# Harnessing the Power of Distributed Computing: 

# Advancements in Scientific Applications, Homomorphic 

## Encryption, and Federated Learning Security

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science and Engineering
by

Olamide Timothy Tawose

Dr. Dongfang Zhao, Dissertation Advisor
© by Olamide Timothy Tawose 2023
All Rights Reserved

## The Graduate School

We recommend that the dissertation prepared under our supervision by

## Olamide Timothy Tawose

entitled

# Harnessing the Power of Distributed Computing: Advancements in Scientific Applications, Homomorphic Encryption, and Federated Learning Security 

 be accepted in partial fulfillment of the requirements for the degree of
# DOCTOR OF PHILOSOPHY 

Dongfang Zhao, Ph.D., Advisor<br>Lei Yang, Ph.D., Committee Member<br>Tin Nguyen, Ph.D., Committee Member<br>Rui Hu, Ph.D., Committee Member<br>Yan Wang, Ph.D., Graduate School Representative<br>Markus Kemmelmeier, Ph.D., Dean, Graduate School


#### Abstract

Data explosion poses lot of challenges to the state-of-the art systems, applications, and methodologies. It has been reported that 181 zettabytes of data are expected to be generated in 2025 which is over $150 \%$ increase compared to the data that is expected to be generated in 2023. However, while system manufacturers are consistently developing devices with larger storage spaces and providing alternative storage capacities in the cloud at affordable rates, another key challenge experienced is how to effectively process the fraction of large scale of stored data in time-critical conventional systems. One transformative paradigm revolutionizing the processing and management of these large data is distributed computing whose application requires deep understanding. This dissertation focuses on exploring the potential impact of applying efficient distributed computing concepts to long existing challenges or issues in (i) a widely data-intensive scientific application (ii) applying homomorphic encryption to data intensive workloads found in outsourced databases and (iii) security of tokenized incentive mechanism for Federated learning (FL) systems.


The first part of the dissertation tackles the Microelectrode arrays (MEAs) parameterization problem from an orthogonal viewpoint enlightened by algebraic topology, which allows us to algebraically parametrize MEAs whose structure and intrinsic parallelism are hard to identify otherwise. We implement a new paradigm, namely Parma, to demonstrate the effectiveness of the proposed approach and report how it outperforms the state-of-thepractice in time, scalability, and memory usage.

The second part discusses our work on introducing the concept of parallel caching of secure aggregation to mitigate the performance overhead incurred by the HE module in outsourced databases. The key idea of this optimization approach is caching selected radix-
ciphertexts in parallel without violating existing security guarantees of the primitive/base HE scheme. A new radix HE algorithm was designed and applied to both batch and incremental HE schemes, and experiments carried out on six workloads show that the proposed caching boost state-of-the-art HE schemes by high orders of magnitudes.

In the third part, I will discuss our work on leveraging the security benefit of blockchains to enhance or protect the fairness and reliability of tokenized incentive mechanism for FL systems. We designed a blockchain-based auditing protocol to mitigate Gaussian attacks and carried out experiments with multiple FL aggregation algorithms, popular data sets and a variety of scales to validate its effectiveness.

## Acknowledgments

I would like to express my deepest gratitude to my advisor Dr. Dongfang Zhao for the exceptional guidance, caring, patience, and engagement throughout the coiurse of my study at the University of Nevada, Reno. Thank you for all your intellectual and emotional support throughout this dissertation work, especially during hard times. He has been a great friend and guide to me during my Ph.D. study. I am very fortunate to have him as my advisor. I would like to express my sincere gratitude to my committee members, Dr. Lei Yang, Dr. Tin Nguyen, Dr. Rui Hu, and Dr. Yan Wang, for their invaluable intellectual input in improving the dissertation work. I am also grateful to Dr. Eelke Folmer (Departmental Chair), Dr. David Feil-Seifer (Graduate Director), Ms. Erin Keith and the entire past (Heather Lara, Julie Hill, Ashley Ricks, Christina Ruymaekers) and present (Alisa Kader) administrative staff of the CSE office for their unwavering support during my program.

I would also like to thank all the members of the High Performance and Data-Intensive Computing (HPDIC) Lab at UNR with whom I had a great chance to collaborate. Their diverse backgrounds, inspiring suggestions, and in-depth discussions have formed an intellectual environment for interdisciplinary research. I am eternally grateful to my family and dependable friends for their endless love and support. I dedicate all my works in this dissertation to my parents, without whom I couldn't have come this far.

## TABLE OF CONTENTS

Abstract ..... i
Acknowledgments ..... iii
List of Figures ..... xi
1 Introduction ..... 1
1.1 Overview ..... 1
1.2 Dissertation Contributions ..... 2
1.2.1 Topological Modeling and Parallelization of Multidimensional Data
on Microelectrode Arrays ..... 2
1.2.2 Toward Efficient Homomorphic Encryption for Outsourced Databases
through Parallel Caching ..... 3
1.2.3 Gaussian Attacks and Blockchain-based Auditing on Tokenized In-
centive for Federated Machine Learning Systems ..... 4
1.3 Organization ..... 4
2 Topological Modeling and Parallelization of Multidimensional Data on Micro-
2.1 Introduction ..... 6
2.2 Background and Problem Formulation ..... 9
2.2.1 Kirchhoff Laws and Maxwell Cyclomatic Numbers ..... 9
2.2.2 Electrode Array and Graph Abstraction ..... 10
2.2.3 Anomaly Detection through Electrode Arrays ..... 11
2.3 Algebraic-Topological Modeling of MEAs ..... 14
2.3.1 Topology Basics ..... 15
2.3.2 Modeling through Homology Groups ..... 17
2.4 Parallel Processing of Multidimensional Electrode Arrays using Algebraic
Invariant ..... 21
2.4.1 Categorizing Vertex-oriented Constraints ..... 21
2.4.2 Parallelization on MEA Manifolds ..... 25
2.4.3 System Optimization ..... 28
2.4.3.1 Balanced Parallel ..... 28
2.4.3.2 Fine-grained Multiprocessing (PyMP-k). ..... 28
2.5 Implementation and Evaluation ..... 30
2.5.1 Implementation ..... 30
2.5.2 Experimental Setup ..... 31
2.5.3 Computation Time ..... 32
2.5.4 Memory Footprint ..... 33
$2.5 .5 \quad$ I/O Cost ..... 35
2.5.6 Scalability ..... 35
2.6 Related Work ..... 36
2.7 Summary ..... 37
3 Toward Efficient Homomorphic Encryption for Outsourced Databases through
Parallel Caching ..... 39
3.1 Introduction ..... 39
3.1.1 Background and Motivation ..... 39
3.1.2 Contributions ..... 41
3.2 Preliminaries and Related Work ..... 42
3.2.1 Confidentiality of Outsourced Data ..... 42
3.2.2 Homomorphic Encryption ..... 44
3.2.3 Provable Security ..... 47
3.3 RHE: Radix Homomorphic Encryption ..... 49
3.3.1 Overview ..... 49
3.3.2 Algorithm ..... 50
3.3.3 Parameterization ..... 52
3.3.3.1 Heuristic Radix Selection ..... 52
3.3.3.2 Optimal Radix in the Worst Case ..... 53
3.4 Rache: Radix-Additive Caching for Homomorphic Encryption ..... 55
3.4.1 Security Definitions and Assumptions ..... 55
3.4.2 Scheme Description ..... 56
3.4.3 Provable Security ..... 58
3.5 Incremental Rache ..... 59
3.5.1 Overview ..... 59
3.5.2 Definitions and Notations ..... 60
3.5.3 Scheme Description ..... 61
3.5.4 Provable Security ..... 63
3.6 Evaluation ..... 64
3.6.1 Objectives ..... 64
3.6.2 Experimental Setup ..... 65
3.6.2.1 Systems and Implementation ..... 65
3.6.2.2 Configurations ..... 66
3.6.2.3 Workloads ..... 67
3.6.3 Performance with and without Homomorphic Encryption ..... 68
3.6.4 Batch Rache ..... 69
3.6.4.1 Encryption vs. Addition ..... 69
3.6.4.2 TPC-H ..... 69
3.6.4.3 Random Numbers ..... 70
3.6.4.4 U.S. COVID-19 Statistics ..... 71
3.6.4.5 Human Genome Reference 38 . ..... 71
3.6.4.6 Bitcoin Trade Volume ..... 72
3.6.4.7 Scalability ..... 73
3.6.5 Incremental Rache ..... 74
3.6.5.1 TPC-H ..... 74
3.6.5.2 Random Numbers ..... 74
3.6.5.3 Aggregating Encrypted Fields ..... 75
3.6.5.4 Computing Nuances On-the-fly ..... 77
3.6.6 Summary of Experimental Results ..... 77
3.7 Summary ..... 78
4 Gaussian Attacks and Blockchain-based Auditing on Tokenized Incentive forFederated Machine Learning Systems79
4.1 Introduction ..... 79
4.2 Background and Related Work ..... 82
4.2.1 Incentive Mechanisms for Federated Learning ..... 82
4.2.2 Attacks on Federated Learning ..... 83
4.2.3 Blockchains for Federated Learning ..... 84
4.3 Gaussian Attack on Token Incentive for FL ..... 85
4.3.1 Threat Model ..... 85
4.3.2 Attack Algorithm ..... 86
4.3.3 Convergence Analysis ..... 87
4.4 Blockchain Audit for Local Models in FL ..... 91
4.4.1 Assumptions ..... 91
4.4.2 Auditing Protocol ..... 91
4.4.3 Complexity Analysis ..... 92
4.5 Evaluation ..... 93
4.5.1 Implementation ..... 93
4.5.2 Experimental Setup ..... 94
4.5.2.1 Test Bed ..... 94
4.5.2.2 Data Sets ..... 95
4.5.2.3 Machine Learning Models. ..... 95
4.5.2.4 Federated Learning Parameters ..... 96
4.5.3 Effectiveness of Gaussian Attacks ..... 96
4.5.4 Cost of Blockchain Auditing ..... 99
4.6 Summary ..... 101
5 Conclusions and Future works ..... 102
5.1 Conclusions ..... 102
5.2 Future works ..... 103
Bibliography ..... 105

## LIST OF FIGURES

2.1 Abstract architecture of a three-dimensional $3 \times 3$ electrode array, in
a physical device. Three horizontal wires $(A, B$, and $C)$ and three vertical
wires $(I, I I$ and $I I I)$ are interconnected through the 18 joints $\{0, \cdots, 17\}$ andnine resistors $\left(R_{i j}, 1 \leq i, j \leq 3\right)$.11
2.2 Abstraction of a general electrode array in two-dimensional space. ..... 13
2.3 A polyhedron of two simplices (triangles $\{a, b, c\}$ and $\{d, e, f\}$ ) that is nota simplicial complex. The overlap of two triangles is segment $\{b, f\}$, whichis not an element of the set of 1-simplices $\{\{a, b\},\{b, c\},\{a, c\},\{d, e\},\{d, f\},\{e, f\}\}$.17
2.4 A corresponding topology between $C$ and $I$ as Figure 2.1. There alsoexists nine paths between $C$ and $I$, which are semantically equivalent toFigure 2.1|but with possible loops across sub-paths.23
2.5 A corresponding topology between the $i$-th horizontal axis and the $j$-th
vertical axis in a $n \times n$ Array, in a 2D space. There are total $(n-1) U a$ 'sand ( $n-1$ ) Ub's as two sets of distinct voltage values for a specific pairof $i j$ end points, assuming the original voltage can be measured between$i$ and $j$. The conditions of subscripts are: $k \in\{1, \cdots, j-1, j+1, \cdots, n\}$;$m \in\{1, \cdots, i-1, i+1, \cdots, n\} ; k^{\prime}=k$ if $k \leq j, k^{\prime}=(k-1)$ otherwise; and$m^{\prime}=m$ if $m \leq i, m^{\prime}=(m-1)$ otherwise.23
2.6 Various approaches for parallel formulation of Kirchhoff law equations. The proposed joint constraints enable various parallelization possibilities, namely Parallel, Balanced Parallel, and PyMP. PyMP delivers the highest performance at scales $n \geq 20$, despite of lower performance than Balanced Parallel at $n=10$ where the parallelization overhead outweighs the speedup.32
2.7 Computation time of various parallelism in PyMP. Applying fine-grainedmultiprocessing leads to a linear decrease in the overall compute time perworkload at scales $n \geq 20$.33
2.8 Cummulative Distribution Functions (CDFs) of Memory Usage. Forall the scales of $n \in\{10 . .100\}$, the peak memory usage is about the sameregardless of data parallelism.34
2.9 The end-to-end time of various degrees of parallelism in PyMP, in-cluding disk I/Os. Utilization of more threads $k \geq 2$ starting from a lowrank of $n=20$ makes significant effect to the overall I/O time.35
2.10 Scalability of Parma across various number of processes and varyingworkloads.36
3.1 Performance with and without encryption schemes. ..... 69
3.2 Homomorphic encryption and addition in Paillier. ..... 69
3.3 Performance comparison on the TPC-H benchmark. ..... 69
3.4 Encoding performance on random numbers. ..... 71
3.5 Encoding the U.S. COVID-19 statistics. ..... 71
3.6 Encoding the human genome reference 38. ..... 71
3.7 Encoding the Bitcoin trade volume. ..... 72
3.8 Weak scaling of the encryption of random numbers. ..... 72
3.9 Encoding a variety of workloads with a fix number of 32 cores. ..... 72
3.10 Performance comparison on TPC-H (scale $=100$ ), 20,000,000 tuples in
table Part. ..... 74
3.11 Performance comparison of Symmetria and Rache on 1,024 random plain-
texts.74
3.12 Performance overhead incurred by pivots and nuances when encrypting
$2^{32}$ random plaintexts. ..... 74
3.13 Time breakdown of aggregating 200,000 tuples of table Part in TPC-H. ..... 76
3.14 Aggregating time with different numbers of pivots on different TPC-H scales. ..... 76
3.15 Rache speedup over Symmetria when computing nuances on-the-fly. ..... 76
4.1 Gaussian attack and blockchain audit for decentralized federated learning ..... 81
4.2 FedAvg loss under Gaussian attacks for MNIST, Fashion-MNIST, CIFAR-
10 , and SVHN ..... 97
4.3 MultiKrum loss under Gaussian attacks for MNIST, Fashion-MNIST, CIFAR-
10, and SVHN ..... 98
4.4 FedAvg accuracy under Gaussian attacks for MNIST, Fashion-MNIST,

CIFAR-10, and SVHN] . . . . . . . . . . . . . . . . . . . . . . . . . . . . 98
4.5 MultiKrum accuracy under Gaussian attacks for MNIST, Fashion-MNIST,

CIFAR-10, and SVHN]. . . . . . . . . . . . . . . . . . . . . . . . . . . . 99
4.6 Computational and I/O costs of blockchain auditing for MNIST, Fashion-

MNIST, CIFAR-10, and SVHN
100

## CHAPTER 1

## INTRODUCTION

### 1.1 Overview

Data explosion has become a phenomenon in this era of data deluge where people have more tools to create and share information which in turn has significantly increased the storage requirements on computer systems. As indicated in a report by Statista [112] and Bernard Marr \& Co [16], 181 zettabytes of data are expected to be generated in 2025 which is over $150 \%$ increase compared to the data that is expected to be generated in 2023. However, while system manufacturers are consistently developing devices with larger storage spaces and providing alternative storage capacities in the cloud at affordable rates, another key challenge experienced is how to effectively process the fraction of large scale of stored data in time-critical conventional systems utilizing the most of available computing resources which usually requires the analyst to have in-depth knowledge from both computer systems and domain sciences in order to achieve desired characteristics like scalability, increased performance, efficient resource utilization and enhanced reliability.

A paradigm shift transforming how we approach complex problems and process these enormous amounts of data stored has evolved, and it is known as distributed computing. The application of distributed computing is a plausible approach that could help achieve the afore-mentioned characteristics. However, a deep understanding of suitable distributed computing concepts is always required to drive innovation, meet the ever-increasing challenges, and unfold new intrinsic possibilities across various domains as there might not be
a one-size-fits-all or universal approach for all cases. This dissertation focuses on exploring the potential impact of applying efficient distributed computing concepts to long existing challenges or issues in (i) a widely data-intensive scientific application (ii) applying homomorphic encryption to data intensive workloads found in outsourced databases and (iii) security of tokenized incentive mechanism for Federated learning (FL) systems.

### 1.2 Dissertation Contributions

This section briefly highlights the contributions of this dissertation.

### 1.2.1 Topological Modeling and Parallelization of Multidimensional Data on Microelectrode Arrays

Microelectrode arrays (MEAs) are physical devices widely used in various science and engineering fields. One common computational challenge when applying a high-density MEA (i.e., a larger number of wires, more accurate locations of abnormal cells) is how to efficiently compute those resistance values provided the nonlinearity of the system of equations with the unknown resistance values per the Kirchhoff law. This part proposes an algebraic-topological model for MEAs such that we can identify the intrinsic parallelism that cannot be identified by conventional approaches. We implement a system prototype called Parma based on the proposed topological methodology. Experimental results show that Parma outperforms the state-of-the-practice in time, scalability and memory usage: the computation time is two orders of magnitude faster on up to 1,024 cores with almost linear
scalability and the memory is much better utilized with proportionally less warm-up time with respect to the number of concurrent threads.

### 1.2.2 Toward Efficient Homomorphic Encryption for Outsourced Databases through Parallel Caching

Many applications deployed to public clouds are concerned about the confidentiality of their outsourced data, such as financial services and electronic patient records. A plausible solution to this problem is homomorphic encryption (HE), which supports certain algebraic operations directly over the ciphertexts. The downside of HE schemes is their significant, if not prohibitive, performance overhead for data-intensive workloads that are very common for outsourced databases, or database-as-a-serve in cloud computing. The objective of this work is to mitigate the performance overhead incurred by the HE module in outsourced databases. To that end, this part proposes a radix-based parallel caching optimization for accelerating the performance of homomorphic encryption (HE) of outsourced databases in cloud computing. The key insight of the proposed optimization is caching selected radixciphertexts in parallel without violating existing security guarantees of the primitive/base HE scheme. We design the radix HE algorithm and apply it to both batch- and incrementalHE schemes; we demonstrate the security of those radix-based HE schemes by showing that the problem of breaking them can be reduced to the problem of breaking their base HE schemes that are known IND-CPA (i.e. Indistinguishability under Chosen-Plaintext Attack). We implement the radix-based schemes as middleware of a 10 -node Cassandra cluster on CloudLab; experiments on six workloads show that the proposed caching can boost state-of-the-art HE schemes, such as Paillier and Symmetria, by up to five orders of magnitude.

### 1.2.3 Gaussian Attacks and Blockchain-based Auditing on Tokenized Incentive for Federated Machine Learning Systems

Federated learning (FL) has emerged as a new distributed computing paradigm to both enrich the available training data and protect the data privacy of participating clients. Due to the critical importance of client participation leading to the success of FL systems, multiple incentive mechanisms have been proposed to attract and retain clients in FL; in particular, a tokenized incentive was recently proposed, which was believed more practical than the existing monetary-based, offline incentive mechanisms. However, this part will demonstrate that, under mild assumptions, the tokenized incentive mechanism for FL systems can be effectively compromised by a fraction of colluded clients who share their local training models with deliberate Gaussian noises. To that end, we design a blockchain-based protocol such that a client suspected to have launched a Gaussian attack will be detected. We have implemented the proposed Gaussian attack and Blockchain-based auditing with FedML; Extensive experiments demonstrate the effectiveness of the Gaussian attack and the efficiency of blockchain audit with reasonable overhead (less than $10 \%$ training time).

