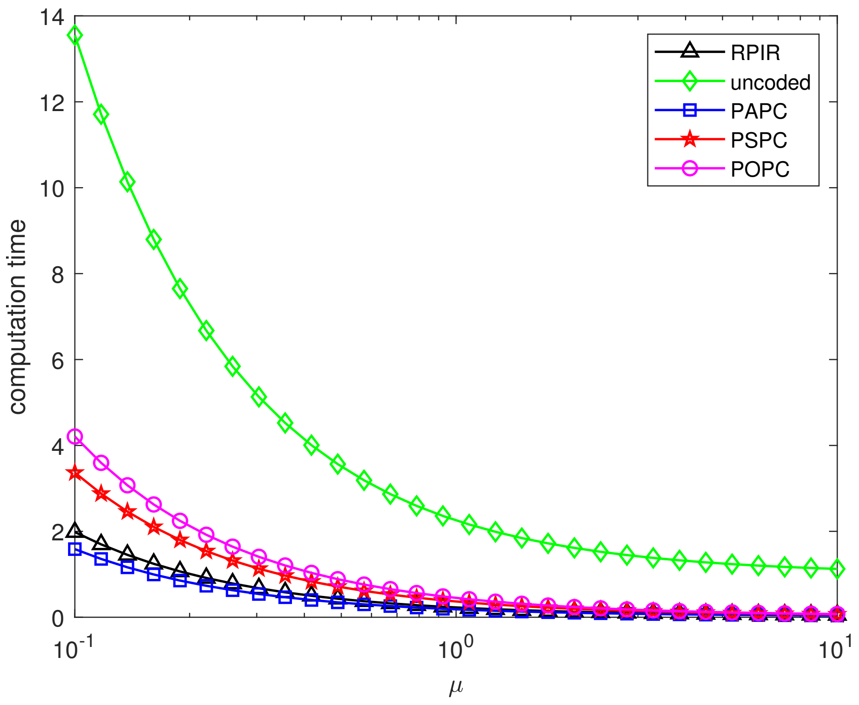


Figure 4.6: The computation time comparison for $N = 12$, $M = 8$, $\gamma = 0.1$, and varying μ.

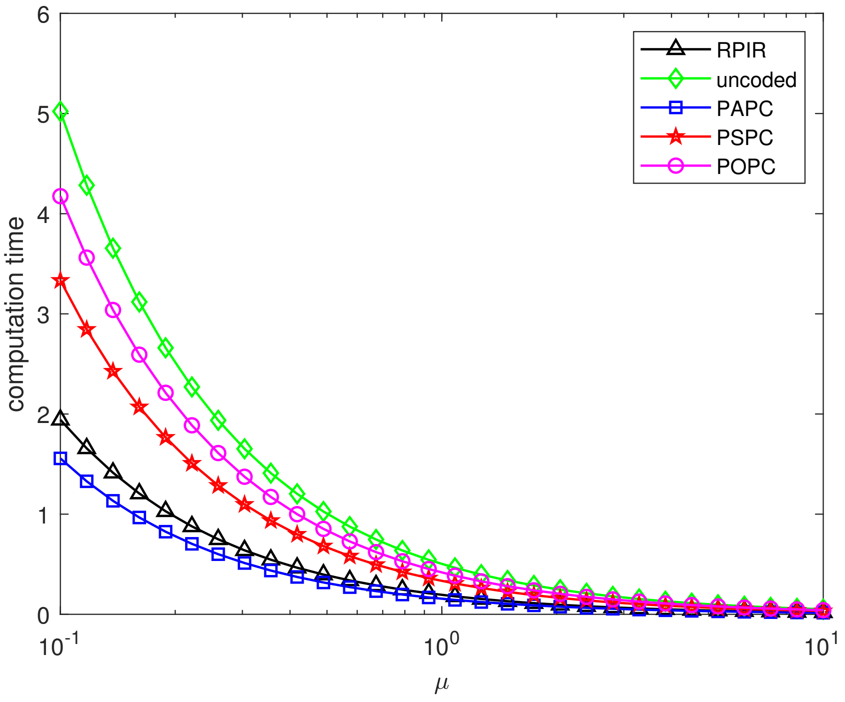


Figure 4.7: The computation time comparison for $N = 12$, $M = 4$, $\gamma = 0.0001$, and varying μ.

