Fully Private Grouped Matrix Multiplication with Colluding Workers

Lev Tauz and Lara Dolecek

Electrical and Computer Engineering, University of California, Los Angeles, USA

15th Annual Non-Volatile Memories Workshop 2024
Outline

1 Background and Motivation

2 Our Scheme

3 Privacy from the Master
Distributed systems are a mainstay of modern big data applications due to high parallelization gains.
Distributed systems are a mainstay of modern big data applications due to high parallelization gains ... in theory.
Distributed systems are a mainstay of modern big data applications due to high parallelization gains ... in theory

With new pay-to-compute services, outsourcing work leads to new issues:
Modern Issues of Distributed Computing

➤ Distributed systems are a mainstay of modern big data applications due to high parallelization gains ... in theory

➤ With new pay-to-compute services, outsourcing work leads to new issues:
  ◆ Straggling Workers
Modern Issues of Distributed Computing

- Distributed systems are a mainstay of modern big data applications due to high parallelization gains ...

- With new pay-to-compute services, outsourcing work leads to new issues:
  - Straggling Workers
  - Privacy Concerns
Coded Computation to the Rescue

- Coded computation addresses these issues by encoding data in a smart way.
Coded Computation to the Rescue

- Coded computation addresses these issues by encoding data in a smart way.

- Shown great success in **Distributed Large Matrix Multiplication**

![Diagram showing the process of coded computation with Encode, Decode, and Result: AB]
Coded Computation to the Rescue

- Coded computation addresses these issues by encoding data in a smart way.
- Shown great success in **Distributed Large Matrix Multiplication**
  - Fundamental building block of modern machine learning algorithms.
Coded Computation to the Rescue

- Coded computation addresses these issues by encoding data in a smart way
- Shown great success in **Distributed Large Matrix Multiplication**
  - Fundamental building block of modern machine learning algorithms
- In this work, we focus on private Distributed Large Matrix Multiplication as the primary use-case
Coded Computation to the Rescue

- Coded computation addresses these issues by encoding data in a smart way.
- Shown great success in **Distributed Large Matrix Multiplication**
  - Fundamental building block of modern machine learning algorithms.
- In this work, we focus on private Distributed Large Matrix Multiplication as the primary use-case.
- We start by discussing previously considered privacy models.

![Diagram showing encoded data and results](image-url)
Short Summary of Previous Privacy Models

1. Data Privacy

Example:
Secure Matrix Multiplication
(Yu ’20)
Short Summary of Previous Privacy Models

1. Data Privacy
2. Request Privacy

Example:

*Fully Private Matrix Multiplication*

(Kim ’19)
Short Summary of Previous Privacy Models

1. Data Privacy
2. Request Privacy
3. Data + Request Privacy

New Privacy Model: Batch Size Privacy

Monitor Data: $A$, Index: $j \in [k]$

Goal:

Worker Data: $\{B_1, B_2, \ldots, B_k\}$

Example:
Secure and Private Matrix Multiplication
(Zhu '22)
Short Summary of Previous Privacy Models

1. Data Privacy
2. Request Privacy
3. Data + Request Privacy

New Privacy Model: Batch Size Privacy

Example: Secure and Private Matrix Multiplication (Zhu ’22)
New Privacy Model: Batch Size Privacy (BSP)

➤ The master wants to hide the number of requests/batches it wishes to compute

Example:

◆ Consider a system where workers store two libraries \{A_1, A_2\} and \{B_1\}

◆ The user can request any group of matrix products e.g. \{A_1 B_1\}, \{A_2 B_1\}, and \{A_1 B_1, A_2 B_1\}

◆ If workers knew the batch size is 1, they reduce the space of possibilities

◆ If workers knew the batch size is 2, they know the exact request

Goal:

\[ W_1 W_2 W_N \]

Worker Data

Master

Private

➤ Can even be problematic to systems without batch size limits

 Tauz, Dolecek (UCLA)  NVMW 2024  6 / 32
New Privacy Model: Batch Size Privacy (BSP)

- The master wants to hide the **number of requests/batches** it wishes to compute
- Example:

  - Consider a system where workers store two libraries \( \{A_1, A_2\} \) and \( \{B_1\} \)
  - The user can request any group of matrix products e.g. \( \{A_1B_1\}, \{A_2B_1\}, \{A_1B_1, A_2B_1\} \)
  - If workers knew the batch size is 1, they reduce the space of possibilities
  - If workers knew the batch size is 2, they know the exact request

![Diagram](https://example.com/diagram.png)
New Privacy Model: Batch Size Privacy (BSP)

- The master wants to hide the number of requests/batches it wishes to compute.

Example:
- Consider a system where workers store two libraries \( \{A_1, A_2\} \) and \( \{B_1\} \)

Goal:
- \( W_1, W_2, W_N \) (Worker Data)
- Private
- \( \text{Index: } S \subseteq [l] \times [k] \)

Master

\{A_i B_j : (i, j) \in S\}

Worker Data
- \( \{A_1, A_2, \ldots, A_l\} \)
- \( \{B_1, B_2, \ldots, B_k\} \)
New Privacy Model: Batch Size Privacy (BSP)

➢ The master wants to hide the number of requests/batches it wishes to compute

➢ Example:
  - Consider a system where workers store two libraries \( \{A_1, A_2\} \) and \( \{B_1\} \)
  - The user can request any group of matrix products e.g. \( \{A_1B_1\}, \{A_2B_1\}\), and \( \{A_1B_1, A_2B_1\}\)
New Privacy Model: Batch Size Privacy (BSP)

➤ The master wants to hide the number of requests/batches it wishes to compute

➤ Example:
  - Consider a system where workers store two libraries \( \{A_1, A_2\} \) and \( \{B_1\} \)
  - The user can request any group of matrix products e.g. \( \{A_1B_1\}, \{A_2B_1\} \), and \( \{A_1B_1, A_2B_1\} \)
  - If workers knew the batch size is 1, they reduce the space of possibilities
New Privacy Model: Batch Size Privacy (BSP)