### 1.3 Organization

The next chapters are organized as follows. Chapter 2 focus on addressing the long-existing computational challenge of multidimensional data in one of the most widely used engineering devices, namely microelectronic array (MEA) used for (almost) real-time anomaly detection such as potential cancer regions, the wound surface of an injured athlete. Chapter 3 introduces the concept of parallel caching of secure aggregation to mitigate the per-
formance overhead incurred by the homomorphic encryption (HE) module in outsourced databases. Chapter 4 addresses an important issue in federated learning: the security of tokenized incentives for participating clients in a federated machine learning system. Chapter 5 presents the conclusion of this dissertation and discusses some of the interesting areas that have been identified for future work.

## CHAPTER 2

# TOPOLOGICAL MODELING AND PARALLELIZATION OF MULTIDIMENSIONAL DATA ON MICROELECTRODE ARRAYS 

### 2.1 Introduction

Microelectrode arrays (MEAs) are physical devices widely used in various science and engineering fields. For example in the pandemic of COVID-19, electrode arrays were involved in both vaccine development [68] and fast testing methods [6,61]. More conventionally: (i) in biomedical engineering [133], an MEA can be applied to a patient's wound surface and report the anomalies of the skin; (ii) in biological sciences [90], an MEA can be placed on a cell medium to electronically detect the potential cancer regions; and (iii) in electronic engineering [22,74], similar techniques are applied for the trade-offs between currents and signals in the very-large-scale integration (VLSI) design of CPU chips. These applications utilize multidimensional arrays as the default format for storing and managing large volumes of measurement data.

One of the most notable limitations of applying MEAs lies in its scalabilty: an MEA cannot be efficiently parametrized due to the complicated, nonlinear equations. Formally, the parameterization of an MEA aims to quantify the physical resistance of the devices given four inputs: (i) the topology of the MEA, (ii) the context where the MEA is placed, e.g., cell medium, patient skin; (iii) a provided voltage, e.g., 5 volts, and (iv) measured current values in the MEA. While in the real-world applications we can easily control the voltage, accurately measure the currents, and discreetly choose the object/context, the
main challenge lies in the arbitrary topology exhibited by MEAs: the complexity of an MEA topology can lead to considerable computational time that is considered impractical for applications. For instance, it takes hours to parameterize a two-dimensional $64 \times 64$ MEA [121]. Specifically, the electrical resistance values in the complex circuit cannot be efficiently computed due to a large number of circuits at a very fine granularity and the nonlinearity of the system of equations with the unknown resistance values per the Kirchhoff law [116]. Kirchhoff law is one of the most fundamental laws governing the physical characteristics of electrode arrays. In practice, the law is applied repeatedly to every entity in the electronic device and more importantly, the equations are correlated and thus hard to be parallelized. To make it worse, if the internal resistors are unknown, the system of equations becomes nonlinear, making the problem prohibitively expensive to solve analytically. Conventional computational approaches include Landweber method [113], linear back projection [7], and Tikhonov regularization methods [118], all of which exhibit an ill-posed computational problem [12,73]: the solution is largely dependent on the input and results in an unacceptable variance, which hinders its adoption in practice.

From a computational point of view, researchers have recently started to seek non-analytic paradigms such as machine learning to estimate the solution, e.g., training a convolutional neural network to approximate the unknown resistor distribution in an electrode array [116]. This approach is demonstrated as an effective means to "learn" the nonlinear function between inputs and outputs: the error rate is as low as $0.49 \%$. In [89], the authors demonstrated a $20 \times 20$ microelectrode array device manufactured in a wet lab, and showed that a graph-theoretical conversion from the original MEA data allowed them to efficiently utilize storage space for the expensive computation. Later, Wang et al. [121] presented a forward labeling technique for effectively training an artificial neural network (ANN) to predict the unknown variables in the $64 \times 64$ MEA, which was more than two orders of magnitude larger than the one shown in [116]. While the ANN can be efficiently trained, how to col-
lect the training data, i.e., parameterizing the MEAs, at such scales pose unprecedented challenges in terms of computation cost. The problem is further exacerbated by the fact that the intrinsic parallelism, if any, does not appear observational.

While aforementioned work focuses on adopting machine learning for an estimated parametrization of MEAs, this chapter tackles the MEA-parametrization problem from an orthogonal viewpoint enlightened by algebraic topology, which allows us to algebraically parametrize MEAs whose structure and intrinsic parallelism are hard to identify otherwise. Firstly, the seemingly complex, interconnected circuit flows among MEA nodes can be abstracted and simplified as a series of abstract complex, a well-studied object in algebraic topology that we will detail shortly. Secondly, we can apply homological analysis of the abstract complex and extract the independent high-dimensional circles for parallelization. We show that the algebraic objects represented by the MEA data are well defined and further, form the highly-desired topological invariant, namely homology groups, under rigorous grouptheoretical analysis. The algebraic invariant, such as Betti numbers, allows us to employ a fine-grained parallelization technique for applying Kirchhoff's laws concurrently, each of which works by itself on a $k$-dimensional cycle.

To demonstrate the effectiveness of the proposed approach, we implement a new paradigm, namely Parma. Preliminary results show that the proposed approach outperforms the state-of-the-practice in various metrics: (i) the computation time is three orders of magnitude faster; (ii) the I/O time is proportionally reduced with multithreading; (iii) the memory is better utilized with proportionally less warm-up time with respect to the number of concurrent processes/threads.

In summary, this chapter discusses the following contributions:

- We take an algebraic-topological approach to model the parametrization of MEAs that involves computationally-intensive Kirchhoff laws; ( $\$ 2.3$ )
- We propose a new parallelization paradigm, which identifies the high-dimensional "holes" that can be computed in parallel; ( $\$ 2.4$ )
- We implement a prototype system called Parma that is extensively evaluated on various test beds at large scales of up to 1,024 cores. ( $\$ 2.5$ )


### 2.2 Background and Problem Formulation

### 2.2.1 Kirchhoff Laws and Maxwell Cyclomatic Numbers

In Kirchhoff's 1847 paper, he proved that the currents of a direct circuit could be uniquely determined by two sets of linear equations given fixed source voltage and wire resistance. The two systems of linear equations are also called the first and second Kirchhoff's laws. The first law ( $L 1$ ) states that overall flow at a specific vertex is zero, and the second law (L2) states that the overall voltage change along a loop of edges stays the same. If we model a circuit as a graph $G(V, E)$, then there are $|V|$ equations of $L 1$, and there are $|E|$ unknown currents. It can be shown that the $|V|$ equations of $L 1$ are not independent, while any $|V|-1$ equations are indeed independent. Consequently, we need to have $|E|-|V|+1$ or more equations from $L 2$ to find the $|E|$ unknown currents. It can be further shown that these $|E|-|V|+1$ equations from $L 2$ are all independent of $|V|$ equations in $L 1$, indicating that both $L 1$ and $L 2$ equations can collectively determine the current values. While Kirchhoff proved this for the physical case where resistances are positive real numbers, a more general case can be proven using algebraic topology, i.e., the introduction of cochain and coboundary,
see [44] for more details.

The number of independent loops represented by $|E|-|V|+1$ is historically called the $c y$ clomatic number by Maxwell in the context of the circuit and is an important topological property in graph theory. It should be noted, however, in many engineering applications, Kirchhoff's laws are applied indirectly: the currents are sometimes easy to measure, and it is the resistance that is of interest and yet unknown, making the systems of equations nonlinear.

### 2.2.2 Electrode Array and Graph Abstraction

Electrode Arrays are widely used in biomedical engineering, electrical engineering, and mechanical engineering. A typical $n \times n$ dimensional electrode array consists of a set of horizontal and vertical wires, joined through point-wise resistors. The $n$ 's scale or size highly depends on the application under consideration. For example, a continuous-flow device [90] used for the geometric screening of core/shell hydrogel microcapsules consists of 15 electrode pairs (i.e., $n=15$ ), whereas a device designed for 2D electrical imaging surveys can consist of more than 20 electrode pairs [77]. Overall, a $n \times n$ array comprises $2 n^{2}$ joints/junctions and $n^{2}$ resistors.

An example of such physical system is shown in Figure 2.1 with a size of $n=3$. It consists of three horizontal wires $(A, B$, and $C)$ and three vertical wires ( $I, I I$ and $I I I$ ) connected with nine resistors resulting into a total of 18 joints $\{0, \cdots, 17\}$.

In general, a $n \times n$ electrode array can be abstracted into a two-dimensional graph where


Figure 2.1: Abstract architecture of a three-dimensional $3 \times 3$ electrode array, in a physical device. Three horizontal wires ( $A, B$, and $C$ ) and three vertical wires ( $I, I I$ and $I I I$ ) are interconnected through the 18 joints $\{0, \cdots, 17\}$ and nine resistors ( $R_{i j}, 1 \leq i, j \leq 3$ ).
each vertex represents a resistor, as shown in Figure 2.2. In practice, the physical device of an electrode array is usually in a square shape, although the following discussion can be trivially extended to arbitrary shapes $m \times n(m \neq n)$.

### 2.2.3 Anomaly Detection through Electrode Arrays

To put it in the real-world context, the electrode device is used for (almost) real-time anomaly detection such as the wound surface of an injured athlete. A common workload of such a $n \times n$ electrode array system is to find out the unknown resistances $R_{i j}$ 's $(1 \leq i, j \leq n)$ given the pair-wise measured resistance $Z_{i j}$ 's between the end-points of $n$ horizontal and $n$ vertical wires. These $R_{i j}$ 's are usually equal and negligible in values from the device. When the tested medium exhibits anomaly areas (e.g., cancer cells), the local resistance (i.e., $R$ ) will significantly increase. Therefore, we have $n^{2}$ unknowns ( $R$ 's) and also $n^{2}$ measured
values ( $Z$ 's), namely $Z_{A, I}, Z_{B, I}, Z_{C, I}, Z_{A, I I}, Z_{B, I I}, Z_{C, I I}, Z_{A, I I I}, Z_{B, I I I}$, and $Z_{C, I I I}$ in the example shown in Figure 2.1.

The challenge here lies in that the measured resistance between two endpoints is a nonlinear function of all the $n^{2}$ unknowns through many possible paths; To see this, take $Z_{B, I I I}$ for example, the most straightforward circuit is through $R_{32}$ (between endpoints 14 and 15). And yet, there are other circuits flowing through, one possible path being:

$$
B \rightarrow 8 \xrightarrow{R_{22}} 9 \rightarrow 7 \xrightarrow{R_{21}} 6 \rightarrow 12 \xrightarrow{R_{33}} 13 \rightarrow I I I
$$

More examples from other pairs of endpoints include (cf. Figure 2.1):

$$
\begin{aligned}
& C \rightarrow 16 \xrightarrow{R_{33}} 17 \rightarrow 15 \xrightarrow{R_{32}} 14 \rightarrow 2 \xrightarrow{R_{12}} 3 \rightarrow I \\
& A \rightarrow 12 \xrightarrow{R_{31}} 13 \rightarrow I I I \\
& C \rightarrow 10 \xrightarrow{R_{23}} 11 \rightarrow 9 \xrightarrow{R_{22}} 8 \rightarrow 14 \xrightarrow{R_{32}} 15 \rightarrow I I I \\
& B \rightarrow 2 \xrightarrow{R_{12}} 3 \rightarrow 5 \xrightarrow{R_{13}} 4 \rightarrow 10 \xrightarrow{R_{23}} 11 \rightarrow I I \\
& A \rightarrow 0 \xrightarrow{R_{11}} 1 \rightarrow 3 \xrightarrow{R_{12}} 2 \rightarrow 8 \xrightarrow{R_{22}} 9 \rightarrow I I
\end{aligned}
$$

For a $n \times n$ array, there are overall $n^{(n+1)}$ possible paths. To see this, we can start with a specific pair of endpoints. Whenever the circuit flows from one joint to the next step, there are $n$ possible choices. In total, there are only $(n-1)$ steps between the source and the destination. Therefore, there are $n^{(n-1)}$ possibilities between any pair of endpoints. Note that there are a total $n^{2}$ pairs of endpoints. Consequently, the total number of paths is $n^{(n-1)} \cdot n^{2}=n^{(n+1)}$. To save all of these possible paths, the required space is even larger than the $n$ exponential because each path has to store all the joint numbers as well. In [89],


Figure 2.2: Abstraction of a general electrode array in two-dimensional space.
authors reported that the data growth is so fast that the path-based approach is unfeasible on mainstream computer hardware and systems when $n>6$.

If we assume the paths can be stored efficiently (which is true only for small $n$ 's), then the question is how to find those paths efficiently. This is a classical problem in graph theory, which is solvable using either depth-first or breadth-first recursive traversal algorithms. Since the number of possible paths is exponential, any algorithm for finding them must be at least exponential, which is indeed the case of both depth-first and breadth-first recursive traversal algorithms. To see this, again, in the case of breadth-first recursion, there are $n$ neighbors to the current position, and each of the $n$ neighbors might lead to a depth of ( $n-1$ ), resulting in the total of an exponential number of recursion calls.

After finding out and storing the paths, the next step is to solve the equations built upon the paths satisfying the constraints, i.e., the total incoming circuit flow is equal to the total outgoing circuit flow according to the Kirchhoff law. In essence, all the paths are considered
as the parallel circuit flows between two endpoints and can be aggregated through this form:

$$
Z_{i j}^{-1}=\sum_{k=1}^{n^{(n-1)}} P_{k}^{-1}(R)
$$

where $P_{k}(R)$ indicates the summation of resistors along the $k$-th path between the $i$-th horizontal wire and the $j$-th vertical wire in a $n \times n$ array. Therefore, the goal is to solve a system of $n^{2}$ nonlinear equations, each of which comprises an exponential number of terms, and each term exhibits a summation of selected unknowns as the divisor. This equation-solving procedure itself is also compute-intensive, requiring iterative method to find roots of the unknown resistors. The state-of-the-art is to leverage deep learning to estimate the unknowns, e.g., conventional neural networks [116]. Once the $R$ values are known (or, estimated), the anomaly can be simply detected.

### 2.3 Algebraic-Topological Modeling of MEAs

This section will first briefly review the basics of algebraic topology and show the natural correspondence between MEAs and the topological objects such as simplex and simplicial complex. We will then demonstrate that this correspondence is mathematically sound, based on which of those topological objects can form more sophisticated ones that exhibit strong and otherwise unnoticeable algebraic invariant, including but not limited to homology groups and Betti numbers. As a result, the proposed modeling and analysis lead to a new parallelization paradigm that will be discussed in $\$ 2.4$.

### 2.3.1 Topology Basics

Mathematically speaking, a topology of a set $S$ is a collection of subsets of $S$, denoted $\mathcal{T}$, satisfying certain properties that distinguish a topology from the set of hyperedges in a hypergraph [111]. One example topology of $S$ is then the power set of $S, \mathcal{P}(S)$, which consists of all the possible $2^{|S|}$ subsets of $S$. This is also called the discrete topology of $S$. The tuple $(S, \mathcal{T})$ is called the topological space of $S$. Each of the subsets $U$ from $\mathcal{T}$ is called an open set, and the complement set $S \backslash U$ is a closed set by definition. A function $g$ from space $X$ to $Y$ is called continuous if $\forall v$ is an open set in $Y$, then $g^{-1}(v)$ is an open set in $X$. The composition of two continuous functions is also continuous. If both $g$ and $g^{-1}$ are continuous and bijective (one-on-one mapping), we call $g$ a homeomorphism. Because a homeomorphism is defined purely on open and closed sets, two topological spaces are considered equivalent if such a homeomorphism exists. Usually, we expect to migrate a complex problem in one topological space to another such that the problem can be solved more efficiently or more intuitively. The aforementioned concepts and techniques are also referred to as point-set topology.

In addition to point-set topology, there is another branch of algebraic-topological methods that study homotopy groups and homology groups. Informally, these groups break the complex down into the smaller pieces and map the geometrical objects into algebraic objects, such as groups. Some of the hardest problems were shown to be elegantly solvable through algebraic topology [46]. Remarkably, a unique subbranch of topology, namely combinatorial topology, specifically studied the topological properties of distributed computing models [55, 56].

[^0]The building blocks we are interested in for topological modeling of MEAs are called simplices (the plural form of simplex). In this work, by simplex $\sigma$ we mean an abstract simplex, defined as a set $S$ of vertices. Any subset of $\sigma$ is also a simplex, and is called a face of $\sigma$. The dimension of $\sigma$ is defined as the number of vertices minus 1:

$$
\operatorname{dim} \sigma=|\sigma|-1
$$

Geometrically, a simplex $\sigma$ consists of all the possible points, edges, triangles, tetrahedrons, and higher-dimensional objects that can be composed of the vertices in $S$. From a combinatorial perspective, a collection of $\sigma$ 's can be thought of as an object representing more sophisticated relationships among the vertices in $S$, which is called an abstract simplicial complex, denoted $K$. The dimension of a complex is defined as the highest dimension from any simplex in the complex:

$$
\operatorname{dim} K=\max (\operatorname{dim} \sigma), \forall \sigma \in K
$$

It is "simplicial" in the sense that any $\sigma_{1} \cap \sigma_{2} \in \sigma_{1}, \sigma_{2}$, meaning that the simplices (including the empty set $\emptyset$ ) shared by $\sigma_{1}$ and $\sigma_{2}$ must also be valid simplices of both $\sigma_{1}$ and $\sigma_{2}$. This requirement might sound self-evident, but actually might be violated in practice: Figure 2.3 shows that the shared line segment $\{b, f\}$ is not an element of

$$
\{\emptyset,\{a\},\{b\},\{c\},\{a, b\},\{b, c\},\{a, c\},\{a, b, c\}\} .
$$



Figure 2.3: A polyhedron of two simplices (triangles $\{a, b, c\}$ and $\{d, e, f\}$ ) that is not a simplicial complex. The overlap of two triangles is segment $\{b, f\}$, which is not an element of the set of 1 -simplices $\{\{a, b\},\{b, c\},\{a, c\},\{d, e\},\{d, f\},\{e, f\}\}$.

### 2.3.2 Modeling through Homology Groups

It follows that an MEA can be represented by an abstract simplicial complex, or complex if no ambiguity arises. Before we go on the discussion on modeling MEAs with complexes, we need to verify that the MEA can be indeed abstracted as a complex, per the above definition.

Proposition 1. Every microelectrode array is an abstract simplicial complex with the set of vertices represented by the joints between wires.

Proof. We will prove this for the two-dimensional case; higher-dimensional cases follow similarly.

First, we show that the dimension of a 2-dimensional MEA is one. Let $P$ denote the polyhedron of the MEA object. We will show that the dimensions of all simplices are (i) larger than or equal to one and (ii) smaller than or equal to one, both of which will collectively prove our claim. For (i), suppose, for contradiction, that $\operatorname{dim} P<1$, that is, $\operatorname{dim} P=0$. However, a 0 -dimensional complex has only vertices without any edges,
indicating an MEA with joints without wires, which is impossible, thus a contradiction. For (ii), suppose, again for contradiction, that $\operatorname{dim} P>1$. We will use induction to show that $\operatorname{dim} P$ cannot be any numbers larger than 1 . We first check $\operatorname{dim} P \neq k, k=2$. This is easy to verify since if $\operatorname{dim} P=2$, there must be at least one triangle in $P$, whose dimension is 2 . However, in 2-dimensional MEAs, there are only vertical and horizontal wires, and forming triangles is not possible. Now we start checking $k+1$. Recall that by the definition of simplex, any subset of a simplex (i.e., a face) is again a simplex. It follows that if $\gamma$ is a $(k+1)$-dimensional simplex, then its subset, say a $k$-dimensional simplex $\sigma$ must be a simplex. But we just show that a simplex cannot have dimension $k$, starting $k=2$, leading to a contradiction.

Second, we show that any shared portion between two simplices in an MEA is a face of both simplices. Because the dimension of a 2-dimensional MEA is 1 , as shown above, we only need to verify that the shared simplex is either a common vertex or a shared edge. It is trivial to check the shared edge, however, because that would indicate that the two wiresegments overlay each other. Therefore, we only need to show that the only intersection between any two edges is their joint (or nothing if they are parallel, which will be covered at the end of this proof). This is indeed the case because otherwise, the two segments would touch each other on two endpoints, making them identical. Lastly, to complete the proof, if two simplices do not share any simplex (e.g., they are parallel wires along the vertical or horizontal axes), then their intersection is $\emptyset$ and belongs to both simplices.