I optimize K for each scheme so that each scheme achieves the fastest computation time for given N, M, γ and μ.

As seen in the figures, the uncoded scheme is the worst and PSPC outperforms POPC. This counter-intuitive result is explained in Remark 4. The modified optimal RPIR scheme is the second best. This is because the original optimal RPIR scheme achieves the minimum amount of download when the master privately downloads $\mathbf{B}_D$ without considering $\mathbf{A}$. However, in my problem setting, the master should jointly consider both $\mathbf{A}$ and $\mathbf{B}$. Furthermore, since the modified optimal RPIR scheme is a synchronous scheme, PAPC achieves the fastest computation time where the workers are exploited more efficiently due to its asynchronous manner.

4.4.2 Communication load

Let $U_{uncoded}$, U_{RPIR}, U_{POPC}, and U_{PAPC} denote the communication load of uncoded scheme, optimal RPIR scheme, POPC, and PAPC, respectively. It is obvious that $U_{uncoded} = N|\mathbf{A}|$. According to the analysis given in Section 3.4.5 and 3.5.2, I have $U_{RPIR} = N|\mathbf{A}|$, $U_{POPC} = N|\mathbf{A}|/m$, and $U_{PAPC} = LN|\mathbf{A}|/m$, where L denotes the number of multiplications assigned to each worker in PAPC. Therefore, from (4.7), the following holds generally.

$$U_{uncoded} = U_{RPIR} \geq U_{PAPC} > U_{POPC} = U_{PSPC}.$$

Consequently, PSPC achieves the smallest communication load and strictly faster computation time than POPC.

Remark 4. Unlike PSPC, the workers should be grouped in POPC, which delays the computation time. I explain the cause of grouping in PPC which inlcude POPC. Since $\mathbf{A}$ is not encrypted in the original PPC, the worker W_i may infer x_i from $\tilde{\mathbf{A}}(x_i)$. Therefore, if $\mathbf{B}_D$ is encoded with the same point x_i, the worker W_i identifies D. As a result, in PPC, $\mathbf{A}$ and $\mathbf{B}$ are separately encoded into $\tilde{\mathbf{A}}(x)$ and $\tilde{\mathbf{B}}(y)$, respectively, and the master should decode $\tilde{\mathbf{A}}(x)\tilde{\mathbf{B}}(y)$. Accordingly, the decoding is two-step process: the interpolation of polynomials in x followed by that in y. For decoding $\tilde{\mathbf{A}}(x)$ whose degree is m, there should be at least $m + 1$ distinct evaluations $\{\tilde{\mathbf{A}}(x_i)\tilde{\mathbf{B}}(y_t)\}_{i=1}^{m+1}$ for given y_t. As a result, in PPC, the workers are divided into groups according to the evaluating point of $\tilde{\mathbf{B}}(y)$, and the computation time is determined by the slowest group, which implies that POPC is slower than PSPC.

4.5 Conclusion

In this chapter, I introduced private secure coded computation as a variation of coded computation that protects the data security and the master's privacy at the same time. As an achievable scheme for private secure coded computation, I proposed private secure polynomial codes based on private polynomial codes in private coded computation. By simulation, I compared the private secure polynomial codes and private polynomial codes in terms of computation time, and shown that the proposed scheme outperforms the previous work. In future work, I will further analyze the performance of private secure polynomial codes and compare the performance with actual distributed machines, e.g., AWS or Google Cloud.

Chapter 5

Conclusion

5.1 Summary

In this dissertation, I proposed cache-aided PIR problem, private coded computation, and private secure coded computation. In addition, I proposed an achievable scheme for each system model.

In Chapter 2, cache-aided PIR problem was proposed, as a new variation of PIR problem. By introducing the user's cache in the PIR problem, the download rate was significantly improved. The achievable scheme was based on the optimal scheme for conventional PIR problem. In the achievable scheme, the pre-store cache was exploited as an side information, which improved the download rate, compared to the PIR problem without cache.

In Chapter 3, private coded computation was proposed, as a new variation of coded computation. In the private coded computation, the user should conceal which file the worker's library is desired by the user. This kind of privacy is similar to that of PIR problem. The achievable scheme, namely private polynomial codes, were based on the polynomial codes which was proposed in the conventional coded computation system. In the achievable scheme, the workers are grouped for the privacy and asynchronous scheme was considered, which was not considered in the conventional polynomial codes. Due to the asynchronous scheme, the proposed scheme achieved the faster computation time, compared to the modified optimal RPIR scheme.