➤ The master wants to hide the number of requests/batches it wishes to compute

➤ Example:
  ◆ Consider a system where workers store two libraries \( \{A_1, A_2\} \) and \( \{B_1\} \)
  ◆ The user can request any group of matrix products e.g. \( \{A_1B_1\}, \{A_2B_1\} \), and \( \{A_1B_1, A_2B_1\} \)
  ◆ If workers knew the batch size is 1, they reduce the space of possibilities
  ◆ If workers knew the batch size is 2, they know the exact request
New Privacy Model: Batch Size Privacy (BSP)

- The master wants to hide the number of requests/batches it wishes to compute

Example:
- Consider a system where workers store two libraries \(\{A_1, A_2\}\) and \(\{B_1\}\)
- The user can request any group of matrix products e.g. \(\{A_1B_1\}, \{A_2B_1\}\), and \(\{A_1B_1, A_2B_1\}\)
- If workers knew the batch size is 1, they reduce the space of possibilities
- If workers knew the batch size is 2, they know the exact request

- Can even be problematic to systems without batch size limits
Motivating Examples

- Inferring Size of Neural Networks
  - Using batched coded computing, a user can train a network using model parallelism
Practical Applications for Batch Size Privacy

Motivating Examples

➤ Inferring Size of Neural Networks

- Using batched coded computing, a user can train a network using model parallelism
- Batch size can help infer the size of the network
Motivating Examples

➤ Identifying Entities in the System

- Batch size can help identify the identities of users or the use case of the request
- Small IoT devices would generally request small batches
- Large analytic engines would request large batch sizes
Motivating Examples

- **Identifying Entities in the System**
  - Batch size can help identify the identities of users or the use case of the request
Motivating Examples

➤ Identifying Entities in the System

- Batch size can help identify the identities of users or the use case of the request
- Small IOT devices would generally request small batches
Practical Applications for Batch Size Privacy

Motivating Examples

Identifying Entities in the System

- Batch size can help identify the identities of users or the use case of the request
- Small IOT devices would generally request small batches
- Large analytic engines would request large batch sizes
In this work, we consider a demonstrative problem setting that incorporates both request and batch size privacy.
In this work, we consider a demonstrative problem setting that incorporates both request and batch size privacy.

For simplicity, we consider a scenario where workers store all the data and, thus, no data privacy is necessary.
In this work, we consider a demonstrative problem setting that incorporates both request and batch size privacy.

For simplicity, we consider a scenario where workers store all the data and, thus, no data privacy is necessary.

The user is allowed to request any number of matrix products among the data stored at the workers.
In this work, we consider a demonstrative problem setting that incorporates both request and batch size privacy.

For simplicity, we consider a scenario where workers store all the data and, thus, no data privacy is necessary.

The user is allowed to request any number of matrix products among the data stored at the workers.

We name this new system Fully Private Grouped Matrix Multiplication (FPGMM).
Fully Private Grouped Matrix Multiplication (FPGMM)

➤ Assume workers store two libraries $A_{[L_A]} = \{A_i \in \mathbb{F}^{\alpha \times \alpha}, \forall i \in [L_A]\}$ and $B_{[L_B]} = \{B_i \in \mathbb{F}^{\alpha \times \alpha}, \forall i \in [L_B]\}$

Fully Private Grouped Matrix Multiplication (FPGMM)

➤ Assume workers store two libraries $A_{[L_A]} = \{A_i \in \mathbb{F}_{\alpha \times \alpha}, \forall i \in [L_A]\}$ and $B_{[L_B]} = \{B_i \in \mathbb{F}_{\alpha \times \alpha}, \forall i \in [L_B]\}$

➤ A master selects uniformly at random a non-empty set $S \subseteq \{(i, j) : i \in [L_A], j \in [L_B]\}$
Fully Private Grouped Matrix Multiplication (FPGMM)

- Assume workers store two libraries $A_{[L_A]} = \{ A_i \in \mathbb{F}^{\alpha \times \alpha}, \forall i \in [L_A] \}$ and $B_{[L_B]} = \{ B_i \in \mathbb{F}^{\alpha \times \alpha}, \forall i \in [L_B] \}$

- A master selects uniformly at random a non-empty set $S \subseteq \{(i, j) : i \in [L_A], j \in [L_B]\}$

- **Goal:** Calculate the matrix products $C_S \triangleq \{ A_i B_j : (i, j) \in S \}$ with the following requirements:
  - Small recovery threshold $R$,
  - Measure of straggler resilience
  - Flexible computation and communication overhead
  - Privacy against any $T$ colluding workers
    - Any group of $T$ workers cannot learn anything about $S$ given their received queries, even the cardinality of $S$
Fully Private Grouped Matrix Multiplication (FPGMM)

- Assume workers store two libraries $A_{[L_A]} = \{A_i \in \mathbb{F}^{\alpha \times \alpha}, \forall i \in [L_A]\}$ and $B_{[L_B]} = \{B_i \in \mathbb{F}^{\alpha \times \alpha}, \forall i \in [L_B]\}$
- A master selects uniformly at random a non-empty set $S \subseteq \{(i, j) : i \in [L_A], j \in [L_B]\}$
- **Goal:** Calculate the matrix products $C_S \triangleq \{A_i B_j : (i, j) \in S\}$ with the following requirements:
  - **Small recovery threshold** $R$, i.e. the minimum number of worker results needed in order for the master to decode
Fully Private Grouped Matrix Multiplication (FPGMM)

➤ Assume workers store two libraries \( A_{[L_A]} = \{ A_i \in \mathbb{F}^{\alpha \times \alpha}, \forall i \in [L_A] \} \)
and \( B_{[L_B]} = \{ B_i \in \mathbb{F}^{\alpha \times \alpha}, \forall i \in [L_B] \} \)

➤ A master selects uniformly at random a non-empty set \( S \subseteq \{(i, j) : i \in [L_A], j \in [L_B]\} \)

➤ **Goal:** Calculate the matrix products \( C_S \triangleq \{ A_i B_j : (i, j) \in S \} \)
with the following requirements:

- **Small recovery threshold** \( R \), i.e. the minimum number of worker results needed in order for the master to decode
  - Measure of straggler resilience
Fully Private Grouped Matrix Multiplication (FPGMM)

➤ Assume workers store two libraries \( A_{[L_A]} = \{ A_i \in \mathbb{F}^{\alpha \times \alpha}, \forall i \in [L_A] \} \) and \( B_{[L_B]} = \{ B_i \in \mathbb{F}^{\alpha \times \alpha}, \forall i \in [L_B] \} \)

➤ A master selects uniformly at random a non-empty set \( S \subseteq \{(i, j) : i \in [L_A], j \in [L_B]\} \)

➤ **Goal:** Calculate the matrix products \( C_S \triangleq \{ A_i B_j : (i, j) \in S \} \) with the following requirements:

- **Small recovery threshold** \( R \), i.e. the minimum number of worker results needed in order for the master to decode
  - Measure of straggler resilience

- **Flexible computation and communication overhead**
Fully Private Grouped Matrix Multiplication (FPGMM)

➤ Assume workers store two libraries \( A_{[L_A]} = \{ A_i \in F^{\alpha \times \alpha}, \forall i \in [L_A] \} \) and \( B_{[L_B]} = \{ B_i \in F^{\alpha \times \alpha}, \forall i \in [L_B] \} \)

➤ A master selects uniformly at random a non-empty set \( S \subseteq \{(i, j) : i \in [L_A], j \in [L_B] \} \)

➤ **Goal:** Calculate the matrix products \( C_S \triangleq \{ A_i B_j : (i, j) \in S \} \) with the following requirements:

- **Small recovery threshold** \( R \), i.e. the minimum number of worker results needed in order for the master to decode
  - Measure of straggler resilience

- **Flexible computation and communication overhead**

- **Privacy against any** \( T \) **colluding workers**
Fully Private Grouped Matrix Multiplication (FPGMM)

➤ Assume workers store two libraries \( A_{[L_A]} = \{ A_i \in \mathbb{F}^{\alpha \times \alpha}, \forall i \in [L_A] \} \) and \( B_{[L_B]} = \{ B_i \in \mathbb{F}^{\alpha \times \alpha}, \forall i \in [L_B] \} \)

➤ A master selects uniformly at random a non-empty set \( S \subseteq \{(i, j) : i \in [L_A], j \in [L_B] \} \)

➤ **Goal:** Calculate the matrix products \( C_S \triangleq \{ A_i B_j : (i, j) \in S \} \) with the following requirements:

- **Small recovery threshold** \( R \), i.e. the minimum number of worker results needed in order for the master to decode
  - Measure of straggler resilience

- **Flexible computation and communication overhead**

- **Privacy against any** \( T \) **colluding workers**
  - Any group of \( T \) workers *cannot learn anything about* \( S \) given their received queries, even the cardinality of \( S \)
Model Overview

**Protocol** :
- **Query**: Master sends queries $q_{[N]}$ to each worker
- **Download**: Workers output $U_i = f(A_{[L_A]}, B_{[L_B]}, q_i)$
- **Decode**: $C_S$ decoded from first $R$ worker outputs

**Privacy Requirement**:
\[ I(S; q_T, A_{[L_A]}, B_{[L_B]}) = 0, \forall T \in [N], |T| \leq T \]
Novelty of Problem

➢ In most coded computation settings, the workers are allowed to know something about the encoding process
Novelty of Problem

➤ In most coded computation settings, the workers are allowed to know something about the encoding process
  ◆ For example, in polynomial coded based schemes, it is ok for workers to know the degree of the polynomial
Novelty of Problem

➤ In most coded computation settings, the workers are allowed to know something about the encoding process
  ◆ For example, in polynomial coded based schemes, it is ok for workers to know the degree of the polynomial
➤ Since batch size impacts the encoding process, we need to further limit the knowledge of the workers to ensure BSP
Novelty of Problem

➤ In most coded computation settings, the workers are allowed to know something about the encoding process
   ♦ For example, in polynomial coded based schemes, it is ok for workers to know the degree of the polynomial
➤ Since batch size impacts the encoding process, we need to further limit the knowledge of the workers to ensure BSP
➤ A related topic is function privacy which focused on:
Novelty of Problem

➤ In most coded computation settings, the workers are allowed to know something about the encoding process
  ◆ For example, in polynomial coded based schemes, it is ok for workers to know the degree of the polynomial

➤ Since batch size impacts the encoding process, we need to further limit the knowledge of the workers to ensure BSP

➤ A related topic is function privacy which focused on:
  ◆ Complex Polynomial Evaluations (Raviv ’19)
Novelty of Problem

- In most coded computation settings, the workers are allowed to know something about the encoding process
  - For example, in polynomial coded based schemes, it is ok for workers to know the degree of the polynomial
- Since batch size impacts the encoding process, we need to further limit the knowledge of the workers to ensure BSP
- A related topic is function privacy which focused on:
  - Complex Polynomial Evaluations (Raviv ’19)
  - Simple Linear Combinations (Sun ’18)
Novelty of Problem

➤ In most coded computation settings, the workers are allowed to know something about the encoding process
  ◆ For example, in polynomial coded based schemes, it is ok for workers to know the degree of the polynomial

➤ Since batch size impacts the encoding process, we need to further limit the knowledge of the workers to ensure BSP

➤ A related topic is function privacy which focused on:
  ◆ Complex Polynomial Evaluations (Raviv ’19)
  ◆ Simple Linear Combinations (Sun ’18)

➤ Our work falls between these two regimes since we focus on the bi-linear operation of matrix multiplication and batch processing
Novelty of Problem

➤ In most coded computation settings, the workers are allowed to know something about the encoding process
  ◆ For example, in polynomial coded based schemes, it is ok for workers to know the degree of the polynomial
➤ Since batch size impacts the encoding process, we need to further limit the knowledge of the workers to ensure BSP
➤ A related topic is function privacy which focused on:
  ◆ Complex Polynomial Evaluations (Raviv ’19)
  ◆ Simple Linear Combinations (Sun ’18)
➤ Our work falls between these two regimes since we focus on the bi-linear operation of matrix multiplication and batch processing
➤ Additionally, in FPGMM, data can be re-used across multiple request, unlike other works that focus on distinct data
Novelty of Problem