Having shown that an MEA is a complex allows us to explore strong properties that have been extensively studied in algebraic topology. Recall that Kirchhoff's laws say that the voltage changes over the "loop" of a circuit. This property can be accurately and efficiently characterized by the topological invariant called homology groups. We do not have space
to elaborate either homology or groups, introductory texts on these topics can be found in [37, 51]. In the following, we will give a very brief overview of the concepts when they are absolutely necessary for our discussion. A group is a set $G$ along with a binary operation $\star$ between two elements in $G$ such that $\star$ is closed and associative in $G$, and there is a special identity element $e \in G$ such that any element has a counterpart to which its multiplication equals $e$ :

$$
\begin{aligned}
& \forall g, h, k \in G, g \star h \in G,(g \star h) \star k=g \star(h \star k), g \star e=g, \\
& \exists g^{-1} \in G \text { such that } g \star g^{-1}=e .
\end{aligned}
$$

Now, think of the set $C$ consists of all the possible subsets of line segments in an MEA, and let us define the binary operation between any pair of subsets as modulo-2 inclusion, meaning that any duplicate simplices will cancel out. So, two 1 -dimensional simplices (i.e., edges), say $\sigma_{1}=\{a, b\} \in G$ and $\sigma_{2}=\{b, c\} \in G$, can be calculated as

$$
\sigma_{1} \star \sigma_{2}=\{a, c\}
$$

This group is called the complex chain group in the literature of algebraic topology, usually denoted $C$. Obviously, some elements of $C$ are cycles and others are not; for example in Figure 2.1, a sequence $0 \rightarrow R_{11} \rightarrow 1 \rightarrow 3 \rightarrow R_{12} \rightarrow 2 \rightarrow 8 \rightarrow R_{22} \rightarrow 9 \rightarrow 7 \rightarrow R_{21} \rightarrow$ $6 \rightarrow 0$ is a cycle. We are interested in this subset of cycles, denoted $D$, because they are closely related to the Kirchhoff laws. Obviously, if we apply the defined modulo-2 operation along the cycle, the eventual result would be empty (i.e., the identity element in $C$ ); in fact, there is another name to summarize the series of modulo- 2 operations above, bound$a r y{ }^{2}$ denoted $\delta$. It is easy to verify that the boundary $\delta$ can map the set of $k+1$-dimensional

[^1]simplices into $k$-dimensional simplices, which results in the following sequence:
$$
\cdots \xrightarrow{\delta} C^{k} \xrightarrow{\delta} C^{k-1} \xrightarrow{\delta} \cdots C^{1} \xrightarrow{\delta} C^{0},
$$
where $C^{k}$ denotes a $k$-dimensional complex chain, or a $k$-chain group. The result, or image, of the $\delta$ operation, is a subset of simplices, called the boundary group, denoted $B^{k}$, and is called $k$-boundary group at dimension $k$. In group theory, the preimage of the empty image, $\delta^{-1}(e)$, is called the kernel of the map $\delta$; therefore, $D^{k}$ is the kernel of $\delta$ whose result is $e \in C^{k-1}$. $D^{k}$ is called the $k$-cycle group.

We can then define $H^{k}=D^{k} / B^{k}$, the quotient group at each dimension, which also compose a series of groups, also called the homology groups. The order, or cardinality, i.e., the number of elements, of these quotient groups can be calculated as:

$$
\left|H^{k}\right|=\left|D^{k}\right| /\left|B^{k}\right|,
$$

according to the Lagrange Law (in group theory). Because the chains of groups in simplicial complexes are defined under the modulo- 2 operation, the number of involved simplices is $\log \left|H^{k}\right|$, which is defined as the rank of a group, or the Betti number for $H^{k}$, denoted $\beta_{k}$, which can be efficiently calculated as:

$$
\begin{aligned}
\beta_{k} & =\operatorname{rank}\left(H^{k}\right)=\log \left|H^{k}\right|=\log \left(\left|D^{k}\right| /\left|B^{k}\right|\right) \\
& =\log \left|D^{k}\right|-\log \left|B^{k}\right|=\operatorname{rank}\left(D^{k}\right)-\operatorname{rank}\left(B^{k}\right) .
\end{aligned}
$$

Betti number implies the number of $k$-dimensional "basic" hole embedded in the topology of the MEA data; by "basic", we mean that the hole is not a composition of other holes. In as it has no direct implication to our discussion.
our MEA applications, the Betti number implies the parallelism for applying Kirchhoff's laws concurrently.

### 2.4 Parallel Processing of Multidimensional Electrode Arrays using

## Algebraic Invariant

This section presents the potential parallelism enabled by the algebraic invariant we developed in $\$ 2.3$. For completeness, we will first briefly review the baseline approach that is built upon the vertex-correlation in graph theory [89]. Then, we describe how to apply work-stealing to improve the parallelism. Finally, we show that the parallelism exhibited by algebraic invariant can be naturally leveraged by popular paradigms such as multithreading and multiprocessing across nodes.

### 2.4.1 Categorizing Vertex-oriented Constraints

Due to the existence of redundant or several possible sub-paths between distinct pair of end points, we propose a new approach that is not tightly correlated to the $n^{(n+1)}$ paths between end points. Instead, we concentrate just on the $n$ joints and attempt to translate a set of paths into a set of joints while preserving all topological features. To put it another way, our aim is to reduce the number of constraints from $O\left(n^{n}\right)$ to $O\left(n^{c}\right)^{3}$ without losing any information, i.e., lossless conversion. Such conversion is only achievable if we can somehow express the partially redundant paths as a single joint. The key idea is inspired by the observation

[^2]that many distinct end-to-end paths take the same sub-paths for a lot of times. Hence, we try to reconstruct a different/equivalent graph topology for the $n \times n$.

Figure 2.4 shows a concrete example of converting all feasible paths (i.e., 9 paths) between two end points $C$ and $I$ in Figure 2.1 where $\mathrm{n}=3$. We identify the following nine paths from $C$ to $I$ to verify that the transformed topology and the original array are equal.

1. $C \rightarrow R_{13} \rightarrow I$
2. $C \rightarrow R_{13} \rightarrow R_{23} \rightarrow R_{21} \rightarrow R_{31} \rightarrow R_{33} \rightarrow I$
3. $C \rightarrow R_{13} \rightarrow R_{33} \rightarrow R_{32} \rightarrow R_{22} \rightarrow R_{23} \rightarrow I$
4. $C \rightarrow R_{12} \rightarrow R_{22} \rightarrow R_{23} \rightarrow I$
5. $C \rightarrow R_{12} \rightarrow R_{32} \rightarrow R_{33} \rightarrow I$
6. $C \rightarrow R_{12} \rightarrow R_{22} \rightarrow R_{21} \rightarrow R_{31} \rightarrow R_{33} \rightarrow I$
7. $C \rightarrow R_{11} \rightarrow R_{31} \rightarrow R_{33} \rightarrow I$
8. $C \rightarrow R_{11} \rightarrow R_{21} \rightarrow R_{23} \rightarrow I$
9. $C \rightarrow R_{11} \rightarrow R_{31} \rightarrow R_{32} \rightarrow R_{22} \rightarrow R_{23} \rightarrow I$

For an arbitrary pair of endpoints between the $i$-th horizontal wire and the $j$-th vertical wire as shown in Figure 2.2, the equivalent topology can be expressed as Figure 2.5. In essence, the most straightforward path between $i$ and $j$ goes only through $R_{i j}$, shown as the top path (or the top main route) in the figure. Then, there are $(n-1)$ main routes starting with $R_{i k}$ where $k \in\{1, \cdots, j-1, j+1, \cdots, n\}$, corresponding to the first set of voltage values called $U a_{i j k^{\prime}}$ where $k^{\prime}=k$ if $k \leq j$ and $k^{\prime}=(k-1)$ otherwise. Similarly, toward the end of each main route, there are $(n-1) R_{m j}$ 's where $m \in\{1, \cdots, i-1, i+1, \cdots, n\}$ and $(n-1)$ $U b_{i j m^{\prime}}$ 's where $m^{\prime}=m$ if $m \leq i$ and $m^{\prime}=(m-1)$ otherwise. We do not assign a variable


Figure 2.4: A corresponding topology between $C$ and $I$ as Figure 2.1. There also exists nine paths between $C$ and $I$, which are semantically equivalent to Figure 2.1 but with possible loops across sub-paths.


Figure 2.5: A corresponding topology between the $i$-th horizontal axis and the $j$-th vertical axis in a $n \times n$ Array, in a 2D space. There are total $(n-1) U a$ 's and $(n-1) U b$ 's as two sets of distinct voltage values for a specific pair of $i j$ end points, assuming the original voltage can be measured between $i$ and $j$. The conditions of subscripts are: $k \in\{1, \cdots, j-1, j+1, \cdots, n\} ; m \in\{1, \cdots, i-1, i+1, \cdots, n\} ; k^{\prime}=k$ if $k \leq j$, $k^{\prime}=(k-1)$ otherwise; and $m^{\prime}=m$ if $m \leq i, m^{\prime}=(m-1)$ otherwise.
to the end-to-end voltage between $i$ and $j$ (i.e., $U_{i j}$ ) because it can be easily measured in practice. Both $U a$ and $U b$ have three subscripts, with $i$ and $j$ indicating the two endpoints and the third subscript indicating the top-down ordering of those voltage values from 1 to $(n-1)$. As we will see soon, this equivalent topology would yield a polynomial number of equations by enforcing the constraints on those $2(n-1)$ voltage points $U a$ 's and $U b$ 's, as opposed to an exponential number of equations as discussed before.

Given the equivalent topology, we are able to enforce the constraints on the joints $(i, j$, $U a$ 's, and $U b$ 's) instead of the paths. The saving is significant: for each pair of endpoints, there are $2 n$ joints $\left(1\right.$ at $i, 1$ at $j,(n-1)$ at $U a$ 's, and $(n-1)$ at $U b$ 's) and $n^{(n-1)}$ paths; or for the entire system, there are a polynomial number $2 n \cdot n^{2}=O\left(n^{3}\right)$ of joints and exponential number $n^{(n-1)} \cdot n^{2}=O\left(n^{n}\right)$ of paths. The following of this section explains how we generate the equations on those $2 n$ joints for a pair of endpoints $i$ and $j$. The $2 n$ equations (satisfying the circuit flow constraints by the Kirchhoff Law) for each pair of endpoints $(i, j)$ are defined as follows:

$$
\begin{cases}\frac{U_{i j}}{Z_{i j}}=\frac{U_{i j}}{R_{i j}}+\sum_{k} \frac{U_{i j}-U_{i j k^{\prime}}}{R_{i k}}, & \text { \# One equation at } i \\ \frac{U_{i j}}{Z_{i j}}=\frac{U_{i j}}{R_{i j}}+\sum_{m} \frac{U_{i j m^{\prime}}}{R_{m j}}, & \text { \# One equation at } j \\ \frac{U_{i j}-U_{i j k^{\prime}}}{R_{i k}}=\sum_{k} \frac{U_{i j k^{\prime}}-U_{i j m^{\prime}}}{R_{m k}}, & \text { \# n-1 eq.'s for } U a \\ \frac{U_{i j m^{\prime}}}{R_{m j}}=\sum_{m} \frac{U_{i j k^{\prime}}-U_{i j m^{\prime}}}{R_{m k}}, & \text { \# n-1 eq.'s for } U b\end{cases}
$$

where (i) $k^{\prime}=k$ if $k \leq j$ and $k^{\prime}=(k-1)$ otherwise; and (ii) $m^{\prime}=m$ if $m \leq i$ and $m^{\prime}=(m-1)$ otherwise. For the entire array, there are $(n-1) \cdot n^{2}$ unknown $U a$ 's, $(n-1) \cdot n^{2}$ unknown $U b$ 's, and $n^{2}$ unknown $R$ 's; all $U_{i j}$ 's and $Z_{i j}$ 's are measured values. The total number of nonlinear equations for the entire $n \times n$ array is $2 n^{3}$, with $(2 n-1) \cdot n^{2}$ unknowns. Although a system of nonlinear equations does not guarantee unique or sensible roots (for instance, resistance cannot be non-positive values), specifying a practical and positive $U_{i j}$ value usually precludes the problem, which is out of the scope of this chapter.

Obviously, all the joints $\$^{4}$ can be categorized into four groups: (i) source points with 1-

[^3]to- $n$ flow constraints; (ii) destination points with $n$-to- 1 flow constraints; (iii) intermediate points close to the source, with 1-to-n flow constraints; and (iv) intermediate points close to the destination, with $n$-to- 1 flow constraints. Each of these four types is independent of the others, thanks to the resistors in-between. Therefore, the baseline implementation for parallelization is to assign a dedicated thread to each of the aforementioned four constraint types. We will refer to this parallelization simply as parallel in the following discussion and evaluation.

In Parallel, we are restricted from having more than four threads or processes to parallelize the entire set of equations. As we will see in $\$ 2.5$, four threads will not saturate the optimization room in this case. Another limitation of Parallel is that users will have to manually split the original system of nonlinear equations into sub-systems, which might represent a technical barrier for end-users without a deep programming background.

### 2.4.2 Parallelization on MEA Manifolds

From a geometric point of view, the circuit flows in an MEA can be thought of as in a vector field of the MEA manifold, if we consider the MEA device is sufficiently "dense" or "smooth" at a local region. By vector field, we mean a function from an $n$-dimensional point $p$ to a vector $v_{p}$ eminated from $p$; by manifold, we mean an arbitrary space where each sufficiently small region is isometric to a Euclidean space. Then, it is a well-known result from differential geometry that the circuit accumulation, i.e., calculus, can be efficiently computed with the local tangent spaces (along with associated metrics such as normal vectors) and drop the global (Euclidean) coordinates. This observation has a deeper implication than it seems: Because calculus can be applied with the local parameters, which are
collectively called a frame in the literature of differential geometry, we can parallelize the computation at a finer granularity.

One advantage of adopting such a differential-geometric approach is the removal of some constraints on manufacturing MEAs. For example, our current 2-dimensional MEA device is an equidistant grid (see Figure 2.1) with orthogonal wires. With the introduction of frames, we can adopt the Jacobian matrix to covert any arbitrary MEA into a locally orthogonal frame for parallel computation on the directions of partial derivatives. That is, let $U_{i, j}$ denote the voltage value at a specific node, then elementary calculus on Euclidean space $\mathbb{R}^{n}$ tells us

$$
\frac{\partial^{2} U_{i, j}}{\partial x \partial y}=\frac{\partial^{2} U_{i, j}}{\partial y \partial x}
$$

where $x$ and $y$ represents the two orthogonal axes in $\mathbb{R}^{2}$, and in a manifold the change of $U_{i, j}$, denoted $D(U)$, can be calculated as

$$
D(U)=\left[\begin{array}{cc}
\frac{\partial U_{i}}{\partial x} & \frac{\partial U_{i}}{\partial y} \\
\frac{\partial U_{j}}{\partial x} & \frac{\partial U_{j}}{\partial y}
\end{array}\right] \cdot\left[\begin{array}{l}
d x \\
d y
\end{array}\right]
$$

which can then be plugged into the usual vector calculus and possibly calculate the voltage change along the wires by applying Stokes' theorem to the voltages:

$$
\int_{x y-\text { boundary }} U=\iint_{x y-\text { patch }} D(U) .
$$

The above discussion shows that as long as the smoothness assumption holds, we can efficiently parametrize MEAs with local voltage values in parallel. In practice, although the spatial gap among MEA nodes is not negligible, we can repeat the measurement and consider the vector of repeated measurements as a more realistic manifold. The practicality
of MEA manifolds depends on the nature of the MEA applications. That is, if the voltage change is continuous, meaning that there is no "abrupt" change exhibited in the application, then $U$ is evidently differentible and integrable. In a microelectronic setup, it is usually assumed that the voltage change is continuous [49].

In $\$ 2.4 .1$, we present a method taking a polynomial time (i.e., $O\left(n^{c}\right), c$ is a constant number, $n$ is the number of endpoints in the MEA) to parametrize MEAs at joints rather than paths. We demonstrate that $c=3$ for a two-dimensional MEA; the complexity can be trivially generalized into $O\left(n^{k+1}\right)$ for an arbitrary $k$-dimensional MEA. With the topological parallelization introduced in this section, we can further improve the asymptotic time cost by paralleling the parameterization for the homology groups, or visually speaking, the "holes". In an $k$-dimensional equidistant MEA, that means we could further improve the parallelism by $(n-1)^{k}$-fold. Therefore, the overall complexity for parametrizing a $k$-dimensional MEA could be theoretically reduced to

$$
\frac{O\left(n^{k+1}\right)}{(n-1)^{k}}=\frac{O\left(n^{k+1}\right)}{O\left(n^{k}\right)}=O(n)
$$

That is, we would be able to achieve a method linear in time for MEA parametrization as long as the device is "smooth" enough, by which we mean the fact that the MEA has sufficiently dense endpoints being concurrently worked by a sufficiently large number of processes. This will be experimentally demonstrated in the evaluation, e.g., cf. Fig. 2.9 and Fig. 2.10

### 2.4.3 System Optimization

### 2.4.3.1 Balanced Parallel

An improved parallelization can be achieved by balancing the workload through workstealing. If we closely examine the four constraint categories, two of them comprise a lot more constraints: the number of sources and destination joints is $n$, while two intermediate types are $n^{2} \cdot(n-1)$-roughly in the cubic order of the former. Therefore, in this optimization, we allow threads to continue working on other tasks instead of waiting idly. In theory, this approach could help reduce the end-to-end execution time if the overhead of switching threads is nicely controlled. We will refer to this implementation as Balanced Parallel.

It should be clear that, however, our implementation takes a deterministic approach to balance the workload rather than making the decision at runtime, which is stochastic. Determinacy, however, is a double-edged sword: it helps reduces the runtime overhead of switching threads, and yet might hurt the flexibility in practice, especially for large-scale applications. In later evaluations, we will see that Balanced Parallel achieves the highest performance at small scales and yet delivers sub-optimal performance at larger scales. If we step back and look at the big picture, Balanced Parallel described here still falls into the category of coarse-grained parallelization.

### 2.4.3.2 Fine-grained Multiprocessing (PyMP-k).

Automatic multiprocessing (e.g., OpenMP) is designed for well-structured loops, which is, unfortunately, not the case in electrode arrays. First, the four constraint types cannot
be programmatically expressed in the same loop. Second, the electrode data are highly skewed with two hefty tasks compared to others.

To leverage the OpenMP-like parallelization, we implement the Betti-number-aware multiprocessing approach by pushing the parallelization into each of the $k$-dimensional loops. That is, in addition to the parallelization between constraint types, we now enable the intratype parallelism regardless of the constraint type. The downside is, however, for small $n$ of lightweight constraints, the efficiency might be low due to the small workload (compared to the overhead). If the dominant workloads are at large scales, the performance gain might outweigh the low efficiency from lightweight workloads. As we will see this in the evaluation section, Parma incorporated with an OpenMP-like library, PyMP ${ }^{5}$, taking the aforementioned approach delivers the highest performance at large scales up to $100 \times 100$ arrays. PyMP utilizes its work-sharing constructs to enable load balancing among processes. Constructs take an amount of work and distribute it over the specified number of processes in a parallel region.

The above approach can be extended to multiple nodes, e.g., being implemented with MPI. In general, the overhead across nodes (e.g., I/O cost of message passing) is higher than the parallelization within a physical node. Therefore, inter-node parallelization is preferable only when the workload share per process is significantly higher than the amortized overhead. We will quantify the workload impact to the performance of different scales (up to 1,024 processes) in the evaluation section.