In Chapter 4, private secure coded computation was proposed, as a new variation of coded computation. In the private secure coded computation, the user should conceal which file the worker's library is desired by the user. Furthermore, in the private secure coded computation, the user has its own data and should secure its data against the workers. The achievable scheme, namely private secure polynomial codes, were based on the polynomial codes which was proposed in the conventional coded computation system. By modifying the private polynomial codes, the private secure polynomial codes and private secure polynomial codes were compared in terms of computation time and communication load. As a result, the private secure polynomial codes achieved faster computation time for the same communication load.

5.2 Future directions

For the cache-aided PIR problem, the improved achievable scheme was proposed after my work, which is partially optimal. The remaining issue in the cache-aided PIR problem is to find the optimal PIR scheme for entire cache size region. For the private coded computation and private secure coded computation, the improved scheme that achieves the faster computation time or smaller communication load can be proposed. On the other hand, there can be more general achievable scheme for private coded computation and private secure coded computation. For example, by considering the entangled polynomial codes, which are more general than the polynomial codes, there can be more general achievable scheme for private coded computation and private secure coded computation.

Bibliography

[1] M. A. Maddah-Ali and U. Niesen, “Fundamental limits of caching,” *in IEEE Transactions on Information Theory*, vol. 60, no. 5, pp. 2856-2867, Aug. 2014.

[2] M. A. Maddah-Ali and U. Niesen, “Decentralized coded caching attains order-optimal memory-rate tradeoff,” *in IEEE/ACM Transactions on Networking (TON)*, vol. 23, no. 4, pp. 1029-1040, 2015.

[3] N. Golrezaei, K. Shanmugam, A. G. Dimakis, A. F. Molisch, and G. Caire, “Femtocaching: Wireless video content delivery through distributed caching helpers,” *in Proc. IEEE INFOCOM*, pp. 1107–1115, Mar. 2011.

[4] R. Pedarsani, M. A. Maddah-Ali, and U. Niesen, “Online Coded Caching,” *in IEEE/ACM Trans. Netw.*, vol. 24, no. 2, pp. 836–845, Apr. 2016.

[5] N. Karamchandani, U. Niesen, M. A. Maddah-Ali, and S. N. Diggavi, “Hierarchical coded caching,” *in IEEE Trans. Inf. Theory*, vol. 62, no. 6, pp. 3212–3229, Jun. 2016.

[6] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, “The exact rate-memory trade-off for caching with uncoded prefetching,” *in IEEE Trans. Inf. Theory*, vol. 64, no. 2, pp. 1281–1296, 2018.

[7] U. Niesen and M. A. Maddah-Ali, “Coded caching with nonuniform demands,” *in IEEE Trans. Inf. Theory*, vol. 63, no. 2, pp. 1146–1158, 2017.

[8] S. P. Shariatpanahi, S. A. Motahari, and B. H. Khalaj, “Multi-server Coded Caching,” *in IEEE Trans. on Inf. Theory*, vol. 62, no. 12, pp. 7253–7271, 2016.

[9] J. Hachem, N. Karamchandani, and S. Diggavi, “Multi-level coded caching,” *in Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, pp. 56–60, Jun./Jul. 2016.

[10] H. Ghasemi and A. Ramamoorthy, “Improved lower bounds for coded caching,” *in IEEE Trans. Inf. Theory*, vol. 63, no. 7, pp. 4388–4413, Jul. 2017.

[11] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, “Private information retrieval,” *in Proc. the 36th Annual Symposium on Foundations of Computer Science*, Wisconsin, USA, pp. 41-50, Oct. 1995.

[12] A. Beimel, Y. Ishai, and E. Kushilevitz, “General constructions for information-theoretic private information retrieval,” *Journal of Computer and Systems Sciences*, vol. 71, no. 2, pp. 213-247, Aug. 2005.

[13] S. Yekhanin, “Locally decodable codes and private information retrieval schemes,” *Ph.D. dissertation*, Massachusetts Institute of Technology, 2007.

[14] H. Sun and S. A. Jafar, “The capacity of private information retrieval,” *arXiv preprint arXiv:1602.09134*, 2016.