By further limiting the knowledge of the workers, many state-of-the-art coded computation schemes are *severely limited* in:
By further limiting the knowledge of the workers, many state-of-the-art coded computation schemes are *severely limited* in:

- Optimizing the communication and computation costs
Novelty of Problem

By further limiting the knowledge of the workers, many state-of-the-art coded computation schemes are severely limited in:

- Optimizing the communication and computation costs (this paper)
Novelty of Problem

By further limiting the knowledge of the workers, many state-of-the-art coded computation schemes are severely limited in:

- Optimizing the communication and computation costs (this paper)
- Providing other forms of privacy such as privacy from the master
Novelty of Problem

➤ By further limiting the knowledge of the workers, many state-of-the-art coded computation schemes are severely limited in:
  ◆ Optimizing the communication and computation costs (this paper)
  ◆ Providing other forms of privacy such as privacy from the master (ongoing work)
1 Background and Motivation

2 Our Scheme

3 Privacy from the Master
Useful Tool: Interpolation of Rational Functions

Rational Polynomial with Fixed Poles

\[ F(z) = \sum_{i=1}^{N} \sum_{j=1}^{u_i} \frac{e_{i,j}}{(z - f_i)^j} + \sum_{j=0}^{K-M-1} e_{0,j} z^j \]

\( f_1, f_2, \ldots, f_N \) are fixed

- Can interpolate \( F(z) \) if the number of interpolation points is equal to the number of rational and polynomial terms (Gasca ’89)
Useful Tool: Interpolation of Rational Functions

Rational Polynomial with Fixed Poles

\[ F(z) = \sum_{i=1}^{N} \sum_{j=1}^{u_i} \frac{e_{i,j}}{(z - f_i)^j} + \sum_{j=0}^{K-M-1} e_{0,j} z^j \]

\( f_1, f_2, \ldots, f_N \) are fixed

- Can interpolate \( F(z) \) if the number of interpolation points is equal to the number of rational and polynomial terms (Gasca ’89)

- Additionally, there are fast methods of interpolation that are comparable to polynomial interpolation (Olshevsky ’01)
Cross Subspace Alignment (CSA) codes are a coded computation scheme for calculating batches of matrix products.

Desired Terms

Garbage Terms

Subspace 1

Subspace 2

\[ \frac{1}{(1 - f_1) a_k} \]

\[ \frac{1}{(1 - f_2) a_k} \]

\{1, x, x^2, \ldots\}

Recovery Threshold = # of Terms

Goal: Minimizing the # of Garbage Terms
Cross Subspace Alignment (CSA) codes are a coded computation scheme for calculating batches of matrix products.

Utilizes *rational functions* in order to encode the data and extract the desired terms.

**Goal:** Minimizing the number of garbage terms.
Another Tool: Cross-Subspace Alignment Codes

- Cross Subspace Alignment (CSA) codes are a coded computation scheme for calculating batches of matrix products.
- Utilizes rational functions in order to encode the data and extract the desired terms.
- Allows for flexibility in communication and computation costs due to its unique grouping capability.

Graph showing desired terms and garbage terms, with recovery threshold equal to the number of terms. The goal is to minimize the number of garbage terms.
Grouping Structure of CSA codes

Rational Bases
Desired Terms

Polynomial Bases
Garbage Terms

\[ \frac{1}{(1 - f_1)} \frac{1}{(1 - f_2)} \ldots \frac{1}{(1 - f_k)} \]

\[ S = \{ A_1B_1, A_2B_2, A_3B_3, A_4B_4 \} \]

\[ \{ x^0, x^1, x^2, \ldots \} \]
Grouping Structure of CSA codes

Rational Bases

Polynomial Bases

Group 1 : $A_1 B_1$

Group 2 : $A_2 B_2$

Group 3 : $A_3 B_3$

Group 4 : $A_4 B_4$

Grouping reduces the recovery threshold at the cost of more computation. The number of groups reveals information about the batch size which breaks privacy.
Grouping Structure of CSA codes

Rational Bases

 Polynomial Bases

 Group 1 : $A_1B_1$

 Group 2 : $A_2B_2$

 Group 3 : $A_3B_3$

 Group 4 : $A_4B_4$

➤ Grouping reduces the recovery threshold at the cost of more computation
Grouping Structure of CSA codes

Rational Bases

Polynomial Bases

Group 1 : $A_1B_1$
Group 2 : $A_2B_2$
Group 3 : $A_3B_3$
Group 4 : $A_4B_4$

➤ Grouping reduces the recovery threshold at the cost of more computation
➤ The number of groups reveals information about the batch size which breaks privacy
Key Idea for Our Scheme

Consider the following partitioning of the matrices given parameters $m$ and $n$:

\[
A_i = \begin{bmatrix}
A_{i,1} \\
\vdots \\
A_{i,m}
\end{bmatrix} \quad \forall i \in [L_A],
B_j = \begin{bmatrix}
B_{j,1} \\
\vdots \\
B_{j,n}
\end{bmatrix} \quad \forall j \in [L_B] \quad (1)
\]
Key Idea for Our Scheme

➤ Consider the following partitioning of the matrices given parameters $m$ and $n$:

$$A_i = \begin{bmatrix} A_{i,1} \\ \vdots \\ A_{i,m} \end{bmatrix}, \forall i \in [L_A], B_j = \begin{bmatrix} B_{j,1} & \cdots & B_{j,n} \end{bmatrix}, \forall j \in [L_B] \quad (1)$$

➤ If the master wants $A_i B_j$, then a sufficient condition is to get

$$A_i B_j = \{ A_{i,a} B_{j,b} \}_{a \in [m], b \in [n]}$$
Key Idea for Our Scheme

Consider the following partitioning of the matrices given parameters $m$ and $n$:

\[
A_i = \begin{bmatrix} A_{i,1} \\ \vdots \\ A_{i,m} \end{bmatrix}, \quad \forall i \in [L_A], \quad B_j = \begin{bmatrix} B_{j,1} & \cdots & B_{j,n} \end{bmatrix}, \quad \forall j \in [L_B] \tag{1}
\]