[^4]
### 2.5 Implementation and Evaluation

Our evaluation focuses on three metrics: the computation time ( $\$ 2.5 .3$ ), the memory footprint ( $\$ 2.5 .4$ ), the I/O cost ( $\$ 2.5 .5$ ), and the scalability ( $\$ 2.5 .6$. Three baseline systems are used when applicable: (i) Single-thread: the serialized implementation of MEA analysis as in the literature [89], (ii) Parallel: the naive parallel processing based on vertexcorrelation [121], and (iii) Balanced Parallel: a work-stealing approach based on Parallel that we discuss in this section ( $\$ 2.4$ ).

### 2.5.1 Implementation

We have implemented the proposed parallelization methods with Python v3.7.0, PyMP v0.4.2. Our whole framework is implemented in Python because the state-of-the-art system [89] upon which ours is built was implemented in Python. There are about 2,600 lines of Python code and other scripts (BASH, R, etc.) in our current implementation, which can be downloaded from the project online repository ${ }^{6}$

The current implementation comprises two main parts:

- MEA: This component converts the original exponential all-pair-path problems into polynomial ones.
- Parma: This component applies various optimizations to parallelize the formation of the system of nonlinear equations.

[^5]We have evaluated the system prototype on up to $100 \times 100$ arrays or end points. The electrode array hardware comprised $64 \times 64$ wires built in the wet lab of our collaborators from the Department of Biomedical Engineering. The environment can be conveniently set up using popular Python frameworks such as Anaconda.

### 2.5.2 Experimental Setup

Our test bed consists of an on-premises system comprised of a many-core server i.e., HP Z820 server and a high-performance computing (HPC) cluster:

1. The HP Z820 server has 32 Intel Xeon E5-2670 cores, 128 GB RAM, a 500 GB SSD, and a 2 TB HDD; and
2. The high-performance computing (HPC) cluster is comprised of 58 nodes interconnected with FDR InfiniBand. Each node is equipped with an Intel Core-i7 2.6 GHz 32-core CPU along with 296 GB 2400 MHz DDR4 memory. It has a remote 2.1 PB storage system managed by GPFS [103]. We use up to 32 nodes, or 1,024 cores, in the following experiments.

All test beds are installed with Ubuntu 16.04, Python 3.7.0, NumPy 1.15.4, SciPy 0.17.0, PyMP v0.4.2, mpi4py v2.0.0, and mpich2 v1.4.1.The performance results we have obtained and illustrated graphically are an average of multiple trials.

All of the experimental data (up to $100 \times 100$ ) are obtained from a microelectrode array device measuring (unknown) numbers of cells atop their media at a wet lab from the Department of Biomedical Engineering. The data are originally saved as Excel files and


Figure 2.6: Various approaches for parallel formulation of Kirchhoff law equations. The proposed joint constraints enable various parallelization possibilities, namely Parallel, Balanced Parallel, and PyMP. PyMP delivers the highest performance at scales $n \geq 20$, despite of lower performance than Balanced Parallel at $n=10$ where the parallelization overhead outweighs the speedup.
converted into text files before being fed to the Parma system prototype. The data at the wet lab are measured four times a day: 0 hour, 6 hour, 12 hour, and 24 hour, after the device setup is completed. The resistance values of cells range between 2,000 and 11,000 Kilohm, while the electrical voltage is 5 volts.

### 2.5.3 Computation Time

In Figure 2.6, we report the performance of three parallelization optimizations applied to Parma. The experiments were carried out on the on-premises system. PyMP delivers the highest performance at scales $n \geq 20$, despite of lower performance than Balanced Parallel at $n=10$ where the parallelization overhead outweighs the speedup. Since PyMP seems to perform best at larger scales, which is not surprising as it offers fine-grained parallelism, the remainder of this subsection will further investigate the properties of PyMP in more detail unless otherwise noted.

In addition, we report the overall compute time at various levels of parallelism $k \in$


Figure 2.7: Computation time of various parallelism in PyMP. Applying fine-grained multiprocessing leads to a linear decrease in the overall compute time per workload at scales $n \geq 20$.
$\{2, \cdots, 32\}$ in PyMP without the I/O time in Figure 2.7. The experiments were carried out on the HPC cluster. It can be observed that the improvement in performance or speedup becomes more significant at scales $n \geq 20$ for the various levels of parallelism $k \in\{2, \cdots, 32\}$ in PyMP despite of the inconsistent performance when $n=10$.

### 2.5.4 Memory Footprint

We report the memory characterization at various scales, as reported in Figure 2.8. For all the scales of $n \in\{10 . .100\}$, the peak memory usage is about the same regardless of data parallelism. However, a higher parallelism on large scales ( $n \geq 40$ ) implies a higher utilization of the memory: for instance, two threads $(k=2)$ on a $100 \times 100$ array $(n=100)$ incur a low memory footprint in about $60 \%$ of time while four threads $(k=4)$ incur the same memory footprint only in about $30 \%$ of time. At small scales ( $n \leq 20$ ), little difference is observed. The memory usage is proportional to the rank of $n$ and is under 20 GB for a $100 \times 100$ array .

This experiment shows that there is negligible memory overhead while improving temporal performance due to the spawning of new processes for any selected number of endpoints or scales. For each selected scale, the peak memory usage for a different number of threads remains almost the same.


Figure 2.8: Cummulative Distribution Functions (CDFs) of Memory Usage. For all the scales of $n \in$ $\{10 . .100\}$, the peak memory usage is about the same regardless of data parallelism.


Figure 2.9: The end-to-end time of various degrees of parallelism in PyMP, including disk I/Os. Utilization of more threads $k \geq 2$ starting from a low rank of $n=20$ makes significant effect to the overall I/O time.

### 2.5.5 I/O Cost

We report the overall time taken to generate the set of equations and write them to a file in disk with Parma. The experiments were carried out on the HPC cluster. Figure 2.9 shows the results at up to $n=100$. In comparison with results reported in Figure 2.7, the time taken to write the set of equations to disk exhibit noticeable differences at scales $n \geq 20$ for threads at various levels of parallelism. The results confirm our conjecture that spawning more threads is preferable for larger workloads such that the overhead can be amortized.

### 2.5.6 Scalability

We report the scalability of Parma in terms of spawning more processes as reported in Figure 2.10. Due to a maximum number of 32 physical cores on a single server, we have implemented the topological parallelism with MPI. We deploy the MPI implementation on up to 1,024 cores, and observe a linear strong scalability for practical workloads


Figure 2.10: Scalability of Parma across various number of processes and varying workloads.
(e.g., $50 \times 50$ or larger MEAs). For smaller workloads (e.g., $10 \times 10$ and $20 \times 20 \mathrm{MEAs}$ ), the inter-node parallelism is not effective and an intra-node parallelization (e.g., OpenMP) is recommended.

### 2.6 Related Work

Loke et al. [77] proposed techniques for the fast computation of electrode arrays for twodimensional (2D) resistivity surveys. An automatic graph-based method [130] was proposed for localizing distantly-spaced cochlear implant electrode arrays in clinical computed tomography with sub-voxel accuracy. In [40], a method based on the concept of Space-Amplitude Transform was proposed to transform time recordings from a 2D electrode array as a one-dimensional (1D) plus time signals in order to speed up and make simpler the data analysis. Kiele et al. presented the principles for a robust and precise
alignment monitoring system, which allows the detection of linear and rotational displacements of two parallel electrode arrays [67]. In [33], finite element method (FEM) modeling was proposed for studying the impact of simultaneous impedance measurement of 100 electrodes of a Utah Electrode Array (UEA). Yassin et al. [127] proposed an energy-efficient spike data extraction solution for a high-density electrode array capable of reducing the data to be transferred by over $85 \%$. Buccino et al. [21] proposed a semi-automatic approach involving an online implementation of the Independent Component Analysis (ICA) algorithm for real-time spike sorting of high-density Multi-Electrode Array data. Also, a method to automate spike sorting in electrical stimulation experiments using large multielectrode arrays, where artifacts are a concern, was proposed in [80].

While aforementioned literature proposed approaches to alleviate existing challenges encountered with the utilization of electrode arrays in various scenarios, this work instead, for the first time, focuses on a new approach to transform the original problem from spatial domain to temporal domain and enables unprecedented parallelization possibilities.

### 2.7 Summary

This chapter addresses the long-existing computational challenge of multidimensional data in one of the most widely used engineering devices, namely microelectronic array (MEA). We propose a new algebraic model to abstract the entities in MEA; the new model then allows us to develop new methodology to parallelize the computation dictated by the Kirchhoff law. We implement a system prototype-namely Parma-with various optimizations backed by the proposed algebraic model and parallelization, and evaluate its performance on up to 1,024 cores. Experimental results show that the proposed approach significantly
outperforms the state-of-the-practice: the computation time is orders of magnitude faster; the $\mathrm{I} / \mathrm{O}$ cost is proportionally reduced; and the memory is efficiently utilized.

## CHAPTER 3

## TOWARD EFFICIENT HOMOMORPHIC ENCRYPTION FOR OUTSOURCED DATABASES THROUGH PARALLEL CACHING

### 3.1 Introduction

### 3.1.1 Background and Motivation

While increasingly more applications are deployed on the public clouds, one of the biggest challenges lies in confidentiality, especially for those applications that usually touch on sensitive data in the fields such as public health [62], bioinformatics [132], and financial services [58]. Although various encryption schemes (e.g., AES [87], RSA [99]) can be applied before the data are sent to the cloud, it would defeat the purpose of cloud computing if the users must download and decrypt the encrypted data for processing: the cloud in this case works merely as remote storage with no computing functionalities. One plausible solution to the above confidentiality problem is adopting specific encryption schemes such that the ciphertexts stored on the cloud can perform certain computations, which are known as homomorphic encryption (HE). Although most HE schemes support only primitive arithmetic operations such as addition and multiplication, it turns out that many commonly-used operations (e.g., comparison) can be constructed on top of circuits of additions and multiplications [43]. However, a scheme supporting both addition and multiplication over ciphertexts, namely fully homomorphic encryption (FHE), usually incurs a much higher performance overhead than (partial) HE, or PHE schemes by orders of magnitude. These

PHE schemes can be categorized into two types depending on how the key is distributed.

The first type of PHE schemes, e.g., Symmetria [102], is implemented as a symmetric operation for the scenarios where a secret key can be securely shared among parties. In order to ensure high security, Symmetria introduces a randomization component in the ciphertext that keeps growing, which might cause significant performance overhead. Seabed [92] is another symmetric PHE cryptosystem but only supports primitive additions (e.g., no subtraction or negation).

The second type of PHE scheme, e.g., Paillier [91], is implemented as an asymmetric operation with a pair of public and private keys. An asymmetric scheme employs hard mathematical problems in number theory and group theory to safely distribute the public keys, rendering it orders of magnitude slower than a symmetric scheme. Although a hybrid scheme can be used with symmetric key for encryption and asymmetric operation for key distribution, key distribution is needed per session in database-as-a-service (DaaS), implying that asymmetric operations would be invoked routinely.

Although PHE is much more efficient than FHE, PHE still cannot meet the performance requirements for data-intensive workloads in DaaS. As we will show later in this chapter (§2.5), the state-of-the-art PHE scheme, Symmetria [102], can only encrypt data at a rate of 3 Mbps -much lower than the commodity network bandwidth (cf. Fig. 3.11) that is in the order of tens of Mbps or even Gbps. That being said, the performance bottleneck of data-intensive applications, such as video analysis [31, 52], lies at the encryption subsystem.

Our long-term goal is to improve the performance of homomorphic encryption applied to large volumes of outsourced data; this chapter attains the above goal, as the first step, by
proposing a new caching approach to reduce the computational overhead in both symmetricand asymmetric-PHE schemes for outsourced databases or DaaS in cloud computing. It is our hope that data-intensive applications would better exploit the high security and low overhead of PHE schemes by incorporating the proposed technique.

### 3.1.2 Contributions

The key insights of our proposed caching technique include: (i) precomputing and caching some homomorphic ciphertexts before encrypting the large volume of plaintexts; (ii) expanding a requested plaintext into a summation of additive radix entries; (iii) constructing the ciphertexts with randomized homomorphic addition, without touching on encryption primitives; and (iv) enabling incremental encryption based on the extended entries of the cached ciphertexts.

Formally, this chapter discusses the following technical contributions:

- Firstly, we propose an algorithm to reconstruct the ciphertext using radixes in the context of homomorphic encryption (HE). We name the new algorithm radix homomorphic encryption, or RHE. We conduct a thorough analysis of parametrization for RHE. (§3.3)
- Secondly, we design a full-fledged protocol called Radix-additive caching for homomorphic encryption (Rache), which adopts RHE to securely encrypt a large volume of data. We articulate the security goal, threat model, and security assumptions, under which the RHE protocol is proven secure. (\$3.4)
- Thirdly, we extend Rache into an incremental protocol that allows for efficient homomorphic encryption of data streams. We also demonstrate the provable security of this incremental protocol. ( $\$ 3.5$ )


### 3.2 Preliminaries and Related Work

### 3.2.1 Confidentiality of Outsourced Data

We review four important techniques to ensure the confidentiality of outsourced data: encrypted storage, encrypted tuples, encrypted fields, and secure multi-party computation.

Encrypted Storage. The database instance from the cloud vendor is considered as storage of encrypted data and the client is responsible for nontrivial queries. This solution is viable only if (i) the relations touched on by the query are small enough that the network overhead of transmitting those relations is acceptable, and (ii) the user has the capability (both computation and storage) to execute the query locally. We stress that this solution might defeat the purpose of outsourcing the database service to the cloud.

Encrypted Tuples. Every tuple of the original relation $R$ is encrypted into a ciphertext that is stored in column $T$ of a new relation $R^{s}$. For each attribute $A_{i}$ in $R$, there is a corresponding attribute $A_{i}^{s}$ in $R^{s}$, whose value is the index of $R . A_{i}$. The index is usually assigned by a random integer based on some partitioning criteria and can be retrieved with the metadata stored on the client, i.e., the user's local node. As a result, the schema stored at the cloud provider is $R^{s}\left(T, A_{1}^{s}, \ldots, A_{i}^{s}, \ldots\right)$. When the user submits a query $Q$, the client splits
$Q$ into two subqueries $Q_{s}$ and $Q_{c} . Q_{s}$ serves as a filter to eliminate those unqualified tuples based on the indices in $R^{s}$ and transmits the qualified tuples (in ciphertexts) to the client. $Q_{c}$ then ensures that those false-positive tuples are eliminated after the encrypted tuples are decrypted using the secret key presumably stored on the client. This approach involves both the client (i.e., the user) and the server (i.e., the cloud provider) when completing a query, often referred to as information hiding approaches [47].

Encrypted Fields. The third approach aims to minimize the involvement of clients when processing the query over the encrypted data stored at the cloud vendor. The idea is to encrypt the relations at a finer granularity-each attribute of a relation is separately encrypted. The key challenge of this approach lies in its expressiveness, e.g., how to apply arithmetic or string operations over the encrypted fields. While fully homomorphic encryption (FHE) [43] can support a large set of computing problems, the performance of current FHE implementations cannot meet the requirements of practical database systems [93, 94]. An alternative solution is partially homomorphic encryption (PHE) schemes [39, 91], which are orders of magnitude faster than FHE but only support a single algebraic operation. Traditional PHE schemes are designed for public-key (asymmetric) encryption, which is desirable for straightforward key distribution over insecure channels but significantly more expensive than secret-key (symmetric) encryption. However, in the context of DaaS, the user usually serves as both the sender and the receiver and there is no need to distribute the key. To this end, symmetric (partially) homomorphic encryption (SHE), was proposed [92,102].

Secure Multi-Party Computation (MPC). In addition to HE-based methods, another widely-used technique for data privacy is secure multi-party computation (MPC), which originated from [126] and has been mostly built upon oblivious transfer [45, 64], threshold homomorphic encryption [28, 29], and secret sharing [96, 106]. MPC has been applied in multiple machine learning frameworks, such as DeepSecure [100], SecureML [84], and

ABY [34].

### 3.2.2 Homomorphic Encryption

The term homomorphic or homomorphism originates from group theory, which depicts such a function that can be applied either before or after the operations conducted in the domain or the image. Formally, we have the following mathematical definition.

Definition 1 (Homomorphism). Given two groups $(F, \oplus)$ and $(G, \otimes)$, a function $h: F \rightarrow G$ is called a homomorphism if $h\left(f_{1} \oplus f_{2}\right)=h\left(f_{1}\right) \otimes h\left(f_{2}\right), \forall f_{1}, f_{2} \in F$.

There are many examples of homomorphism. The following is a simple one we have seen in basic mathematics.

Example 1. We can define two groups $F=(\mathbb{R},+)$ and $G=\left(\mathbb{R}^{+}, \times\right)$with the regular arithmetic operations, where $\mathbb{R}$ and $\mathbb{R}^{+}$denotes real numbers and positive real numbers, respectively. Moreover, we define a function $h(x)=2^{x}$, where $x \in \mathbb{R}$. Evidently, the following equation holds: $h(a+b)=2^{a+b}=2^{a} \times 2^{b}=h(a) \times h(b)$.

Homomorphic encryption (HE) is a specific type of encryption where certain operations between operands can be performed directly on the ciphertexts in the sense that the result can be decrypted into the same value as if the operations were applied to the plaintexts. If we connect HE to the group-theoretical definition of homomorphism, the encryption function can be thought of the homomorphism, the set of plaintexts as the domain of the homomorphism, and the set of ciphertexts as the image of the homomorphism.

An HE scheme that supports the arithmetic addition over the ciphertexts is called additive. That is to say, we can define an addition operation $\oplus$ between two ciphertexts, say enc $(x)$ and enc(y) encrypted by function $\operatorname{enc}(\cdot)$, such that

$$
\begin{equation*}
\operatorname{dec}(e n c(x) \oplus e n c(y))=x+y \tag{3.1}
\end{equation*}
$$

where $\operatorname{dec}(\cdot)$ denotes the decryption function corresponding to $\operatorname{enc}(\cdot)$. It should be noted that Eq. (3.1) does not necessarily imply a mathematical homomorphism as defined in Def. 1] that is, we generally do not require $\operatorname{enc}(x) \oplus \operatorname{enc}(y)=\operatorname{enc}(x+y)$. This is more of a practical security consideration rather than a mathematical one: randomness is always required for cryptographic schemes in practice (e.g., to defeat chosen-plaintext cryptanalysis), and therefore, repeated encryption of the same plaintext should look different, i.e., random.

Many encryption schemes in the literature are homomorphic, such as Symmetria [102] and Paillier [91]. Symmetria is a symmetric encryption scheme, meaning that a single secret key is used to both encrypt and decrypt the messages. By contrast, Paillier is asymmetric, where a pair of public and private keys are used for encryption and decryption, respectively. Due to the expensive arithmetical operations performed by the asymmetric encryption, Paillier is orders of magnitude slower than Symmetria. However, Paillier is particularly useful when there is no secure channel to share the secret key among parties.

An HE scheme that supports multiplication is called multiplicative. Symmetria [102] is also multiplicative using a distinct scheme than the one for addition. Other well-known multiplicative HE schemes include RSA [99] and ElGamal [39]. A multiplicative HE
scheme ensures the following equality,

$$
\operatorname{dec}(e n c(a) \otimes e n c(b))=a \times b,
$$

where $\otimes$ denotes the multiplication defined over ciphertexts.

An HE scheme that supports both addition and multiplication is called a fully HE (FHE) scheme. This requirement should not be confused with specific addition and multiplication parameters, such as Symmetria [102] and NTRU [57]. That is, the addition and multiplication must be supported homomorphically under exactly the same scheme:

$$
\left\{\begin{array}{l}
\operatorname{dec}(e n c(a) \oplus e n c(b))=a+b \\
\operatorname{dec}(e n c(a) \otimes e n c(b))=a \times b
\end{array}\right.
$$

It turned out to be extremely hard to construct FHE schemes until Gentry [43] demonstrated that such a scheme can be constructed using lattice theory. Indeed, multiple implementations are available today, such as BGV [48], BFV [41], and CKKS [26]. Nonetheless, the performance overhead of FHE implementations still cannot meet the requirement of many real-world applications, especially those data-intensive applications. Two popular opensource libraries of FHE schemes are IBM HElib [54] and Microsoft SEAL [105]. Some more recent implementations are optimized for machine learning and vector computation, such as TenSEAL [15].