[15] R. Tandon, ”The capacity of cache-aided private information retrieval,” *arXiv preprint arXiv:1602.091341706.07035*, 2017.

[16] Y. Wei, K. Banawan, and S. Ulukus, ”Fundamental limits of cache-aided private information retrieval with unknown and uncoded prefetching,” *arXiv preprint arXiv:1709.01056*, 2017.

[17] S. Kadhe, B. Garcia, A. Heidarzadeh, S. El Rouayheb, and A. Sprintson, ”Private information retrieval with side information,” *arXiv preprint arXiv:1709.00112*, 2017.

[18] Z. Chen, Z. Wang, and S. Jafar, ”The Capacity of Private Information Retrieval with Private Side Information,” *arXiv preprint arXiv:1709.03022*, 2017.

[19] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, “Speeding up distributed machine learning using codes,” in *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1514-1529, Mar. 2018.

[20] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, “A fundamental tradeoff between computation and communication in distributed computing,” in *IEEE Transactions on Information Theory*, vol. 64, no. 1, pp. 109-128, Jan. 2018.

[21] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *International Conference on Machine Learn (ICML)*, 2017, pp. 3368-3376.

[22] S. Dutta, V. Cadambe, and P. Grover, "Coded convolution for parallel and distributed computing within a deadline," in *Proc. International Symposium on Information Theory (ISIT)*, June 2017, pp. 2403-2407.

[23] S. Dutta, V. Cadambe, and P. Grover, "Short-dot: computing large linear transforms distributedly using coded short dot product," in *Proc. 31th Annual Conference on Neural Information Processing Systems (NIPS)*, Dec. 2017.

[24] R. Bitar, P. Parag, and S. E. Rouayheb, "Minimizing latency for secure distributed computing," in *Proc. International Symposium on Information Theory (ISIT)*, Jun. 2017.

[25] W. Halbawi, N. Azizan-Ruhi, F. Salehi, and B. Hassibi, "Improving distributed gradient descent using Reed-Solomon codes," in *arXiv preprint arXiv:1706.05436*, 2017.

[26] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "A unified coding framework for distributed computing with straggling servers," in *Proc. Global Communications Conference (GLOBECOM) Workshops*, Dec. 2017.

[27] K. Lee, C. Suh, and K. Ramchandran, "High-dimensional coded matrix multiplication," in *Proc. International Symposium on Information Theory (ISIT)*, Jun. 2017.

[28] C. Karakus, Y. Sun, S. Diggavi, and W. Yin, "Straggler Mitigation in Distributed Optimization through Data Encoding," in *Advances in Neural Information Processing Systems (NIPS)*, pp. 5440–5448. 2017.

[29] C. Karakus, Y. Sun, and S. Diggavi, "Encoded distributed optimization," in *IEEE International Symposium on Information Theory (ISIT)*, pp. 2890–2894. 2017.

[30] Z. Charles, D. Papailiopoulos, and J. Ellenberg, "Approximate gradient coding via sparse random graphs," in *arXiv preprint arXiv:1711.06771*, 2017.

[31] R. K. Maity, A. S. Rawat, and A. Mazumdar, "Robust gradient descent via moment encoding with ldpc codes," in *arXiv preprint arXiv:1805.08327*, 2018.

[32] Y. Yang, P. Grover, and S. Kar, "Coded Distributed Computing for Inverse Problems," in *Advances in Neural Information Processing Systems (NIPS)*, pp. 709–719. 2017.

[33] S. Li, S. M. M. Kalan, Q. Yu, M. Soltanolkotabi, and A. S. Avestimehr, "Polynomially coded regression: Optimal straggler mitigation via data encoding," in *arXiv preprint arXiv:1805.09934*, 2018.

[34] Y. Yang, P. Grover, and S. Kar, "Coding for a single sparse inverse problem," in *IEEE International Symposium on Information Theory (ISIT)*, pp. 1575–1579. 2018.

[35] Y. Yang, M. Chaudhari, P. Grover, and S. Kar, "Coded iterative computing using substitute decoding," in *arXiv preprint arXiv:1805.06046*, 2018.

[36] F. Haddadpour, Y. Yang, M. Chaudhari, V. R. Cadambe, and P. Grover, "Straggler-resilient and communication-efficient distributed iterative linear solver," in *arXiv preprint arXiv:1806.06140*, 2018.