If the master wants $A_i B_j$, then a sufficient condition is to get

\[
A_i B_j = \{ A_{i,a} B_{j,b} : a \in [m], b \in [n] \}
\]

Hence, a sufficient condition to get $C_S$ is to calculate

\[
\{ A_{i,q} B_{j,s} : (i, j) \in S, (q, s) \in [m] \times [n] \}
\]
Key Idea for Our Scheme

Consider the following partitioning of the matrices given parameters $m$ and $n$:

$$A_i = \begin{bmatrix} A_{i,1} \\ \vdots \\ A_{i,m} \end{bmatrix}, \forall i \in [L_A], B_j = [B_{j,1} \cdots B_{j,n}], \forall j \in [L_B] \quad (1)$$

If the master wants $A_iB_j$, then a sufficient condition is to get $A_iB_j = \{A_{i,a}B_{j,b}\}_{a \in [m], b \in [n]}$

Hence, a sufficient condition to get $C_S$ is to calculate

$$\{A_{i,q}B_{j,s} : (i, j) \in S, (q, s) \in [m] \times [n]\}$$

This is another instance of FPGMM but the batch size is now a multiple of $mn$
Let us re-index the submatrices as follows:

\[
\tilde{A}_i = A_{\left\lfloor \frac{i-1}{m} \right\rfloor + 1, (i-1 \mod m) + 1}, i \in [mL_A]
\]

\[
\tilde{B}_j = B_{\left\lfloor \frac{j-1}{n} \right\rfloor + 1, (j-1 \mod n) + 1}, j \in [nL_B]
\]
Grouping Independently of the Batch Size

➤ Let us re-index the submatrices as follows:

\[ \tilde{A}_i = A_{\left[ \frac{i-1}{m} \right] + 1, (i-1 \mod m) + 1}, i \in [mL_A] \]

\[ \tilde{B}_j = B_{\left[ \frac{j-1}{n} \right] + 1, (j-1 \mod n) + 1}, j \in [nL_B] \]

➤ We can define a new FPGMM problem with computation list \( \tilde{S} \):

\[ \tilde{S} = \{ m(i - 1) + q, n(j - 1) + s : (i, j) \in S, (q, s) \in [m] \times [n] \} \]
Grouping Independently of the Batch Size

➤ Let us re-index the submatrices as follows:

\[
\tilde{A}_i = A\left[\frac{i-1}{m}\right]+1,(i-1 \mod m)+1, i \in [mL_A]
\]

\[
\tilde{B}_j = B\left[\frac{j-1}{n}\right]+1,(j-1 \mod n)+1, j \in [nL_B]
\]

➤ We can define a new FPGMM problem with computation list \(\tilde{S}\):

\[
\tilde{S} = \{m(i - 1) + q, n(j - 1) + s : (i, j) \in S, (q, s) \in [m] \times [n]\}
\]

➤ **Fact:** \(mn|\tilde{S}\)
Grouping Independently of the Batch Size

Let us re-index the submatrices as follows:

\[
\tilde{A}_i = A_{\left\lfloor \frac{i-1}{m} \right\rfloor +1, (i-1 \mod m)+1}, i \in [mL_A]
\]

\[
\tilde{B}_j = B_{\left\lfloor \frac{j-1}{n} \right\rfloor +1, (j-1 \mod n)+1}, j \in [nL_B]
\]

We can define a new FPGMM problem with computation list \(\tilde{S}\):

\[
\tilde{S} = \{m(i - 1) + q, n(j - 1) + s : (i, j) \in S, (q, s) \in [m] \times [n]\}
\]

Fact: \(mn|\tilde{S}\)

Key Idea: We can group according to the partitioning parameters and not the original batch size
Grouping Independently of the Batch Size

Let us re-index the submatrices as follows:

\[ \tilde{A}_i = A_{\left\lfloor \frac{i-1}{m} \right\rfloor + 1, (i-1 \mod m) + 1}, i \in [mL_A] \]

\[ \tilde{B}_j = B_{\left\lfloor \frac{j-1}{n} \right\rfloor + 1, (j-1 \mod n) + 1}, j \in [nL_B] \]

We can define a new FPGMM problem with computation list \( \tilde{S} \):

\[ \tilde{S} = \{ m(i - 1) + q, n(j - 1) + s : (i, j) \in S, (q, s) \in [m] \times [n] \} \]

Fact: \( mn | \tilde{S} \)

Key Idea: We can group according to the partitioning parameters and not the original batch size

Hence, we can still achieve flexibility in overhead costs without compromising privacy
Our Scheme

We breakdown our scheme into 3 main stages:

1. Encoding
2. Query and Computation
3. Decoding
The master determines the *partitioning parameters* $m$ and $n$ and the *grouping parameter* $r$ such that $r|mn$. 

1. The master groups $S$ into $r$ equal, non-overlapping groups of size $\delta = \frac{|S|}{mn}$ denoted by $Q_1, \ldots, Q_r$. For each $(i, j) \in S$, the master associates a distinct element $f_{i,j}$.
Encoding Phase: Create Grouping

1. The master determines the *partitioning parameters* $m$ and $n$ and the *grouping parameter* $r$ such that $r | mn$

2. The master groups $\tilde{S}$ into $r$ equal, non-overlapping groups of size $\delta = \frac{|S| mn}{r}$ denoted by $Q_1, \ldots, Q_r$
Encoding Phase: Create Grouping

1. The master determines the *partitioning parameters* $m$ and $n$ and the *grouping parameter* $r$ such that $r|m,n$

2. The master groups $\tilde{S}$ into $r$ equal, non-overlapping groups of size $\delta = \frac{|S|m}{r}$ denoted by $Q_1, \ldots, Q_r$