A lot of research efforts have been put to optimize the performance of HE schemes. For instance, hardware-based optimization [36, 98, 101] has been heavily exploited. A recent article argues that the current performance bottleneck of HE lies in the memory wall [32]. The notion of incremental cryptography was first formalized in 1990s [13, 14], mainly
from a theoretical perspective. More recent work on incremental encryption schemes can be found in [8, 66, 82]. Incremental encryption recently draws a lot of research interests for efficient data encoding in the resource-constraint contexts such as mobile computing [18, 63, 119].

### 3.2.3 Provable Security

When employing an encryption scheme in an application, it is highly desirable to demonstrate its security in a provable manner. Formally, we need to clearly identify the following three important pieces for provable security of a given encryption scheme: security goal, threat model, and assumptions. The security goal spells out the desired effect when the application is under attack; the threat model articulates what an adversary can do with the attack, such as what information of the plaintext/ciphertext can be collected and the resource/time limitation of the attack; the assumption lists the presumed specifics of the subsystems or components of the cryptographic scheme, which is usually an important building block for the security proof, e.g., reduction. The security goal and threat model are usually called security definition collectively.

One well-accepted security definition with a good balance between efficiency and security is that the adversary is able to launch a chosen-plaintext attack (CPA), defined as follows.

Definition 2 (Chosen-Plaintext Attack). Given a security parameter n, i.e., the bitstring length of the key, an adversary can obtain up to poly(n) of plaintext-ciphertext pairs ( $m, c$ ), where $m$ is arbitrarily chosen by the adversary and poly $(\cdot)$ is a polynomial function on $n$. With such information, the adversary tries to decrypt a $c^{\prime}$ that is not included in the polynomial number of known ciphertexts.

The polynomial requirement is only for practical reasons, as we usually assume that the adversary should only be able to run a polynomial algorithm without unlimited resources. Accordingly, we want to design encryption schemes that are CPA secure: even if the adversary $\mathcal{A}$ can obtain those extra pieces of information, $\mathcal{A}$ should not be able to decode the ciphertext better than a random guess up to a very small probability. To quantify the degree of this small probability, negligible function is defined as below.

Definition 3. A function $\mu(\cdot)$ is called negligible iffor all polynomials poly(n) the inequality $\mu(n)<\frac{1}{\text { poly }(n)}$ holds for sufficiently large n's.

For completeness, we list the following lemmas for negligible functions that will be used in later sections. We state them without the proofs, which can be found in introductory cryptography or complexity theory texts.

Lemma 1 (Summation of two negligible functions is a negligible function). Let $\mu_{1}(n)$ and $\mu_{2}(n)$ be both negligible functions. Then $\mu(n)$ is a negligible function that is defined as $\mu(n) \stackrel{\operatorname{def}}{=} \mu_{1}(n)+\mu_{2}(n)$.

Lemma 2 (Quotient of a polynomial function over an exponential function is a negligible function). $\frac{\text { poly }(n)}{2^{n}}$ is a negligible function. That is, $\exists N \in \mathbb{N}, \forall n \geq N: \frac{p o l y(n)}{2^{n}}<\frac{1}{\text { poly }(n)}$, where $\mathbb{N}$ denotes natural numbers.

### 3.3 RHE: Radix Homomorphic Encryption

### 3.3.1 Overview

Our key observation is that although a HE encryption operation is costly, the algebraic operation over the ciphertexts is comparatively cheaper. While the concrete performance gap is dependent on how a specific HE scheme is implemented and to which data the scheme is applied, we exemplify such gaps in our experiments: Figure 3.2 in \$3.6.4.1 shows that the addition of two ciphertexts takes less than $1 \%$ time than the encryption of a plaintext in Paillier [91]. With that said, if we convert the expensive encryption operation of a given plaintext into an equivalent set of algebraic operations over existing (i.e., cached) ciphertexts, we may obtain a performance edge. There are two questions, however, in this idea.

First, which ciphertexts should we cache? Evidently, we can always cache only he(1) and then compute $h e(m)$ of $n$-bit plaintext $m$ with $\oplus_{i=1}^{m} h e(1)$. However, the accumulative overhead caused by a lot of homomorphic additions would at some point outweigh the encryption cost due to $O\left(2^{n}\right)$ additions. We propose to only cache a set of selective ciphertexts; specifically, let $r$ be a radix (and we will show how to pick $r$ in $\S 3.3 .3$ ), then the ciphertexts of $r$-power series will be pre-computed: $h e\left(r^{i}\right)$, where $r^{i} \leq 2^{n}$. By doing so, the target ciphertext will be constructed through $O(n)$ additions. It should be noted that the target ciphertext at this point is merely a deterministic ciphertext with no security.

Second, how to ensure the randomness of the ciphertext? Randomness must be added to the ciphertext to achieve a practical security level, e.g., anti- chosen-plaintext attack (CPA).

Informally, the randomness must be probabilistic small, which usually takes the form of picking a piece of data out of an exponential space. From the above discussion, we have $n$ cached ciphertexts; we will use these ciphertexts as ingredients to add a random $h e(0)$ to the deterministic ciphertext. The random $h e(0)$ is constructed by working on every radixpower $r^{i}$ : randomly adding radix-power $h e\left(r^{i}\right)$ and if so, then subtracting $r$ times of $h e\left(r^{i-1}\right)$. Overall, there are $O(r n)$ homomorphic additions that will result in he $(0)$, which is randomly selected from an exponential space $O\left(2^{n}\right)$. The above radix-wise homomorphic additions can be parallelized with the many-core architecture in modern CPUs. Before formalizing the algorithm, we illustrate the idea of Rache in an oversimplified scenario Example 2.

Example 2. Let's try to encrypt number 100 using the Rache encryption scheme. For the sake of simplicity, let $r=2, \operatorname{Ctxt}[]$ be the list of cached $r$-power ciphertexts, and $\oplus$ be the addition on the ciphertexts. Obviously, $100=64+32+4=r^{6}+r^{5}+r^{2}$. Therefore, $\operatorname{Rache}(100)=\operatorname{Rache}\left(r^{6}\right) \oplus \operatorname{Rache}\left(r^{5}\right) \oplus \operatorname{Rache}\left(r^{2}\right)=\operatorname{Ctxt}[6] \oplus \operatorname{Ctxt}[5] \oplus \operatorname{Ctxt}[2]$. That is, instead of calculating Rache(100) using sophisticated number-theoretical rules, we can simply construct Rache(100) through two homomorphic additions of cached ciphertexts, which are much simpler and faster.

### 3.3.2 Algorithm

Algorithm 1 formalizes the radix-based procedure. Let $n$ denote the security parameter of the underlying PHE scheme, i.e., the bitstring length of the key $k$ that is usually generated by $k \leftarrow \operatorname{Gen}\left(1^{n}\right)$, where $\operatorname{Gen}()$ is a pseudorandom generator. For the sake of clarity, we assume that the original plaintext value can be converted into a bitstring of length $n$ or smaller; this should not be a technical limitation in practice, as we can always split a large value into multiple blocks of $n$-bits, each of which is encrypted with randomization. In

```
Algorithm 1: RHE: Radix Homomorphic Encryption
    Input: An array of plaintexts Ptxt[], each being a padded \(n\)-bitstring; A
            homomorphic encryption function he(•) s.t.
            \(\forall a_{i} \in \operatorname{Ptxt}[], \bigoplus_{i} h e\left(a_{i}\right)=h e\left(\sum_{i} a_{i}\right) ;\) Radix \(r ;\)
    Output: An array of ciphertexts \(C t x t[]\) such that \(\forall i, h e^{-1}(C t x t[i])==\operatorname{Ptxt}[i]\),
            where \(h e^{-1}\) denotes the decryption function;
    // Initialization
    \(m:=2^{n}-1\)
    for \(i=0 ; i<=\left\lfloor\log _{r} m\right\rfloor ; i++\mathbf{d o}\)
        radixes \([i]:=\) he \(\left(r^{i}\right)\)
    end
    radixes \(\left\lfloor\left\lfloor\log _{r} m\right\rfloor+1\right]:=h e(0)\)
    // Encoding
    for \(i=0 ; i ;\) Ptxt.size() \() i++\) do
        for \(j=0 ; j<=\left\lfloor\log _{r} m\right\rfloor ; j++\mathbf{d o}\)
        \(i d x[j]:=\left(\operatorname{Ptxt}[i] / r^{j}\right) \% r\)
        end
        \(/ / \operatorname{Ptxt}[i]=\sum_{j} i d x[j] \times r^{j}\)
        Ctxt \([i]:=\bigoplus_{k=0}^{\left\lfloor\log _{r} m\right\rfloor} \bigoplus_{j=1}^{i d x[k]}\) radixes \([k]\)
        // Randomization
        isSwap \(\stackrel{\$}{\leftarrow}\{0,1\}\)
        if \(1==\) isSwap then
            \(\operatorname{Ctxt}[i]:=\operatorname{Ctxt}[i] \oplus\) radixes \(\left[\left\lfloor\log _{r} m\right\rfloor+1\right]\)
        end
        for \(j=1 ; j<\left\lfloor\log _{r} m\right\rfloor ; j++\mathbf{d o}\)
            isSwap \(\stackrel{\$}{\leftarrow}\{0,1\}\)
            if \(1==\) isSwap then
                Ctxt \([i]:=C t x t[i] \oplus \operatorname{radixes}[j]\)
            for \(k=0 ; k<r ; k++\mathbf{d o}\)
                \(\operatorname{Ctxt}[i]:=\operatorname{Ctxt}[i] \ominus \operatorname{radixes}[j-1]\)
            end
        end
    end
    end
```

other words, we construct a block cipher using Algorithm 1. If there are identical blocks, the security is nonetheless guaranteed because Algorithm 1 is randomized (Lines 11 and 16).

Lines $1-5$ initialize the reused entries of the integral powers of radix $r$ for future construction of ciphertexts. Specifically, Line 5 precomputes the homomorphic encryption of plaintext 0 , which will be used for the base case during the randomization (Lines 11-14). Lines 6-24 encode the plaintexts, each of which is computed directly over the encoded radixes that are initialized at the beginning of the protocol. For each plaintext, Lines 11-14 randomize the radix summation of ciphertexts such that repeated plaintexts will result in distinct ciphertexts. The idea of the randomization is to iterate every precomputed ciphertext radixes $[i]$ and randomly add it to the ciphertext; if the addition happens, we subtract ciphertext radixes $[i-1]$ repeatedly $r$ times.

The correctness of Algorithm $\mathbb{1}$ can be verified by straightforward algebraic computation. We skip the full computation here due to space constraints.

### 3.3.3 Parameterization

### 3.3.3.1 Heuristic Radix Selection

This section will discuss heuristic methods to decide the radix value $r$ in practice. The discussion will remain mostly informal as there are unlimited factors in real-world applications; a more rigorous approach to be presented in the next section ( $\S 3.3 .3 .2$ focuses on the worst-case scenario, where we can make more assumptions of the factors that allow us
to conduct a more quantitative analysis.

In practice, the initialization cost can be thought of a constant cost because it can be amortized by a large number of follow-up computations. As a result, the key trade-off lies at the cost of $\oplus$ 's and that of encrypting the plaintext message $m$. Let $g$ denote the ratio of computational costs of ciphertext addition over homomorphic encryption:

$$
g \stackrel{\operatorname{def}}{=} \frac{\operatorname{Time}(\operatorname{Ctxt}[i] \oplus \operatorname{Ctxt}[j])}{\operatorname{Time}(\operatorname{Rache}(m))}
$$

where Time() denotes the time function and Ctxt[] denotes the list of cached ciphertexts. Evidently, the bottom line is to ensure the average cost of $\sum_{k \in K} C t x t[k]$ for a requested ciphertext is lower than that of Rache $(m)$, or $g|K|<1$, because otherwise there is no performance improvement from caching the ciphertext. In a specific HE scheme, $g$ can be estimated using some benchmarks; for example, Figure 3.2 shows that ciphertext addition is two orders of magnitude faster than encryption in Paillier: $g=0.01$. This implies that, on average, $|K|$ should be smaller than 100. With radix $r$, the maximal possible upper bound would be $r^{100}$. Therefore, we need to pick $r$ to ensure that the maximal value of the plaintext set is smaller than $r^{100}$ in Paillier. If $M$ is the maximal message, then we require $M<r^{100}$ or $r>M^{\frac{1}{100}}$. If the plaintext space is a set of 256-bit strings, then $M=2^{256}$ and $r>\left(2^{256}\right)^{\frac{1}{100}}>2^{2.56} \approx 5.9$. Therefore, $r$ can be set to 6 .

### 3.3.3.2 Optimal Radix in the Worst Case

This section will investigate the optimal radix in the worst case. Let $m \geq 2$ denote the maximal value to be encrypted in the application. Let $r \geq 2$ denote the radix or base of the homomorphic encryption. Obviously, given an arbitrary number $x$, where $0 \leq x \leq m$,
there are $k+1$ radix entries: $r^{0}, r^{1}, \ldots, r^{k}$, where $k=\left\lfloor\log _{r}^{m}\right\rfloor$. Let $0 \leq \kappa \leq k$. In the worst case, each $r^{k}$ radix-entry incurs $r-2$ times of homomorphic addition, i.e., when computing $(r-1) \cdot x^{k}$. Since one more homomorphic addition needs to be taken for the summation of each radix, the overall times of homomorphic addition, in the worst case when $m$ is one less than the next integral power of $r$ (i.e., $\left\lfloor\log _{r}^{m}\right\rfloor=\log _{r}^{m+1}-1$ ), is

$$
f(r)=(r-2)(k+1)+k=(r-1) \log _{r}^{m+1}-1 .
$$

Our goal is therefore to find out the optimal $r$ that minimizes $f(r)$. This can be achieved by calculating the first-order and second-order derivatives of $f(r)$. We skip the detailed computation here for the sake of space; the following elementary calculus and algebra sketch the procedure to derive that $r=2$ leads to the minimum number of homomorphic additions in the worst case.

$$
f^{\prime}(r)=\frac{d}{d r} f(r)=\ln (m+1) \cdot(\ln r)^{-2} \cdot r^{-1} \cdot(r \ln r-r+1) .
$$

The stationary point is therefore the solution to $g(r)=f^{\prime}(r)=0$, which yields $r=1$. Since we require $r \geq 2$, we need to find another qualified radix. First, we calculate $g(2)$ :

$$
g(2)=2 \ln 2-2+1 \geq 2 \times 0.69-1>0 .
$$

Then, let $r \geq 3$, therefore $\ln r>1$, which yields:

$$
\left.g(r)\right|_{r \geq 3}=r \ln r-r+1=r(\ln r-1)+1>0 .
$$

Note that by definition, the following equation holds:

$$
f^{\prime}(r)=\ln (m+1) \cdot(\ln r)^{-2} \cdot r^{-1} \cdot g(r)
$$

If we assume $m \geq 2$, then $\ln (m+1)>0$. Both $(\ln r)^{-2}$ and $r^{-1}$ factors are obviously positive. Therefore, $f^{\prime}(r)$ is always positive, meaning that $f(r)$ is a monotonically increasing function. It follows that the minimal qualified radix $r=2$ leads to the minimum number of homomorphic additions.

### 3.4 Rache: Radix-Additive Caching for Homomorphic Encryption

### 3.4.1 Security Definitions and Assumptions

The security goal of our target outsourced databases is computational secrecy, which implies that any adversary cannot differentiate between the encrypted data and a random string with a probability significantly larger than $50 \%$, coined as indistinguishability. This means that when an adversary is given a ciphertext, he or she cannot do much better than randomly guessing the corresponding plaintext with reasonable resources. Technically, the degree of closeness is quantified by a negligible function; we refer readers to $\$ 3.2 .3$ for more technical details. Indeed, if we want to be strict on the $50 \%$ requirement, then it is called perfect secrecy (information-theoretical secrecy), which is beyond the scope of this chapter.

In the context of computational secrecy, we assume that the adversary cannot obtain unlimited computing resources and can only run probabilistic polynomial-time (PPT) algo-
rithms. We also assume that the adversary can launch a chosen-plaintext attack (CPA), meaning that the adversary can obtain $\operatorname{poly}(n)$ arbitrary pairs of (plaintext, ciphertext), where $n$ denotes the security parameter and $\operatorname{poly}(\cdot)$ denotes a polynomial function. We call a scheme IND-CPA if it exhibits indistinguishability under CPA.

Finally, we assume the primitive homomorphic encryption schemes, into which radixcaching is integrated, are IND-CPA. This is technically required because we will need this assumption to prove that Rache is IND-CPA. We call those original homomorphic encryption schemes base schemes, whose encryption function must not be deterministica necessary (but not sufficient) requirement for any scheme to be IND-CPA. In practice, many existing base schemes have been proven IND-CPA; for instance, both base schemes (Paillier [91], Symmetria [102]) used by Rache are IND-CPA.

### 3.4.2 Scheme Description

We start with integrating RHE into a symmetric homomorphic encryption scheme. We denote a quadruple

$$
\Pi=(G e n, E n c, D e c, \oplus)
$$

as a symmetric homomorphic encryption, where Gen denotes the function to generate a random key $k$ of length $n$, Enc denotes the encryption function parameterized with $k$ to encode a plaintext $m$ into a ciphertext $c$, Dec denotes a decryption function with parameter $k$ to decode $c$ back into $m$, and $\oplus$ denotes the additive operation over two ciphertexts $\operatorname{Enc}\left(m_{1}\right)$ and $\operatorname{Enc}\left(m_{2}\right)$ such that

$$
\operatorname{Dec}_{k}\left(E n c_{k}\left(m_{1}\right) \oplus E n c_{k}\left(m_{2}\right)\right)=m_{1}+m_{2} .
$$

A symmetric Rache scheme built upon $\Pi$ is a triple

$$
\begin{equation*}
\widetilde{\Pi}(G e n, R H E, D e c) \tag{3.2}
\end{equation*}
$$

where $R H E$ denotes the procedure defined in Algorithm 1. Note that $R H E(m)$ is equal to $\operatorname{Enc}(m)$ up to $O(n)$ random ciphertexts of zeros (out of the overall $r^{n}$ parameter space):

$$
R H E_{k}(m) \equiv E n c_{k}(m) \quad\left(\bigoplus_{I} E n c_{k}(0)\right)
$$

where $I$ is an index set whose cardinality is a polynomial on $n$. By definition, the equality $\operatorname{Dec}_{k}\left(R H E_{k}(m)\right)=m$ holds.

An asymmetric Rache scheme can be similarly built upon an asymmetric base HE scheme, except for the keys for Enc and Dec: two random keys—public key $p k$ and private key $s k$ are generated by Gen. For instance, we now require the following equality holds when RHE is built upon an asymmetric base scheme:

$$
\operatorname{Dec}_{s k}\left(R H E_{p k}\left(m_{1}\right) \oplus R H E_{p k}\left(m_{2}\right)\right)=m_{1}+m_{2} .
$$

Because RHE touches on only the encryption function, there is no need to differentiate between symmetric and asymmetric base schemes. Therefore, in the following discussion, we assume the underlying base scheme is symmetric for more succinct notations.

### 3.4.3 Provable Security

This subsection proves that the Rache scheme is IND-CPA. We first explain the intuition why Rache is CPA-secure and then give the formal proof.

Recall that Rache precomputes and caches $\log _{r} 2^{n}$ radix entries. If we assume the system picks the optimal $r=2$ in the worst case, then the scheme will simply cache $n$ radix entries. Therefore, those ciphertexts cached by Rache should not significantly help the adversary—who presumably runs a probabilistic polynomial-time (PPT) Turing machineas the overall space is exponential (Lines 11-23, Algorithm 1).