[37] F. Haddadpour and V. R. Cadambe, "Codes for distributed finite alphabet matrix-vector multiplication," in *Proc. International Symposium on Information Theory (ISIT)*, pp. 1625– 1629. 2018.

[38] H. Park, K. Lee, J.-y. Sohn, C. Suh, and J. Moon, "Hierarchical coding for distributed computing," in *arXiv preprint arXiv:1801.04686*, 2018.

[39] Q. Yu, M. Maddah-Ali, and S. Avestimehr, "Polynomial codes: An optimal design for high-dimensional coded matrix multiplication," in *Proc. 31th Annual Conference on Neural Information Processing Systems (NIPS)*, Dec. 2017, pp. 4406-4416.

[40] S. Kiani, N. Ferdinand and S. C. Draper, "Exploitation of Stragglers in Coded Computation," in *arXiv preprint arXiv:1806.10253*, 2018.

[41] H. Yang and J. Lee, “Secure Distributed Computing With Straggling Servers Using Polynomial Codes,” in *IEEE Transactions on Information Forensics and Security* , vol. 14, no. 1, pp. 141-150, 2019.

[42] W. Chang and R. Tandon, “On the Capacity of Secure Distributed Matrix Multiplication,” in *arXiv preprint arXiv:1806.00469* , 2018.

[43] J. Kakar, S. Ebadifar, and A. Sezgin, “Rate-Efficiency and Straggler-Robustness through Partition in Distributed Two-Sided Secure Matrix Computation,” in *arXiv preprint arXiv:1810.13006* , 2018.

[44] Q. Yu, N. Raviv, J. So, and S. Avestimehr, “Lagrange Coded Computing: Optimal Design for Resiliency, Security and Privacy,” in *arXiv preprint arXiv:1806.00939* , 2018.

[45] R. G. D′Oliveira, S. E. Rouayheb, and D. Karpuk, “GASP codes for secure distributed matrix multiplication,” in *arXiv preprint arXiv:1812.09962* , 2018.

[46] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, “Private information retrieval,” in *Journal of the ACM*, 45(6):965-981, 1998.

[47] H. Sun and S. A. Jafar, “The Capacity of Private Information Retrieval,” in *IEEE Transactions on Information Theory*, vol. 63, no. 7, pp. 4075-4088, Jul. 2017.

[48] H. Sun and S. A. Jafar, “The Capacity of Private Information Retrieval with Colluding Databases,” in *Proc. IEEE Global Conf. Signal Inf. Process*, pp. 941-946, Dec. 2016.

[49] K. Banawan and S. Ulukus, “The Capacity of Private Information Retrieval From Coded Databases,” in *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1945-1956, 2018.

[50] K. Banawan and S. Ulukus, “The Capacity of Private Information Retrieval from Byzantine and Colluding Databases,” in *arXiv preprint arXiv:1706.01142*, 2017.

[51] M. Kim, H. Yang and J. Lee, “Cache-aided Private Information Retrieval,” in *IEEE Asilomar*, Oct. 2017.

[52] Y.-P. Wei, K. Banawan, and S. Ulukus, “Cache-aided private information retrieval with partially known uncoded prefetching: Fundamental limits,” in *IEEE Jour. on Selected Areas in Communications*, vol. 36, no. 6, pp. 1126-1139, Jun. 2018.

[53] H. Sun and S. A. Jafar, ”The Capacity of Robust Private Information Retrieval with Colluding Databases,” in *arXiv preprint arXiv:1606.08828*, 2016.

[54] Z. Jin, H. Sun and S. A. Jafar, ”Cross Subspace Alignment and the Asymptotic Capacity of X-secure T-private Information Retrieval,” in *arXiv preprint arXiv:1808.07457*, 2018.

[55] M. Kim, H. Yang and J. Lee, “Private Coded Matrix Multiplication,” *IEEE Transactions on Information Forensics and Security*, Early Access, 2019.

[56] M. Mirmohseni and M. A. Maddah-Ali, ”Private Function Retrieval,” in *arXiv preprint arXiv:1711.04677*, 2017.

[57] S. A. Obead and J. Kliewer, ”Achievable Rate of Private Function Retrieval from MDS Coded Databases,” in *IEEE International Symposium on Information Theory (ISIT)*, pp. 2117-2121, 2018.