3. For each $(i, j) \in \tilde{S}$, the master associates a distinct element $f_{i,j}$
The master then creates the encoding functions for $k \in [r]$:

\[
\begin{align*}
a_{i,k}(x) &= \omega_k(x) \left( \sum_{(q,s) \in Q_k: q=i} \frac{1}{x - f_{q,s}} + z_{i,k}^a(x) \right), \\
b_{j,k}(x) &= \sum_{(q,s) \in Q_k: s=j} \frac{1}{x - f_{q,s}} + z_{j,k}^b(x), \\
\omega_k(x) &= \prod_{(i,j) \in Q_k} (x - f_{i,j})
\end{align*}
\]
The master then creates the encoding functions for $k \in [r]$: \[ a_{i,k}(x) = \omega_k(x) \left( \sum_{(q,s) \in Q_k: q = i} \frac{1}{x - f_{q,s}} + z^a_{i,k}(x) \right), \]
\[ b_{j,k}(x) = \sum_{(q,s) \in Q_k: s = j} \frac{1}{x - f_{q,s}} + z^b_{j,k}(x) \]
\[ \omega_k(x) = \prod_{(i,j) \in Q_k} (x - f_{i,j}) \]

- $z^a_{i,k}(x)$ and $z^b_{j,k}(x)$ are random polynomials of degree $T - 1$
The master then creates the encoding functions for \( k \in [r] \):

\[
a_{i,k}(x) = \omega_k(x) \left( \sum_{(q,s) \in \mathcal{Q}_k: q = i} \frac{1}{(x - f_{q,s})} + z_{i,k}^a(x) \right),
\]

\[
b_{j,k}(x) = \sum_{(q,s) \in \mathcal{Q}_k: s = j} \frac{1}{(x - f_{q,s})} + z_{j,k}^b(x)
\]

\[
\omega_k(x) = \prod_{(i,j) \in \mathcal{Q}_k} (x - f_{i,j})
\]

- \( z_{i,k}^a(x) \) and \( z_{j,k}^b(x) \) are random polynomials of degree \( T - 1 \)
- By Shamir’s secret sharing scheme, ensures that any \( T \) evaluations of \( a_{i,k}(x) \) and \( b_{j,k}(x) \) are uniformly random variables
Encoding Phase: Create Encoding Functions

4 The master then creates the encoding functions for $k \in [r]$:

\[
a_{i,k}(x) = \omega_k(x) \left( \sum_{(q,s) \in Q_k:q=i} \frac{1}{x - f_{q,s}} + z^a_{i,k}(x) \right),
\]

\[
b_{j,k}(x) = \sum_{(q,s) \in Q_k:s=j} \frac{1}{x - f_{q,s}} + z^b_{j,k}(x)
\]

\[
\omega_k(x) = \prod_{(i,j) \in Q_k} (x - f_{i,j})
\]

- $a_{i,k}(x)$ and $b_{j,k}(x)$ have the following property:

\[
a_{i,k}(x) b_{j,k}(x) = \beta_{i,j,k}(x) + \begin{cases} 
\frac{\gamma_{i,j,k}}{x - f_{i,j}} & (i,j) \in Q_k, \\
0 & \text{otherwise},
\end{cases}
\]

where $\gamma_{i,j,k}$ is a non-zero constant and $\beta_{i,j,k}(x)$ is a polynomial of degree $\delta + 2T - 2$.
For worker $g \in [N]$, the master associates an element $x_g$ that is distinct from $\{f_{i,j} : (i,j) \in \tilde{S}\}$.
1. For worker $g \in [N]$, the master associates an element $x_g$ that is distinct from $\{f_{i,j} : (i, j) \in \tilde{S}\}$

2. The master then constructs a query for each worker that contains the following:
For worker $g \in [N]$, the master associates an element $x_g$ that is distinct from $\{f_{i,j} : (i, j) \in \tilde{S}\}$.

The master then constructs a query for each worker that contains the following:
Query and Computation

1. For worker \( g \in [N] \), the master associates an element \( x_g \) that is distinct from \( \{f_{i,j} : (i, j) \in \tilde{S}\} \)

2. The master then constructs a query for each worker that contains the following:
   - The partitioning parameters \( m \) and \( n \)

For worker $g \in [N]$, the master associates an element $x_g$ that is distinct from \{\(f_{i,j} : (i, j) \in \tilde{S}\)\}

The master then constructs a query for each worker that contains the following:

- The partitioning parameters $m$ and $n$
- The grouping parameter $r$
For worker $g \in [N]$, the master associates an element $x_g$ that is distinct from $\{f_{i,j} : (i, j) \in \tilde{S}\}$.

The master then constructs a query for each worker that contains the following:

- The partitioning parameters $m$ and $n$
- The grouping parameter $r$
- The encoding coefficients $\{\{a_{i,k}(x_g)\}_{i=1}^{mL_A}\}_{k=1}^r$, $\{\{b_{j,k}(x_g)\}_{j=1}^{nL_B}\}_{k=1}^r$
Query and Computation

1. For worker $g \in [N]$, the master associates an element $x_g$ that is distinct from \{ $f_{i,j} : (i, j) \in \tilde{S}$ \}

2. The master then constructs a query for each worker that contains the following:

   - The partitioning parameters $m$ and $n$
   - The grouping parameter $r$
   - The encoding coefficients $\{ \{ a_{i,k}(x_g) \}_{i=1}^{mL_A} \}_{k=1}^{r}$, $\{ \{ b_{j,k}(x_g) \}_{j=1}^{nL_B} \}_{k=1}^{r}$

Any $T$ collection of these queries does not reveal anything about $\tilde{S}$
The worker then partitions the matrices and encodes them using the encoding parameters:

$$\tilde{A}_k = \sum_{i=1}^{mL_A} \tilde{A}_i a_{i,k}(x_g), \quad \tilde{B}_k = \sum_{j=1}^{nL_B} \tilde{B}_j b_{j,k}(x_g)$$
The worker then partitions the matrices and encodes them using the encoding parameters:

\[ \hat{A}_k = \sum_{i=1}^{mL_A} \tilde{A}_{i}a_{i,k}(x_g), \quad \hat{B}_k = \sum_{j=1}^{nL_B} \tilde{B}_{j}b_{j,k}(x_g) \]