Technically, we want to reduce the problem of breaking the base homomorphic encryption scheme $\Pi$ to the problem of breaking its Rache extension $\widetilde{\Pi}$. That is, if a PPT adversary $\mathcal{A}$ takes an algorithm alg to break $\widetilde{\Pi}$, then $\mathcal{A}$ can efficiently (i.e., in polynomial time) construct another algorithm $a l g^{\prime}$ that calls $a l g$ as a subroutine to break $\Pi$ as well (simulating alg' with $a l g$ ). However, we assume that the base scheme is IND-CPA, so the above cannot happen-leading to a contradiction. We formalize the above in the following proposition.

Proposition 2. If HE scheme $\Pi$ is IND-CPA, then its Rache-extension $\widetilde{\Pi}$ defined in Eq.(3.2) is IND-CPA.

Proof. Let $C P A_{X}^{\mathcal{P}}$ denote the indistinguishability experiment with scheme $X$. The probability for $\mathcal{A}$ to successfully break $\Pi$ and $\widetilde{\Pi}$ are $\operatorname{Pr}\left[C P A_{\Pi}^{\mathcal{A}}=1\right]$ and $\operatorname{Pr}\left[C P A_{\widetilde{\Pi}}^{\mathcal{A}}=1\right]$, respectively. By assumption, the following inequality holds:

$$
\begin{equation*}
\operatorname{Pr}\left[C P A_{\Pi}^{\mathcal{A}}=1\right] \leq \frac{1}{2}+\epsilon, \tag{3.3}
\end{equation*}
$$

where $\epsilon$ is a negligible probability. By comparing $\Pi$ and $\widetilde{\Pi}$, the latter yields $n$ additional pairs of plaintexts and ciphertexts out of the total $2^{n}$ possible pairs in the worst case. Therefore, the following inequality holds:

$$
\begin{equation*}
\operatorname{Pr}\left[C P A_{\tilde{\Pi}}^{\mathcal{H}}=1\right]-\operatorname{Pr}\left[C P A_{\Pi}^{\mathcal{A}}=1\right] \leq \frac{\operatorname{poly}(n)}{2^{n}} . \tag{3.4}
\end{equation*}
$$

Combining Eq. (3.3) and Eq. (3.4) yields the following inequality:

$$
\operatorname{Pr}\left[C P A_{\bar{\Pi}}^{\mathcal{Y}}=1\right] \leq \frac{1}{2}+\epsilon+\frac{\operatorname{poly}(n)}{2^{n}}
$$

Now, we only need to show that the summation of the last two terms, $\epsilon+\frac{\text { poly(n) }}{2^{n}}$, is negligible. According to Lemma 1 and Lemma $2(\$ \sqrt[3]{3.2}$, this is indeed the case. Therefore, the probability for the adversary $\mathcal{A}$ to succeed in the $C P A_{\tilde{\Pi}}^{\mathcal{A}}$ experiment is only negligibly higher than $\frac{1}{2}$, proving that Rache is IND-CPA, as claimed.

### 3.5 Incremental Rache

### 3.5.1 Overview

While Rache can effectively precompute and cache those selected ciphertexts given an upper bound of the plaintexts, the principle cannot be applied to data streams where the maximal value is unknown a priori. To that end, we propose to dynamically precompute those $r$ powers when a newly seen maximum is observed. The key idea is straightforward: whenever the cipher encounters a plaintext that is significantly larger than the largest
(cached) value, we submit a request to expand the list of cached values by adding a few precomputed ciphertexts that are closer to the new large plaintext. The remaining job is then to quantify the meaning of significantly and $a \mathrm{few}$, which will be elaborated on in the remainder of this section. Before the formal discussion, we illustrate the high-level idea of incremental Rache by extending Example 2 into the following Example 3r recall that we have a good set of cached ciphertexts now for up to $r^{6}$, where $r=2$.

Example 3. Now let's assume that a new value 200 is being encrypted. In theory, we could compute Rache $(200)=\operatorname{Rache}\left(r^{6}\right) \oplus \operatorname{Rache}\left(r^{6}\right) \oplus \operatorname{Rache}\left(r^{6}\right) \oplus \operatorname{Rache}\left(r^{3}\right)$; however, this naive approach would not scale: at some point the cost of many $\oplus$ 's would outweigh that of the original encryption. An alternative is to precompute some larger ciphertexts and append them into Ctxt[]: Ctxt[7] $=\operatorname{Rache}\left(r^{7}\right)=\operatorname{Rache}(128)$. As a result, we can compute $\operatorname{Rache}(200)=\operatorname{Rache}\left(r^{7}\right) \oplus \operatorname{Rache}\left(r^{6}\right) \oplus \operatorname{Rache}\left(r^{3}\right)=\operatorname{Ctxt}[7] \oplus \operatorname{Ctxt}[6] \oplus \operatorname{Ctxt}[3]$, which saves one $\oplus$ in this example; but for larger plaintexts, the saving would look much more significant.

### 3.5.2 Definitions and Notations

We begin by defining two important building blocks of incremental Rache, pivot and nuance.

Definition 4 (Pivot). A pivot in incremental Rache is one plaintext whose ciphertext is precomputed and cached.

By definition, the preimage of every entry of the radixes [] array discussed in Alg. 11 is a pivot. However, the converse is not true in general for incremental Rache: we might optionally choose to cache more "important" ciphertexts in addition to those in radixes[].

Definition 5 (Nuance). A nuance in incremental Rache is a pair $(\xi, R H E(\xi)$ ), where $\xi$ is a plaintext and $R H E(\xi)$ is the Rache ciphertext of $\xi$.

We use $p=\Theta(\operatorname{poly}(n))$ to denote the asymptotic number of pivots that will be preprocessed. Common values for $p$ include $n^{c}, 1 \leq c \leq 5$ [9]. Similarly, we use $d=\Theta(\operatorname{poly}(n))$ to denote the asymptotic number of nuances that will be cached. We assume the plaintext can be encoded with the security parameter $n$. Again, we can pad shorter ones or break longer ones into blocks to ensure the aligned lengths. We denote by $m$ the number of plaintexts (thus $m \leq 2^{n}$ ).

### 3.5.3 Scheme Description

To make it more concrete, we slightly extend the triple expression of an HE scheme into a quintuple by considering the spaces of plaintexts and ciphertexts. Formally, we denote by quintuple $\Pi=(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ an HE scheme, where $\mathcal{P}$ is the set of plaintexts, $\mathcal{C}$ is the set of ciphertexts, $\mathcal{K}$ is the set of secret keys (for succinctness assuming the scheme is symmetric), $\mathcal{E}$ and $\mathcal{D}$ are sets of keyed encryption and decryption functions that satisfy the following predicate,

$$
\forall K \in \mathcal{K}, \forall x \in \mathcal{P}, \exists e_{K} \in \mathcal{E}, \exists d_{K} \in \mathcal{D}, d_{K}\left(e_{K}(x)\right)=x
$$

An incremental Rache is a septuple extended from $\Pi$ :

$$
\begin{equation*}
\widetilde{\Pi}=(\mathcal{P}, C, \mathcal{K}, \widetilde{\mathcal{E}}, \mathcal{D}, \mathcal{B}, \mathcal{N}) \tag{3.5}
\end{equation*}
$$

where $\mathcal{B}$ is a function from plaintexts to the set of the indexed pivots, $\mathcal{N}$ is a nuance function from a polynomial number of plaintexts to their ciphertexts, and $\widetilde{\mathcal{E}}$ is the set of keyed functions for incremental encryption. While $\mathcal{P}, \mathcal{C}, \mathcal{K}$, and $\mathcal{D}$ inherit the same semantics from $\Pi$, others need more explanation. We elaborate on $\mathcal{B}, \mathcal{N}$, and $\widetilde{\mathcal{E}}$ as follows.

We start with $\mathcal{B}$. Recall that we assume the size of the current data set is $m$, implying its index $m-1$ (counting from 0 ). The newly added data point, therefore, has index $m$. The value of function $\mathcal{B}(m)$ is calculated as the encryption of the largest pivot that is smaller than the new data point. If we sort the pivots $P_{i}$ 's in an increasing order ( $P_{0} \leq P_{1} \leq P_{2} \leq \ldots$ ), then we can formally define $\mathcal{B}$ as follows:

$$
\mathcal{B}(m) \stackrel{\operatorname{def}}{=} e_{K}\left(P_{i}\right)
$$

where $P_{i} \leq m<P_{i+1}$ and $i$ denotes the pivot index.

The nuance function $\mathcal{N}$ maps a logarithmic distance from $P_{i}$ to its encryption:

$$
\left.\begin{array}{rl}
\left.\mathcal{N}:\left[1, \left\lvert\, \frac{P_{i+1}-P_{i}}{2}\right.\right\rceil\right] & \rightarrow C \\
& \xi
\end{array}\right)
$$

where $\xi \in\left\{2^{j}: \forall j \in \mathbb{N}, 2^{j} \leq\left\lceil\frac{P_{i+1}-P_{i}}{2}\right\rceil\right\}$ and $e_{K} \in \widetilde{\mathcal{E}}$. By convention, we use $\operatorname{dom}(\mathcal{N})$ to denote the domain of function $\mathcal{N}$, i.e., the set of nuance plaintexts between two adjacent pivots. It is evident to see that the new data point, denoted Ptxt $[m]$, can be calculated as follows:

$$
\operatorname{Ptxt}[m]=P_{i}+\sum_{j=1}^{|\operatorname{dom}(\mathcal{N})|}\{0,1\} \times 2^{j} .
$$

We are now ready to define $\widetilde{\mathcal{E}}$. Let $R H E_{K}^{\text {inc }} \in \widetilde{\mathcal{E}}$ with key $K$, then an incremental encryption function in $\widetilde{\mathcal{E}}$ is defined as follows:

$$
\begin{aligned}
R H E_{K}^{i n c}(m) & \stackrel{\operatorname{def}}{=} e_{K}\left(P_{i}+\sum_{j=1}^{|\operatorname{dom}(\mathcal{N})|}\{0,1\} \times 2^{j}\right) \\
& =e_{K}\left(P_{i}\right) \oplus e_{K}\left(\sum_{j=1}^{|d o m(\mathcal{N})|}\{0,1\} \times 2^{j}\right) \\
& =e_{K}\left(P_{i}\right) \oplus \bigoplus_{j=1}^{\mid \operatorname{dom}(\mathcal{N} \mid} e_{K}\left(\{0,1\} \times 2^{j}\right) \\
& =\mathcal{B}(m) \oplus \bigoplus_{\xi \in \operatorname{dom}(\mathcal{N})} \mathcal{N}(\xi) \times\{0,1\} .
\end{aligned}
$$

### 3.5.4 Provable Security

We will demonstrate that incremental Rache is IND-CPA. We formalize the proof in the following proposition.

Proposition 3. If a homomorphic encryption $\Pi$ is IND-CPA, then its corresponding incremental Rache-extension $\widetilde{\Pi}$ defined in Eq. (3.5) is IND-CPA.

Proof. The probability for an adversary $\mathcal{A}$ to successfully break $\Pi$ and $\widetilde{\Pi}$ are $\operatorname{Pr}\left[C P A_{\Pi}^{\mathcal{A}}=1\right]$ and $\operatorname{Pr}\left[C P A_{\widetilde{\Pi}}^{\mathcal{P}}=1\right]$, respectively. By assumption, the following inequality holds:

$$
\begin{equation*}
\operatorname{Pr}\left[C P A_{\Pi}^{\mathcal{A}}=1\right] \leq \frac{1}{2}+\epsilon, \tag{3.6}
\end{equation*}
$$

where $\epsilon$ is a negligible probability. By comparing $\Pi$ and $\widetilde{\Pi}$, the latter yields $p+d$ additional pairs of plaintexts and ciphertexts out of the total $2^{n}$ possible pairs in the worst case.

Therefore, the following inequality holds:

$$
\begin{equation*}
\operatorname{Pr}\left[C P A_{\bar{\Pi}}^{\mathcal{A}}=1\right]-\operatorname{Pr}\left[C P A_{\Pi}^{\mathcal{A}}=1\right] \leq \frac{p+d}{2^{n}} . \tag{3.7}
\end{equation*}
$$

Combining Eq. (3.6 and Eq. 3.7) yields the following inequality:

$$
\operatorname{Pr}\left[C P A_{\bar{\Pi}}^{\mathcal{A}}=1\right] \leq \frac{1}{2}+\epsilon+\frac{p+d}{2^{n}}=\frac{1}{2}+\epsilon+\frac{\operatorname{poly}(n)}{2^{n}},
$$

where the last equality comes from the simple fact that the summation of two polynomials is also a polynomial:

$$
\forall x, y \in \operatorname{poly}(n):(x+y) \in \operatorname{poly}(n) .
$$

Now, we only need to show that the summation of the last two terms, $\epsilon+\frac{\text { poly }(n)}{2^{n}}$, is negligible. According to Lemma 1 and Lemma $2(8.2$ ), this is indeed the case. Therefore, the probability for the adversary $\mathcal{A}$ to succeed in the $C P A_{\widetilde{\Pi}}^{\mathcal{P}}$ experiment is only negligibly higher than $\frac{1}{2}$, proving that incremental Rache is IND-CPA, as claimed.

### 3.6 Evaluation

### 3.6.1 Objectives

We aim to answer the following questions experimentally:

- What is the performance overhead of encryption in outsourced databases? (\$3.6.3)
- How does Rache perform comparing with state-of-the-art HE schemes in term of computational time and scalability? ( $\$$ 3.6.4)
- How does incremental Rache help reduce the performance overhead of encrypting data streams? (§3.6.5)

Specifically, in $\$ 3.6 .3$, we report the performance overhead of homomorphic encryption schemes, i.e., Cassandra performance with and without data encryption. In $\S$ 3.6.4, we report the performance of Rache from three perspectives: comparison on three micro benchmarks ( $\S \S 3.6 .4 .1$ 3.6.4.3) , comparison on three real-world applications ( $\$$ 3.6.4.43.6.4.6), and scalability on the number of parallel cores and input sizes ( $\$ 3.6 .4 .7$ ). In $\$$ 3.6.5, we report the performance of incremental Rache from the following three perspectives. The performance and overhead are reported in sections $\S \$ 3.6 .5 .1$ 3.6.5.2. The effectiveness of incremental encryption for aggregation functions is reported in $\$$ 3.6.5.3. Lastly in $\S$ 3.6.5.4. we show that incremental Rache outperforms Symmetria even for an arbitrary message with the original cache.

### 3.6.2 Experimental Setup

### 3.6.2 1 Systems and Implementation

We implement Rache (both the batch and the incremental versions) upon two base schemes, an asymmetric scheme Paillier [91] and a symmetric one Symmetria [102]. Both base schemes have proven to be IND-CPA [91, 102]. Our implementation follows the same spirit of CryptDB [94], which leaves the vanilla database unchanged but plugs in the cryptographic subsystem as a middleware. As a result, we integrate Rache into Cassandra [72]
through the DataStax Java driver [30].

The project is managed by Maven 3.6.3 and compiled with Java 11. The parallelization (e.g., randomized radix additions on Lines 15-23, Algorithm 1 is implemented with OpenMPI 4.0.3 [85]. At the time of writing this chapter, the implementation consists of 29,584 lines of code.

We deploy the Rache-enabled Cassandra on a 10-node cluster hosted at CloudLab [38]. Each node is equipped with two 36-core Intel Xeon Platinum 8360Y CPUs, 256 GB ECC DDR4-2666 memory, and two 1 TB SSDs. The operating system image is Ubuntu 20.04.3 LTS. All servers are connected via a 1 Gbps control link (Dell D3048 switches) and a 10 Gbps experimental link (Dell S5048 switches). We only use the experimental links for our evaluation.

### 3.6.2.2 Configurations

Some of the most important parameters of Cassandra are as follows. The replica factor is set to three. Hinted handoff is enabled globally. The maximum throttle of each thread is the default $1,024 \mathrm{~KB}$. The internal buffers are flushed to disk every 10 seconds. The partitioner is the default Murmur3Partitioner. There is one seed node (i.e., node 0 ) with the SimpleSeedProvider class (implementing the SeedProvider interface). The concurrency of reads and writes (including materialized view writes) is set to 32 . The full specification can be found in the cassandra. yaml file in the source code.

### 3.6.2.3 Workloads

We have tested the system prototype with six workloads, all of which are publicly available. These workloads include three micro-benchmarks and three real-world applications.

The first benchmark is a micro-benchmark to quantify the cost of homomorphic encryption and homomorphic addition, respectively. For the former, a sequence of integers [0, 32,768 ) are homomorphically encrypted; for the latter, the ciphertexts stored at radix entries are homomorphically summed up in a round-robin fashion 32,768 times.

The second benchmark is TPC-H ver. 3.0.0 [117], a standard relational database benchmark. TPC-H allows the user to specify the scales of the generated data; in this work we set the scale as one, resulting in about one gigabyte of data. We will focus on the part table, which consists of 200,000 tuples.

The third benchmark is a dynamic set of random numbers for homomorphic encryption. This benchmark is mainly used for the purpose of weak scaling, allowing for the scalability test ranging between 1,024 and 32,768 numbers.

The first application is the U.S. national COVID-19 statistics from April 2020 to March 2021 [27]. The data set has 341 days of 16 metrics, such as death increase, positive increase, and hospitalized increase.

The second application is the human genome reference 38 [59], commonly known as $h g 38$, which includes 34,424 rows of singular attributes, e.g., transcription positions, coding regions, and number of exons, last updated in March 2020.

The third application is the history of Bitcoin trade volume [19] since it was first exchanged in the public in February 2013. The data consists of the accumulated Bitcoin exchange on a 3-day basis from February 2013 to January 2022, totaling 1,086 large numbers.

### 3.6.3 Performance with and without Homomorphic Encryption

We record the execution time of Cassandra when inserting three real-world data sets. The configuration of Cassandra and data set specification can be found in the previous section $\$ 3.6 .2$. We repeat the experiments three times and report both the average and the standard variation in the figure. To eliminate the possible caching effect, we truncate the table every time before starting the timer for the execution.

Figure 3.1 reports the results, which clearly shows that Rache significantly improves the performance of homomorphic encryption. For the Covid-19 data set (left column), the original Paillier scheme incurs $20 \times$ overhead while Rache only incurs about $2 \times$. For the Bitcoin data set (center column), the originial Paillier scheme incurs $4 \times$ overhead while Rache's overhead is negligible. Similarly, for the hg38 data set (right column), Paillier incurs about $10 \times$ overhead and Rache's overhead is marginal.


Figure 3.1: Performance with and without encryption schemes.


Figure 3.2: Homomorphic encryption and addition in Paillier.


Figure 3.3: Performance comparison on the TPC-H benchmark.

### 3.6.4 Batch Rache

### 3.6.4.1 Encryption vs. Addition

Fig. 3.2 shows that the homomorphic addition is a much cheaper operation than homomorphic encryption in Paillier. Regardless of the number of available cores, homomorphic encryption takes more than two orders of magnitude time than homomorphic addition.

### 3.6.4.2 TPC-H

We report Rache's performance of encoding the TPC-H [117] data in Fig. 3.3. We report the execution time of initializing the radixes and that of encoding with Rache, respectively. The former is referred to as Rache Init and the latter as Rache Exec in the figure. The initialization time of Rache is roughly flattened, showing a marginal increase when more cores are involved due to the inter-process communication (IPC) overhead. It should be noted that, however, the Rache Init overhead is a one-time cost. Specifically, the Init cost is the execution time to construct the Ptxt[] vector, which stores the radix values for future additive computation over ciphertexts. We observe that Rache outperforms Paillier by four orders of magnitude at all scales.