[58] S. A. Obead, H.-Y. Lin, E. Rosnes, J. Kliewer, ”Capacity of Private Linear Computation for Coded Databases,” in *arXiv preprint arXiv:1810.04230*, 2018.

[59] H. Sun and S. A. Jafar, ”The capacity of private computation,” in *arXiv preprint arXiv:1710.11098*, 2017.

[60] N. Raviv and D. A. Karpuk, ”Private Polynomial Computation from Lagrange Encoding,” in *arXiv preprint arXiv:1812.04142*, 2018.

[61] E. Ozafatura, D. Gunduz and S. Ulukus, ”Speeding Up Distributed Gradient Descent by Utilizing Non-persistent Stragglers,” in *arXiv preprint arXiv:1808.02240*, 2018.

[62] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, “Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding,” in *arXiv preprint arXiv:1801.07487.*, 2018.

초 록

많은 양의 데이터 저장이나 데이터 계산을 위해서는 분산 시스템이 필수적이다. 이러한 분산 시스템의 데이터 저장과 계산의 효율의 높이는 반면, 데이터의 보안과 프라이버시에 대한 위험도 증가시킨다. 본 논문에서는 데이터 저장과 데이터 계산을 위한 분산 시스템에서 데이터에 대한 보안과 프라이버시를 고려한다. 특히, 이러한 시스템에 대하여 보안과 프라이버시를 보장하는 부호화 기법을 제안한다.

우선, 유저가 사전에 캐시에 일정량의 데이터를 저장하고 있는 cache-aided PIR을 제안한다. 제안하는 기법은 기존 PIR 문제의 최적 기법을 기반으로 한다. 제안하는 기법에서, 캐시에 저장된 데이터는 부가정보로 이용되며, 이는 캐시가 없을 때 대비 다운로드양의 감소로 이어진다.

두 번째로, 부호화된 분산 컴퓨팅 시스템에서 마스터의 프라이버시를 고려한다. 이 시스템에서 워커들과 마스터는 각각 고유한 데이터를 가지며, 워커들의 데이터는 라이브러리 형태로 이루어진다. 마스터는 자신의 데이터와 데이터 라이브러리 내 특정 데이터의 함수를 계산해야 한다. 이 때 마스터의 프라이버시는 워커들이 마스터가 라이브러리 안의 어떤 데이터를 원하는지 모르는 것을 뜻한다.

이러한 시스템을 private coded computation이라 하며, 제안하는 기법을 private polynomial codes라 한다. 제안하는 기법에서는 기존의 polynomial codes에서는 고려되지 않았던 비동기적 기법이 도입된다. 이로 인하여 제안하는 기법은 변형된 최적의 RPIR 기법대비 더 빠른 계산시간을 달성한다.

끝으로, 부호화된 분산 컴퓨팅 시스템에서 마스터의 프라이버시와 데이터 보안을 동시에 고려한다. 데이터 보안은 마스터의 고유한 데이터를 워커들로부터 보호하는 것을 의미한다. 이러한 시스템을 private secure coded computation이라 하며, 제안하는 기법을 private secure polynomial codes라 한다. Private polynomial codes를 변형하여 private secure polynomial codes와 private polynomial codes를 계산시간과 통신량 측면에서 비교한다. 그 결과, 같은 양의 통신량에 대하여, private secure polynomial codes가 더 빠른 계산 시간을 달성한다.

주요어: 분산 컴퓨팅, Private Information Retrieval, 정보이론, 보안

학번: 2014-21647

ACKNOWLEDGEMENT

Part of this work was presented at 2017 51st Asilomar Conference on Signals, Systems, and Computers, 2019 IEEE International Symposium on Information Theory, published at IEEE Transactions on Information Forensics and Security, IEEE Communications Letters, and is covered by the following copyrights : ©2017 IEEE. Reprinted, with permission, from Minchul Kim, "Cache-Aided Private Information Retrieval", 51st Asilomar Conference on Signals, Systems, and Computers, 2017, ©2019 IEEE. Reprinted, with permission, from Minchul Kim, "Private Secure Coded Computation", IEEE International Symposium on Information Theory, 2019, ©2019 IEEE. Reprinted, with permission, from Minchul Kim, "Private Secure Coded Computation", IEEE Communications Letters, ©2019 IEEE. Reprinted, with permission, from Minchul Kim, "Private Coded Matrix Multiplication", IEEE Transactions on Information Forensics and Security.