The worker computes and outputs:

\[ \sum_{k=1}^{r} \hat{A}_k \hat{B}_k \]
Query and Computation

3 The worker then partitions the matrices and encodes them using the encoding parameters:

\[
\hat{A}_k = \sum_{i=1}^{mL_A} \tilde{A}_i a_{i,k}(x_g), \quad \hat{B}_k = \sum_{j=1}^{nL_B} \tilde{B}_j b_{j,k}(x_g)
\]

4 The worker computes and outputs:

\[
\sum_{k=1}^{r} \hat{A}_k \hat{B}_k = \sum_{k=1}^{r} \sum_{i=1}^{mL_A} \sum_{j=1}^{nL_B} \tilde{A}_i \tilde{B}_j a_{i,k}(x_g)b_{j,k}(x_g)
\]
The worker then partitions the matrices and encodes them using the encoding parameters:

\[
\hat{A}_k = \sum_{i=1}^{mL_A} \tilde{A}_{i}a_{i,k}(x_g), \quad \hat{B}_k = \sum_{j=1}^{nL_B} \tilde{B}_{j}b_{j,k}(x_g)
\]

The worker computes and outputs:

\[
\sum_{k=1}^{r} \hat{A}_k \hat{B}_k = \sum_{k=1}^{r} \sum_{i=1}^{mL_A} \sum_{j=1}^{nL_B} \tilde{A}_{i} \tilde{B}_{j} a_{i,k}(x_g) b_{j,k}(x_g)
\]

\[
= \sum_{k=1}^{r} \left( \sum_{(i,j) \in Q_k} \frac{\gamma^{i,j,k} \tilde{A}_i \tilde{B}_j}{(x_g - f_{i,j})} + \sum_{i=1}^{mL_A} \sum_{j=1}^{nL_B} \tilde{A}_{i} \tilde{B}_{j} \beta_{i,j,k}(x_g) \right)
\]

\[
= \sum_{(i,j) \in \bar{S}} \frac{\gamma^{i,j,k_i,j} \tilde{A}_i \tilde{B}_j}{(x_g - f_{i,j})} + \mathbf{I}(x_g)
\]

where \( \mathbf{I}(x) \) is a polynomial matrix of maximum degree \( \delta + 2T - 2 \)
The worker then partitions the matrices and encodes them using the encoding parameters:

\[
\begin{align*}
\hat{A}_k &= \sum_{i=1}^{mL_A} \tilde{A}_i a_{i,k}(x_g), \quad \hat{B}_k = \sum_{j=1}^{nL_B} \tilde{B}_j b_{j,k}(x_g) \\
\end{align*}
\]

The worker computes and outputs:

\[
\begin{align*}
\sum_{k=1}^{r} \hat{A}_k \hat{B}_k &= \sum_{k=1}^{r} \sum_{i=1}^{mL_A} \sum_{j=1}^{nL_B} \tilde{A}_i \tilde{B}_j a_{i,k}(x_g) b_{j,k}(x_g) \\
&= \sum_{(i,j)\in Q_k} \gamma^{i,j,k} \frac{\tilde{A}_i \tilde{B}_j}{(x_g - f_{i,j})} + \sum_{i=1}^{mL_A} \sum_{j=1}^{nL_B} \tilde{A}_i \tilde{B}_j \beta_{i,j,k}(x_g) \\
&= \sum_{(i,j)\in \tilde{S}} \gamma^{i,j,k,i,j} \frac{\tilde{A}_i \tilde{B}_j}{(x_g - f_{i,j})} + I(x_g)
\end{align*}
\]

where \(I(x)\) is a polynomial matrix of maximum degree \(\delta + 2\tau - 2\).
The master now gets evaluations of the following function

$$\sum_{(i,j) \in \tilde{S}} \frac{\gamma_{i,j,k} \tilde{A}_i \tilde{B}_j}{(x - f_{i,j})} + I(x)$$
The master now gets evaluations of the following function

\[ \sum_{(i,j) \in \tilde{S}} \frac{\gamma^{i,j,k} \tilde{A}_i \tilde{B}_j}{(x - f_{i,j})} + I(x) \]

This function has \(|\tilde{S}| = |S|mn\) rational terms and \(\delta + 2T - 1 = \frac{|S|mn}{r} + 2T - 1\) polynomial terms
1. The master now gets evaluations of the following function

\[ \sum_{(i,j) \in \tilde{S}} \frac{\gamma^{i,j,k} \tilde{A}_i \tilde{B}_j}{x - f_{i,j}} + I(x) \]

2. This function has \(|\tilde{S}| = |S|m n \) rational terms and
\[ \delta + 2T - 1 = \frac{|S|m n}{r} + 2T - 1 \] polynomial terms

3. Can be interpolated from \((\frac{r+1}{r}) |S|m n + 2T - 1 \) evaluations
Main Result

Achievability

Given parameters $m, n, r$ such that $r | mn$, our scheme achieves

**Recovery Threshold:** $R = \left( \frac{r + 1}{r} \right) |S| mn + 2T - 1$

**Normalized Download Cost:** $\frac{R}{|S|} \times \frac{\alpha^2}{mn} = \alpha^2 \left( \frac{r + 1}{r} + \frac{2T - 1}{|S| mn} \right)$

**Normalized Computational Complexity:** $O \left( \frac{\alpha^3 r}{|S| mn} \right)$

while guaranteeing batch size privacy against any $T$ colluding workers.
Main Result

Achievability

Given parameters $m, n, r$ such that $r \mid mn$, our scheme achieves

Recovery Threshold: $R = \left( \frac{r+1}{r} \right) |S|mn + 2T - 1$

Normalized Download Cost: $\frac{R}{|S|} \times \frac{\alpha^2}{mn} = \alpha^2 \left( \frac{r+1}{r} + \frac{2T-1}{|S|mn} \right)$

Normalized Computational Complexity: $O \left( \frac{\alpha^3 r}{|S|mn} \right)$

while guaranteeing batch size privacy against any $T$ colluding workers.
Main Result