In general, the overhead incurred by Rache on different number of cores comes from the coordination of multiple processes and threads, such as MPI_Reduce that aggregates the partial summations over ciphertexts. The overhead discrepancy of different workloads, however, largely depends on the maximal value in the message space (assuming the radix $r$ is fixed). As we will see soon in the following sections, the Rache initialization overhead (i.e., Rache Init) is lower than others (i.e., Figures 3.4 3.7). This can be best explained by the fact that the Part relation in TPC-H has its maximal numeric values in the order of thousands, which are much smaller than other benchmarks. Because the maximal value is smaller in TPC-H, Rache needs to precompute and cache fewer ciphertexts during the initialization phase, which results in smaller overhead than other benchmarks. This observation also explains why the overhead stays roughly constant from one core to 32 cores: each core precomputes the same set of cached ciphertexts that are determined by radix $r$ and the maximal plaintext message, both of which are the same on $1-32$ cores.

### 3.6.4.3 Random Numbers

In this benchmark, $n$ random numbers are generated in a uniform distribution by modular $n$. We report the results of Rache and Paillier in Fig. 3.4. The Rache overhead stays roughly constant for different numbers of cores, but not as low as TPC-H. Despite the overhead, we observe that Rache's encoding time is about two orders of magnitude lower than Paillier's at all scales.


Figure 3.4: Encoding performance on random numbers.


Figure 3.5: Encoding the U.S. Figure 3.6: Encoding the human COVID-19 statistics.
 genome reference 38 .

### 3.6.4.4 U.S. COVID-19 Statistics

Fig. 3.5 reports the encoding performance of the U.S. COVID-19 statistics published at [27]. We observe that with few cores (e.g., 1 and 2) the overhead is smaller than the encoding cost, while with more cores (e.g., 16, 32) the per-core encoding is very efficient and takes less time than the overhead. Some of the overhead, i.e., precomputing and caching the large radixes, is unnecessary for those small values, and yet has to exist due to those extremely large values. We stress that the overhead is a one-time thing though: if there were, say, ten years of COVID-19 data, the overhead would look roughly the same and would be outweighed by the increased cost of encoding the data.

### 3.6.4.5 Human Genome Reference 38

Fig. 3.6 reports the encoding performance of Rache and Paillier on a database of human genome [59] (hg38) that was last updated in March 2020, under the umbrella of the Augustus gene prediction project [10]. As expected, Rache outperforms Paillier at all scales by orders of magnitude. In sheer contrast to the COVID-19 dataset, the initialization overhead of Rache in hg38 is much less significant: even at 32 -core, the overhead is less than $30 \%$. This is mainly due to a large number of plaintexts $(172,120)$, whose encoding time


Figure 3.7: Encoding the Bitcoin trade volume.


Figure 3.8: Weak scaling of the encryption of random numbers.


Figure 3.9: Encoding a variety of workloads with a fix number of 32 cores.
greatly outweighs the initialization, which is not trivial: 29 radixes for values as large as 248,937,123.

### 3.6.4.6 Bitcoin Trade Volume

We apply Rache and Paillier to the historical trade volume of Bitcoin exchanges since 2013 [19]. Fig. 3.7]shows that Rache outperforms Paillier by more than one order of magnitude, which is consistent with what we have found so far. The notable thing here is the large overhead incurred by Rache: on a single core, the overhead is on par with Rache's encoding time; on 32 cores, the overhead is on par with the Paillier processing time and orders of magnitude larger than Rache's encoding time. This phenomenon is due to two reasons. First, the Bitcoin trade volume consists of very large numbers-most are in the order of millions and the largest one is $4,956,849,516$ requiring 34 radixes. Second, the number of plaintexts is relatively small: there are 1,086 plaintexts, each of which records the Bitcoin exchange for the last three days.

### 3.6.4.7 Scalability

We evaluate the scalability of Rache in this section. We focus on the data sets of random numbers rather than specific benchmarks or applications simply because we can generate arbitrarily large data sets of random numbers. Fig. 3.8 reports the conventional weakscaling experiment. We control the workload to be proportional to the number of cores: 1,024 plaintexts for every core. That is, the workloads range from 1,024 to 32,768 plaintexts of uniformly distributed random numbers. In each workload, the maximal value is close to the maximal number due to the uniform distribution.

Rache outperforms Paillier by orders of magnitude at all scales. However, Rache seems to exhibit a higher slope of encoding time. We stress that the absolute values of Rache performance are sub-seconds (and the $y$-axis is logarithmic), therefore the overhead can be best explained by the IPC overhead. To verify this, we conduct the following experiment, in which we fix the number of cores but increase the workloads.

Fig. 3.9 shows the encoding time when we fix the number of cores as 32 but increase the number of plaintexts from 1,024 to 32,768 . We observe that when the IPC overhead is fixed (for 32 cores), the encoding time is proportionally increased regarding the workload size.

### 3.6.5 Incremental Rache

### 3.6.5.1 TPC-H

We compare Rache ${ }^{\text {I }}$ and Symmetria on TPC-H with the option "-s 100"; there are overall $20,000,000$ tuples in the Part table. We vary the number of pivots (i.e., $p$ ) on the $x$-axis between 2 and 64. We report the performance of Rache (without the overhead of constructing the pivots $p$ 's and nuances $d$ 's, which will be reported in the next experiment), and compare it against Symmetria in Fig. 3.10. Generally speaking, larger $p$ values allow Rache to complete faster because of the finer granularity of the gaps among $p$ 's as well as fewer nuances. Notably, Rache is about 3 x faster than Symmetria when $p=32$. If the plaintexts are overly split (e.g., $p=64$ ), the extra cost for maintaining the pivots may outweigh the benefit of $d$ dictionaries, causing suboptimal performance.


Figure 3.10: Performance parison on TPC-H (scale = 100), 20,000,000 tuples in table Part.


Figure 3.12: Performance overhead incurred by pivots and nuances when encrypting $2^{32}$ random plaintexts.

### 3.6.5.2 Random Numbers

We compare the performance of Symmetria and Rache when encrypting 1,024 random numbers of variable lengths in Fig. 3.11. We on the $x$-axis vary the ( $n, p$ ) pairs ranging

[^6]between 8 and 32, where $n$ indicates the bitstring length and $p$ indicates the number of pivots, respectively. We observe that Rache consistently outperforms Symmetria for all $(n, p)$ pairs by up to $50 \%$ reduction in running time, which is aligned with the results of the TPC-H benchmark in Fig. 3.10.

We measure the time overhead for precomputing pivots and nuances of $2^{32}$ random values. Note that this experiment has a much larger data set than that in Fig. 3.11(i.e., $1,024=2^{10}$ ) because we will, to a large extent, vary both the number of pivots $p=n^{x}, 2 \leq x \leq 5$ ( $x$ is considered as a practical upper bound in complexity theory [9]), and the number of nuances $d=n^{y}, x \leq y$. We set $n=32$, meaning that there are up to $2^{32}$ distinct values in the underlying data set. The $x$-axis of Fig. 3.12 enumerates those $(x, y)$ pairs.

### 3.6.5.3 Aggregating Encrypted Fields

For a simple aggregate query shown in Listing 3.1 (i.e., the average part size), its Rache execution on the scale-10 TPC-H is illustrated in the following equation:

$$
e_{k}\left(\sum_{i=1}^{2,000,000} s_{i}\right)=\bigoplus_{i=1}^{2,000,000} e_{K}\left(s_{i}\right),
$$

where $s_{i}$ denotes the value of the $P$ Size field of the $i$-th row of relation Part.

Listing 3.1: A simple SQL aggregate query on TPC-H.
-- TPC-H 3.0, "dbgen $-s$ 10", two million tuples

## SELECT AVG(P_Size)

FROM Part;

Directly adding up $e_{K}\left(s_{i}\right)$ is more costly than arithmetic operations because $\oplus$ on ciphertexts is number-theoretical. Rache allows us to cache the ciphertexts of both pivot and nuance along with their frequencies in plaintexts. Therefore, we can reduce the frequency of $\oplus$ by arithmetic $\times$ if the HE scheme supports it (Symmetria [102] does) and calculate the result as follows:

$$
e_{k}\left(\sum_{i=1}^{2,000,000} s_{i}\right)=\operatorname{freq}_{i}^{p} \times \bigoplus_{i=1}^{p} e_{K}\left(P_{i}\right)+\operatorname{freq}_{j}^{\xi} \times \bigoplus_{j=1}^{d} e_{K}\left(\xi_{j}\right),
$$

where $p$ and $d$ are much smaller than 200,000 (e.g., $p=d=32$ ), fre $q_{x}^{y}$ indicates the frequency of the $x$-th element in the $y$-container, and $e_{K}(\cdot)$ 's are part of the entries (trees of pivots and dictionaries of nuances) cached in memory.


Figure 3.13: Time breakdown of aggregating 200,000 tuples of table Part in TPC-H.


Figure 3.14: Aggregating time with different numbers of pivots on different TPC-H scales.


Figure 3.15: Rache speedup over Symmetria when computing nuances on-the-fly.

Fig. 3.13 reports the time for aggregating 200,000 Part.P_Size fields on scale-1 TPC-H, where each step aggregates additional 10,000 encrypted fields. We observe that the onestep cost of Symmetria is not constant: in a later step, it takes more time to aggregate the same number of new ciphertexts. This is concerning because it implies that the batch HE scheme is not scalable and would stop working at some point. To investigate how bad it could become, Fig. 3.14 reports the same workload on TPC-H of both scales-1 and scale-10; we did not report the scale-100 results because Symmetria finished only $53 \%$ $(10,550,000$ out of $20,000,000)$ ciphertext additions after 100 hours of execution. We observe that Rache can aggregate $2,000,000$ fields within a second while Symmetria takes
hours to complete the same workload.

### 3.6.5.4 Computing Nuances On-the-fly

The previous experiments assume that there is sufficient memory capacity to accommodate $p$ pivots and $d$ nuances. In certain application scenarios (e.g., edge computing [1,2], supply chains [109], system-on-chip [23]), we might have limited resources and may not be able to hold, say, $2^{32}$ nuances. Therefore, the following experiment will investigate the worst-case scenario where we are forced to compute nuances on the fly. We report the performance of adopting a single nuance for a random value in $\left[0,2^{64}\right)$ in Fig. 3.15 . The worst-case overhead of calculating a single nuance leads to as low as 1.3 x speedup over the vanilla Symmetria encryption. In the best case, i.e., when nuance is set to one, the speedup is over 2.1x.

### 3.6.6 Summary of Experimental Results

Rache Both micro benchmarks and real-world applications confirm the efficiency of Rache: Rache incurs insignificant overhead to Cassandra while the conventional Paillier encryption is 2-10 times slower. Rache also exhibits strong scalability on up to 32 cores and $32 \times$ larger input data size.

Incremental Rache Incremental Rache is $2-3 \times$ faster than Symmetria and the initialization overhead is as low as 10 ms . In particular, incremental Rache is $3-5$ orders of magnitude faster than Symmetria for aggregation workloads that are commonly deployed
in outsourced databases. Finally, incremental Rache outperforms Symmetria by 1.3-2.2× speedup even though incremental ciphertexts are not cached.

### 3.7 Summary

This chapter proposes radix-based parallel caching optimization for accelerating the performance of homomorphic encryption (HE) of outsourced databases in cloud computing. The key insight of the proposed optimization is caching selected radix-ciphertexts in parallel without violating existing security guarantees of the original HE scheme. We design the radix HE algorithm and apply it to both batch and incremental HE schemes; we demonstrate the security of those radix-based HE schemes by reducing the Rache-extended problem to the base HE schemes that are known IND-CPA. We implement the radix-based schemes as middleware of a 10 -node Cassandra cluster on CloudLab; experiments on six workloads show that the proposed caching significantly improves the performance of state-of-the-art HE schemes.

## CHAPTER 4

## GAUSSIAN ATTACKS AND BLOCKCHAIN-BASED AUDITING ON TOKENIZED INCENTIVE FOR FEDERATED MACHINE LEARNING SYSTEMS

### 4.1 Introduction

Federated learning (FL) has emerged as a new distributed computing paradigm to both enrich the available training data and protect the data privacy of participating clients. It has been found very useful in various fields ranging from industrial engineering [76], smart cities [131], and health care [70] to name a few due to its major advantage of distributed processing and effective privacy protection. To attract and retain clients in an FL system, multiple incentive mechanisms have been proposed; in particular, a tokenized incentive [50] was recently proposed, which was believed more practical than the existing monetary-based, offline incentive mechanisms. Other approaches to FL incentives include the use of game theory [65], auctions [35], contract theory [122], matching theory [24], and currency [129].

This chapter demonstrates that, under mild assumptions, the tokenized incentive mechanism [50] for FL systems can be effectively compromised by a fraction of colluded clients who share their local training models with deliberate Gaussian noises-a new type of model-poisoning attack on FL incentives. We will show that, both analytically and empirically, if the noises are well controlled, the FL aggregation will converge and therefore
the Gaussian attack cannot be detected. As a result, those malicious clients could be rewarded with disproportionate tokens in addition to the degraded performance of FL systems due to the poisoned local models. To make matters more concrete, Figure 4.1 illustrates the proposed Gaussian attack on tokenized incentives in a decentralized FL system. In this figure, we present a simplified FL system with five clients, two of which ( $X$ and $Y$ ) are malicious and launch a Gaussian attack: only one malicious client, say $X$, follows the FL aggregation algorithm to train a local model and the other malicious client $Y$ simply copies over the model of $X$ and adds some well-controlled Gaussian noises. Suppose other three honest clients $A, B$, and $C$ produce accurate models, then the tokens rewarded to $X$ and $Y$ are $\frac{2}{5}$ of the budgeted tokens, which is unfair to those honest clients because $X$ and $Y$ only contribute $\frac{1}{4}$ to the overall computational power.

To mitigate the Gaussian attack, this chapter then designs a blockchain-based auditing protocol, in which (i) clients must hash their local training models into a Merkle tree [81] in each round of global synchronization and (ii) clients must persist the hashed models into a permissioned blockchain, e.g., SciChain [4], where all local models are peer-verified with Byzantine fault tolerance. As a result, if a client is suspected to have launched a Gaussian attack, it will be slated for auditing by verifying its immutable training provenance on the blockchain. We also sketch the proposed blockchain-based auditing protocol in Figure 4.1 . the right half of the figure illustrates the idea among three honest clients who hash and persist their local training models on a blockchain.

We have implemented the proposed Gaussian attack and Blockchain-based auditing with FedML [53] and SciChain [4]. Our extensive experiments with multiple FL algorithms (FedAvg [78], MultiKrum [20]), popular data sets (MNIST [83], Fashion-MNIST [123], CIFAR-10 [71], SVHN [88]), and a variety of scales (50, 100, 200, and 500 clients) demonstrate the effectiveness of the attack (e.g., Gaussian variance under 0.04) and the efficiency


Figure 4.1: Gaussian attack and blockchain audit for decentralized federated learning
of blockchain-based auditing ( $10 \%$ training overhead).

In summary, this chapter discusses the following contributions.

- We develop a new model-poisoning attack on FL in which a fraction of malicious clients shares the same training model with deliberated Gaussian noises to receive disproportionate incentive tokens. Theoretically, we prove that the Gaussian attack
does not impact the convergence of FL aggregation algorithms such as FedAvg and MultiKrum.
- We propose a new auditing mechanism enlightened by blockchains to detect the Gaussian attack on tokenized incentive for FL: in each global synchronization round, every client must construct a Merkle tree of all local models and verify them in a decentralized manner. The verified local modes, once persisted to the blockchain, cannot be altered unilaterally and will serve as evidence of malicious activities.
- We implement both the attack and audit on state-of-the-art FL and blockchain systems. Extensive experiments are carried out on four popular data sets, two widelyused FL aggregation algorithms, and on up to 500 clients, which demonstrate the effectiveness of the proposed attack (0.04 Gaussian variance) and the efficiency of the blockchain audit ( $10 \%$ time overhead).


### 4.2 Background and Related Work

### 4.2.1 Incentive Mechanisms for Federated Learning

Client participation in the FL process incurs some cost for contributing to the FL model with their local data set and computational power [50]. Various forms of incentive mechanisms have been proposed to promote the participation of clients in the FL process. Khan et al. [65] proposed a game-theoretic mechanism (i.e., Stackelberg game) for FL to ensure communication efficiency and model accuracy improvement for the leader and revenue improvement for the followers. Deng et al. [35] proposed an auction-based incentive mechanism (i.e., reverse auction) for FL where the learning tasks allocation and payment
determination are determined by solving the problem of maximizing the sum of the quality of aggregated model updates. Wu et al. [122] presented a contract-theoretical incentive for FL by jointly considering the task expenditure and privacy risk of data owners. Chen et al. [24] proposed a matching-theoretical incentive for FL systems. Han et al. [50] address FL incentives as tokens where tokens paid by consumers are given to the selected providers proportionately according to their contribution to model update and the remaining tokens are distributed to all providers according to their participation frequencies to promote longterm active participation in a federated machine learning system.

### 4.2.2 Attacks on Federated Learning

Numerous attacks could occur in FL systems due to vulnerabilities leading to unfairness or degradation of system operation. Some examples of such attacks are the free rider attacks and poisoning attacks. Free rider attacks are one in which a client or portion of system clients maliciously receive incentives (called tokens in our case) from the services of others without contributing any of their resources and datasets [42]. There are two different freerider attack scenarios: anonymous free-riders (AFRs) who do not possess any computation power and privacy datasets and selfish free-riders (SFRs) who possess privacy dataset but are unwilling to participate in model training nor utilize their computation power [120].

A poisoning attack is also a major form of attack that could arise while using machine learning models. For example, when adversary or malicious clients deliberately add compromised samples to the training pool of the model [5], it was called a data poisoning attack. Similarly, a malicious client can upload an arbitrary model to the aggregator-a model poisoning attack (also termed adversarial attack) [75]. Many known attacks belong
to this category, such as backdoor attacks [11, 17, 124, 125]. Several defense approaches have been proposed to alleviate poisoning attacks in FL. These approaches can be broadly classified as server-based or client-based defense approaches. Sun et al. [115] proposed a client-based defense, named White Blood Cell for Federated Learning (FL-WBC) to mitigate model poisoning attacks that have already polluted the global model. Yin et al. [128] proposed a server-based defense approach through robust aggregation to improve the robustness of FL against model poisoning attacks. Similarly, Sun et al. [114] proposed a server-based defense by clipping local updates to mitigate poisoning attacks. Krum [20] mitigated poisoning attacks in FL through the utilization of the similarity of benign clients' local updates. Shejwalkar and Houmansadr [108] designed a defense against FL poisoning called divide-and-conquer.

The free-rider attack is a fairly general type of attack as long as the client doesn't train its local model. From this general perspective, the proposed Gaussian attack (i.e., modelpoisoning attack) belongs to this category. However, we would like to highlight that the proposed attack exhibits two differences: (i) the goal of adversaries is to obtain unfair token rewards that were recently proposed in [50], and (ii) the adversaries are assumed to be willing to collude, which may or may not be true for general free-rider attacks.

### 4.2.3 Blockchains for Federated Learning

Through immutable transaction records and distributed consensus mechanisms, blockchain systems [60] can enable secure interaction in an untrustworthy environment without a centralized mediator. The integration of FL and blockchain thus has drawn a lot of research interest. Zhang et al. [129] proposed a blockchain-powered FL system called Refiner to
tackle the challenges introduced by engaging self-interested and malicious clients. In [25], a blockchain-powered FL is proposed to guard against adversary attacks. In this framework, clients upload updates to verifiers, who will vote to select benign updates, and then the selected updates are aggregated and written to blocks via the blockchain network. Biscotti [107] is a Blockchain-based FL architecture proposed to address a single point failure, poisoning attack, and privacy leakage in FL. Likewise, Kim et al. [69] presented BlockFL to address issues of a single point of failure and the lack of motivation among participants in FL. As opposed to the above work, this work identifies a new attack on FL incentives and designs a blockchain-based auditing protocol to detect such malicious activities.

### 4.3 Gaussian Attack on Token Incentive for FL

### 4.3.1 Threat Model

We assume the FL system adopts a decentralized architecture without a centralized parameter server or aggregator. That is, the proposed Gaussian attack does not rely on a centralized aggregator. Clients can go malicious and launch arbitrary attacks, i.e., Byzantine failures. This implies that a client can poison the data, the model, or the label during local training. This also means multiple clients can collude, e.g., a malicious client could share its (either clear or poisoned) model with other malicious ones. However, we do assume the ratio of malicious clients does not dominate the system, i.e., exclusively less than $50 \%$. In addition, we assume that a malicious client will not share its private key.