Achievability

Given parameters $m, n, r$ such that $r | mn$, our scheme achieves

**Recovery Threshold:** $R = \left( \frac{r + 1}{r} \right) |S| mn + 2T - 1$

**Normalized Download Cost:** $\frac{R}{|S|} \times \frac{\alpha^2}{mn} = \alpha^2 \left( \frac{r + 1}{r} + \frac{2T - 1}{|S| mn} \right)$

**Normalized Computational Complexity:** $\mathcal{O} \left( \frac{\alpha^3 r}{|S| mn} \right)$

while guaranteeing batch size privacy against any $T$ colluding workers.
Main Result

Achievability

Given parameters $m, n, r$ such that $r | mn$, our scheme achieves

**Recovery Threshold:** \( R = \left( \frac{r + 1}{r} \right) |S| mn + 2T - 1 \)

**Normalized Download Cost:** \( \frac{R}{|S|} \times \frac{\alpha^2}{mn} = \alpha^2 \left( \frac{r + 1}{r} + \frac{2T - 1}{|S| mn} \right) \)

**Normalized Computational Complexity:** \( O \left( \frac{\alpha^3 r}{|S| mn} \right) \)

while guaranteeing batch size privacy against any $T$ colluding workers.
Experimental Simulations

- The following simulations fix the partitioning such that $mn = 24$. 

![Graph showing varied $|S|$, $T = 10$]
The following simulations fix the partitioning such that $mn = 24$. 

![Graph showing better amortization with larger batch sizes]
The following simulations fix the partitioning such that $mn = 24$. 

Better Amortization with Larger Batch sizes
The following simulations fix the partitioning such that $mn = 24$. 

- Better Amortization with Larger Batch sizes
- Overhead due to privacy increases slowly with $T$
Outline

1 Background and Motivation

2 Our Scheme

3 Privacy from the Master
Privacy from the Master (PFM)

Privacy from the master (PFM) is another privacy constraint where the master cannot learn anything beyond what they requested.
Privacy from the Master (PFM)

- Privacy from the master (PFM) is another privacy constraint where the master cannot learn anything beyond what they requested.
- In the context of FPGMM, the master cannot learn anything more about the matrix libraries given the messages it receives, beyond the information that $C_S$ provides.
Privacy from the Master (PFM)

- Privacy from the master (PFM) is another privacy constraint where the master cannot learn anything beyond what they requested.
- In the context of FPGMM, the master cannot learn anything more about the matrix libraries given the messages it receives, beyond the information that $C_S$ provides.
- Batch size privacy incurs significant overhead with current state of the art techniques for PFM.
Current Approaches to Handling PFM

➤ Most coded computation techniques encode desired terms into certain polynomial/rational basis functions.

Example: Generalized CSA codes with Noise Alignment (Chen '21) achieve PFM by adding noise to the polynomial terms (and a few rational terms).
Current Approaches to Handling PFM

➤ Most coded computation techniques encode desired terms into certain polynomial/rational basis functions

➤ Thus, a natural solution to achieve PFM is adding random noise to the basis functions that do not contain the desired terms
Current Approaches to Handling PFM

➤ Most coded computation techniques encode **desired terms** into certain **polynomial/rational basis functions**

➤ Thus, a natural solution to achieve PFM is adding **random noise** to the basis functions that **do not** contain the desired terms

➤ Example: Generalized CSA codes with Noise Alignment (Chen ’21) achieve PFM by adding noise to the polynomial terms (and a few rational terms)
To guarantee PFM, workers need to add a **pessimistic amount** of noise which significantly increases the recovery threshold.
We are currently researching novel techniques to address this issue based on recent advances in coded computation.
We initiated the first investigation into batch size privacy for coded computation

- Introduced the novel problem of FPGMM that highlights the key issues of the new privacy model
- Provided an achievable scheme utilizing CSA-like codes that guarantees privacy, offers good straggler resilience, and provides flexible communication and computation costs
- Highlighted that batch size privacy also complicates other privacy models such as privacy from the master
Summary

➤ We initiated the first investigation into batch size privacy for coded computation
➤ Introduced the novel problem of FPGMM that highlights the key issues of the new privacy model
Summary

➤ We initiated the first investigation into \textbf{batch size privacy} for coded computation.

➤ Introduced the \textbf{novel problem of FPGMM} that highlights the key issues of the new privacy model.

➤ We provided an achievable scheme utilizing CSA-like codes that guarantees privacy, offers good straggler resilience, and provides flexible communication and computation costs.
We initiated the first investigation into **batch size privacy** for coded computation

Introduced the **novel problem of FPGMM** that highlights the key issues of the new privacy model

We provided an achievable scheme utilizing CSA-like codes that guarantees privacy, offers good straggler resilience, and provides flexible communication and computation costs

We highlighted that batch size privacy also complicates other privacy models such as **privacy from the master** and discuss our ongoing work into the topic
References

➤ (Gasca ’89) M. Gasca et al., ”Computation of rational interpolants with prescribed poles”, *Journal of Computation and Applied Math*, 1989


➤ (Sun ’18) H. Sun et al., ”The capacity of private computation.” *IEEE TIT* 2018

➤ (Raviv ’19) N. Raviv et al., ”Private polynomial computation from Lagrange encoding.” *IEEE Transactions on Information Forensics and Security*, 2019

➤ (Kim ’19) M. Kim et al., “Private coded matrix multiplication,” *IEEE Transactions on Information Forensics and Security*, 2019

➤ (Yu ’20) Q. Yu et al.,”Entangled Polynomial Codes for Secure, Private, and Batch Distributed Matrix Multiplication: Breaking the ”Cubic” Barrier”, *ISIT* 2020

➤ (Chen ’21) Chen, Zhen, et al. ”GCSA codes with noise alignment for secure coded multi-party batch matrix multiplication.” *IEEE JSAIT*, 2021

➤ (Zhu ’21) J. Zhu et al., ”Improved constructions for secure multi-party batch matrix multiplication.” *IEEE TCOM* 2021

➤ (Zhu ’22) J. Zhu and S. Li, “A systematic approach towards efficient private matrix multiplication,” *IEEE JSAIT*, 2022