The following three model assumptions are needed for the theoretical proof of conver-
gence of the proposed Gaussian attack on tokenized incentives. We denote the objective function $f$, any network model parameter $\boldsymbol{\theta}_{1}$ and $\boldsymbol{\theta}_{2}$, and gradient descents $\nabla f(\boldsymbol{\theta})$.

1. The objective function is convex:

$$
f\left(\boldsymbol{\theta}_{2}\right) \geq f\left(\boldsymbol{\theta}_{1}\right)+\left\langle\nabla f\left(\boldsymbol{\theta}_{1}\right), \boldsymbol{\theta}_{2}-\boldsymbol{\theta}_{1}\right\rangle, \forall \boldsymbol{\theta}_{1}, \boldsymbol{\theta}_{2} .
$$

2. The network parameters are bounded:

$$
\left\|\boldsymbol{\theta}_{2}-\boldsymbol{\theta}_{1}\right\|_{2} \leq D .
$$

3. The gradient descents are bounded:

$$
\|\nabla f(\boldsymbol{\theta})\|_{2} \leq G .
$$

### 4.3.2 Attack Algorithm

Algorithm 2 illustrates the Gaussian attack launched collectively by $F$ malicious clients. Lines 1 and 16 are straightforward: they initialize the network parameters and broadcast their local models to the entire network of $N$ clients in each round, respectively. Recall that there is no single aggregation server under our assumption.

Lines 2-17 iteratively carry out $T$ global rounds of training. The idea is to select only one of those $F$ malicious clients to train a reasonably accurate model and then share this model with all other $F-1$ colluding clients. Line 3 selects a random client $S$ out of the
$F$ malicious clients. We assume here that those $F$ malicious clients all agree on this $S$ selection, e.g., through some consensus protocols. Then, the attack differentiates between the seed and its followers:

- The seed initializes the network model from the previous (global) round on Line 8 and then iterates overall local epochs and batches on Lines $9-13$. Finally, the seed shares its local model with other colluding clients by broadcasting the model on Line 14.
- The followers simply wait for the trained model from the seed on Line 5. The followers then add decaying noises on Line 6, i.e., $\left\|\boldsymbol{\sigma}^{\boldsymbol{t}}\right\|_{2}=\frac{K}{t^{p}}$, to the seed's model to fake it as their locally-trained models such that (i) the entire FL system cannot identify the replication and (ii) the entire FL system still converges and the cheating cannot be detected. The objective of this so-called "Gaussian attack" is not to compromise the FL system; rather, this collusion of $F$ clients (i) earns $F$ shares of tokens while only investing a single share of computation resources and (ii) deviates the accuracy of the FL system with a fraction of $\frac{F-1}{N}$ poisoned local models.


### 4.3.3 Convergence Analysis

The goal of this section is to show that the Gaussian attack depicted in Alg. 2 will render the loss function to converge to 0 when the number of global rounds tends to infinity. Formally, we want to show the following:

$$
\begin{equation*}
\frac{\sum_{t=1}^{T} f_{t}\left(\boldsymbol{\theta}^{t}\right)-\sum_{t=1}^{T} f_{t}\left(\boldsymbol{\theta}^{*}\right)}{T} \longrightarrow 0, \text { as } T \longrightarrow \infty \tag{4.1}
\end{equation*}
$$

```
Algorithm 2: Gaussian attack on FL incentives
                \(p \in \mathbb{R}\); bounded random Gaussian variance \(\sigma\);
    Output: Global model \(\boldsymbol{\theta}\);
    Initialize \(\boldsymbol{\theta}_{i}^{0}\)
    for \(t=1 ; t \leq T ; t++\) do
        \(S \leftarrow\{1 . . F\}\)
        if \(i \neq S\) then
            \(\operatorname{recv}\left(\boldsymbol{\theta}_{S}^{t}\right)\)
            \(\boldsymbol{\theta}_{i}^{t} \leftarrow \boldsymbol{\theta}_{S}^{t}+\boldsymbol{\sigma}_{i}^{t}\)
    else
        \(\boldsymbol{\theta}_{S}^{t}:=\boldsymbol{\theta}_{S}^{t-1}\)
        for in range \((E)\) do
            \(\overline{\text { for }}\) in \(\mathcal{B}\) do
            \(\mid \overline{\boldsymbol{\theta}}_{S}^{t}:=\boldsymbol{\theta}_{S}^{t}-\alpha_{t} \cdot \nabla f_{t}\left(\boldsymbol{\theta}_{S}\right)\)
            end
            end
            Broadcast \(\boldsymbol{\theta}_{S}^{t}\) to \(\left\{n_{1}, \ldots, n_{F}\right\}\)
        end
        Broadcast \(\boldsymbol{\theta}_{i}^{t}\) to \(\left\{n_{1}, \ldots, n_{N}\right\}\)
    end
```

    Input: A total of \(N\) clients, in which \(F\) are malicious: \(\left\{n_{1}, \ldots, n_{F}\right\}\); the index \(i\) of
            current malicious client \(n_{i}\); total number of global rounds \(T\); total number
            of local epochs \(E\); local training samples \(\mathcal{D}_{i}\) on \(n_{i}\); loss function \(f(\cdot)\);
            decaying learning rate \(\alpha_{t}=\frac{C}{t}\), where \(1 \leq t \leq T, t \in \mathbb{N}\) and \(p \in(0,1)\),
    where $\boldsymbol{\theta}^{*}$ denotes the optimal model parameter (i.e., minimizing $f$ ): $\boldsymbol{\theta}^{*} \stackrel{\text { def }}{=} \arg \min _{\boldsymbol{\theta}} \sum_{t=1}^{T} f_{t}(\boldsymbol{\theta})$. Let $R(T) \stackrel{\operatorname{def}}{=} \sum_{t=1}^{T} f_{t}\left(\boldsymbol{\theta}^{t}\right)-\sum_{t=1}^{T} f_{t}\left(\boldsymbol{\theta}^{*}\right)$ and $\boldsymbol{g} \stackrel{\text { def }}{=} \nabla f(\boldsymbol{\theta})$. According to model assumption 1, the following holds

$$
\begin{equation*}
R(T) \leq \sum_{t=1}^{T}\left\langle\boldsymbol{g}, \boldsymbol{\theta}^{t}-\boldsymbol{\theta}^{*}\right\rangle \tag{4.2}
\end{equation*}
$$

According to Line 11 in Alg. 2. we have

$$
\boldsymbol{\theta}_{S}^{t+1}-\boldsymbol{\theta}^{*}=\boldsymbol{\theta}_{S}^{t}-\boldsymbol{\theta}^{*}-\alpha_{t} \boldsymbol{g}_{t} .
$$

Taking the squared $\ell_{2}$-norm on both sides yields:

$$
\left\|\boldsymbol{\theta}_{S}^{t+1}-\boldsymbol{\theta}^{*}\right\|_{2}^{2}=\left\|\boldsymbol{\theta}_{S}^{t}-\boldsymbol{\theta}^{*}\right\|_{2}^{2}+\alpha_{t}^{2}\left\|\boldsymbol{g}_{t}\right\|_{2}^{2}-2 \alpha_{t}\left\langle\boldsymbol{g}_{t}, \boldsymbol{\theta}_{S}^{t}-\boldsymbol{\theta}^{*}\right\rangle .
$$

Moving the inner product to the left yields:

$$
\left\langle\boldsymbol{g}_{t}, \boldsymbol{\theta}_{S}^{t}-\boldsymbol{\theta}^{*}\right\rangle=\frac{\left\|\boldsymbol{\theta}_{S}^{t}-\boldsymbol{\theta}^{*}\right\|_{2}^{2}-\left\|\boldsymbol{\theta}_{S}^{t+1}-\boldsymbol{\theta}^{*}\right\|_{2}^{2}}{2 \alpha_{t}}+\frac{\alpha_{t}\left\|\boldsymbol{g}_{t}\right\|_{2}^{2}}{2}
$$

Consider Line 6 in Alg. 2 , we know that when $i \neq S$ :

$$
\left\langle\boldsymbol{g}_{t}, \boldsymbol{\theta}_{i}^{t}-\boldsymbol{\theta}^{*}\right\rangle=\left\langle\boldsymbol{g}_{t}, \boldsymbol{\theta}_{S}^{t}+\sigma_{i}^{t}-\boldsymbol{\theta}^{*}\right\rangle \leq\left\langle\boldsymbol{g}_{t}, \boldsymbol{\theta}_{S}^{t}-\boldsymbol{\theta}^{*}\right\rangle+\frac{K\left\|\boldsymbol{g}_{t}\right\|_{2}}{t^{p}} .
$$

Plug the above two equations into Eq. (4.2):

$$
\begin{equation*}
R(T) \leq R_{1}(T)+R_{2}(T) \tag{4.3}
\end{equation*}
$$

where

$$
R_{1}(T)=\sum_{t=1}^{T} \frac{\left\|\boldsymbol{\theta}_{S}^{t}-\boldsymbol{\theta}^{*}\right\|_{2}^{2}-\left\|\boldsymbol{\theta}_{S}^{t+1}-\boldsymbol{\theta}^{*}\right\|_{2}^{2}}{2 \alpha_{t}}
$$

and

$$
R_{2}(T)=\sum_{t=1}^{T} \frac{\alpha_{t}\left\|\boldsymbol{g}_{t}\right\|_{2}^{2}}{2}+\frac{K\left\|\boldsymbol{g}_{t}\right\|_{2}}{t^{p}}
$$

If we expand $R_{1}(T)$, the intermediate terms will cancel in pairs and the last term will be negative, which yields:

$$
\begin{align*}
R_{1}(T) & \leq \frac{\left\|\boldsymbol{\theta}^{0}-\boldsymbol{\theta}^{*}\right\|_{2}^{2}}{2 \alpha_{0}}+\sum_{t=2}^{T}\left(\frac{1}{2 \alpha_{t}}-\frac{1}{2 \alpha_{t-1}}\right)\left\|\boldsymbol{\theta}^{t}-\boldsymbol{\theta}^{*}\right\|_{2}^{2} \\
& \leq \frac{D^{2}}{2 \alpha_{0}}+D^{2} \sum_{t=2}^{T}\left(\frac{1}{2 \alpha_{t}}-\frac{1}{2 \alpha_{t-1}}\right) \\
& =\frac{D^{2}}{2 \alpha_{T}}=\frac{D^{2}}{2 C} \cdot T^{p} \tag{4.4}
\end{align*}
$$

where the second inequality is due to model assumption 2 . Now, expanding $R_{2}(T)$ with nontrivial $T>1$ and applying model assumption3 yields:

$$
\begin{align*}
R_{2}(T) & \leq \frac{G^{2}}{2} \sum_{t=1}^{T} \alpha_{t}+K G \sum_{t=1}^{T} \frac{1}{t^{p}} \\
& =\left(\frac{C G^{2}}{2}+K G\right) \sum_{t=1}^{T} \frac{1}{t^{p}} \\
& \leq\left(\frac{C G^{2}}{2}+K G\right) \int_{1}^{T} \frac{d t}{t^{p}} \\
& =\frac{C G^{2}+2 K G}{2(1-p)} \cdot T^{1-p} . \tag{4.5}
\end{align*}
$$

Combining Equations (4.2), (4.4), and (4.5) yields

$$
R(T) \leq \frac{D^{2}}{2 C} \cdot T^{p}+\frac{C G^{2}+2 K G}{2(1-p)} \cdot T^{1-p}=O\left(T^{\max (p, 1-p)}\right),
$$

where $0<p<1$. Without loss of generality, let $0<p<0.5$ and we have $\max (p, 1-p)=$ $1-p$. Direct computation is then in order:

$$
\lim _{T \rightarrow \infty} \frac{R(T)}{T}=\lim _{T \rightarrow \infty} \frac{O\left(T^{1-p}\right)}{T}=\lim _{T \rightarrow \infty} \frac{1}{O\left(T^{p}\right)}=0
$$

as desired in Equation (4.1).

### 4.4 Blockchain Audit for Local Models in FL

### 4.4.1 Assumptions

We assume the hashing function for the local updated models is practically strong and cannot be distinguished from a random oracle. While the existence of a perfectly random oracle is still debatable, from a practical point of view we simply assume the hashing function looks random to the adversaries. If a collision is indeed found, the hash will be replaced. By convention, we also assume the malicious clients can only run efficient algorithms, i.e., the attack should take up to probabilistic polynomial time (PPT).

### 4.4.2 Auditing Protocol

Algorithm 3 illustrates the auditing protocol of persisting local models to blockchains. We only present the high-level, descriptive statements instead of implementation details, which can be found in the source code on Github. The algorithm is again for a specific client $n_{i}$, $i \in \mathbb{N}, 1 \leq i \leq N$. After initializing the model parameter on Line 1 , the client gets involved in $T$ rounds of iterations. On line 3, the client $n_{i}$ is supposed to update its local model, e.g., following the civil procedure as depicted on Lines 8-13 of Algorithm 2 (as an honest client or the seed of collusion of malicious clients).

Starting Line 4, the protocol mandates each client to carry out a series of hashing operations on its local model weights through the MerkleTree() method. The local model weights are usually stored as a list of tensors, each of which corresponds to a specific

```
Algorithm 3: Blockchain auditing of local models
    Input: A total of \(N\) clients; the index \(i\) of current malicious client \(n_{i}\); total number
            of global rounds \(T\);
    Output: Global model \(\boldsymbol{\theta}\);
    Initialize \(\boldsymbol{\theta}_{i}^{0}\)
    for \(t=1 ; t \leq T ; t++\mathbf{d o}\)
        \(\boldsymbol{\theta}_{i}^{t}:=\) updated local model // Lines 8-13, Alg. 2
        \(H_{i}^{t}:=\operatorname{MerkleTree}\left(\boldsymbol{\theta}_{i}^{t}\right)\)
        Broadcast \(H_{i}^{t}\) to \(\left\{n_{1}, \ldots, n_{N}\right\}\)
        Receive \(\left\{H_{1}^{t}, \ldots, H_{N}^{t}\right\}\)
        Construct block \(B_{i}^{t}\) from \(\left\{H_{1}^{t}, \ldots, H_{N}^{t}\right\}\)
        \(S:=\) client ID with BFT consensus protocol
        if \(i==S\) then
            Append \(B_{i}^{t}\) to blockchain
        end
        Broadcast \(\boldsymbol{\theta}_{i}^{t}\) to \(\left\{n_{1}, \ldots, n_{N}\right\}\)
    end
```

layer in the local (neural network) model. On Lines 5-6, each client broadcasts its updated model and receives all others' models. On Line 7, each client packages its hash values into a block, which is slated to be appended to the blockchain. However, only one client will be able to do so, and this is determined by a consensus protocol on Line 8. If $n_{i}$ turns out to be the client to append the new block, as has been agreed on by all clients, $n_{i}$ will append the hashed value of all local models in this round to the blockchain on Line 10. On Line 12 , each client broadcasts its local model to the entire network, just as the conventional decentralized FL system.

### 4.4.3 Complexity Analysis

We are interested in two metrics regarding the performance of the proposed blockchain auditing: the additional messages and the total number of rounds of message passing. The former measures the (network) I/O overhead and the latter measures the time overhead incurred by Algorithm 3, respectively. In the following analysis, we do not consider the
costs of the original FL algorithm (e.g., Lines 1, 3, and 12) since they are incurred by the blockchain auditing.

I/O overhead. Lines 5 and 6 incur $N^{2}$ messages. Each message is a hash value of constant size $M$, e.g., 32 bytes for SHA-256. Line 8 involves $O\left(N^{2}\right)$ messages due to BFT. Line 10 incurs $O(M N)$ I/O overhead. Recall that each broadcast is invoked $T$ times (Line $2)$, therefore, the overall I/O overhead is $O\left(T M N^{2}\right)$.

Time overhead. Line 4 takes an additional $O(L \log L)$ to construct the Merkle tree, where $L$ denotes the number of layers in the network model. Lines 5 and 6 take $O\left(\frac{M N}{I}\right)$, where $I$ denotes the bandwidth of the network. Line 7 takes $O(N)$ to package all of $N$ hash values. Line 8 takes a constant times of message broadcasting, $O\left(\frac{M}{I}\right)$. Therefore, the overall time overhead is $O\left(T L \log L+\frac{T M N}{I}\right)$.

### 4.5 Evaluation

### 4.5.1 Implementation

We implement the proposed attack and auditing with FedML [53], SciChain [4], PyTorch [95], MPI [85], and MPI4py [86]. FedML is a popular FL framework that supports a variety of APIs, such as MPI, RPC, and HTTPS (ongoing). SciChain is a permissioned blockchain system originally designed for tracking the provenance of scientific applications. PyTorch is a popular machine learning framework with many built-in data sets. MPI is the de facto communication primitive for high-performance computing, and MPI4py is the Python
wrapper of the MPI library. We use SHA256 as the hashing algorithm.

Each client is assigned a dedicated CPU core that is managed by an MPI rank. The consensus protocol is implemented with C++ and MPI. The FL-level communication is implemented with collective communication calls with MPI4py. The Gaussian attack is implemented as part of the iterative training in FedML: overwriting the local model by the seed's model with the addition of Gaussian variances. Blockchain auditing is implemented with the chained hashing of various layers of the local model in the same way as the Merkle tree and the C++ implementation of the BFT protocol in SciChain.

### 4.5.2 Experimental Setup

All experiments are repeated at least three times and the average numbers are reported.

### 4.5.2.1 Test Bed

We carry out extensive experiments on a cluster of 16 nodes, each of which is equipped with 32 Intel Xeon Gold-6142 cores at 2.6 GHz and 384 GB RAM. Overall, there are 512 physical cores and we use up to 500 cores to evaluate 500 clients. The operating system is Ubuntu 20.04. For execution, we use OpenMPI 4.0.3, Python 3.8.10, and MPI4py 3.1.3. The Python libraries are managed by pip 20.0.2. As for PyTorch, we use torch 1.12.0 and torchvision 0.13.0.

### 4.5.2 2 Data Sets

We choose four popular datasets, all of which are publicly available from PyTorch: MNIST [83], Fashion-MNIST or FMNIST [123], CIFAR-10 [71], and SVHN [88].

- Both MNIST and Fashion-MNIST have 50,000 training samples and 10,000 testing samples.
- CIFAR-10 has 50,000 training samples and 10,000 testing samples.
- SVHN (with extra turned on) has 604,388 training samples and 26,032 testing samples.


### 4.5.2.3 Machine Learning Models

We pick two machine learning models to train neural networks on those four data sets: convolutional neural network (CNN) and multi-layered perception (MLP). Both models use ReLU and SoftMax as the activation functions.

- We apply the CNN model to MNIST and CIFAR-10. The CNN model comprises two convolutional layers: the first from input data to 10 with kernel size 5 and the second from 10 to 20 with kernel size 5 . The CNN model applies two linear layers: $320 \rightarrow 50 \rightarrow 10$.
- We apply the MLP model to Fashion-MNIST and SVHN. The MLP model has a 64-neuron hidden layer, and the final network has three layers: $784 \rightarrow 64 \rightarrow 10$.


### 4.5.2.4 Federated Learning Parameters

The number of global rounds is set as 10 . The number of local epochs is set as 20 . The fraction of clients for each round of model updating is $100 \%$ : all of the clients participate in the model updating. The local batch size is set to 100 . The SGD momentum is 0.5 and the learning rate is $\frac{1}{20 \cdot t^{0.9}}$, where $t$ denotes the epoch number. The training samples are randomly and uniformly split into available clients. We test two popular FL aggregation algorithms: FedAvg [78] and MultiKrum [20].

- The FedAvg algorithm requests all the clients to broadcasts their local models and calculate the average weights of all the received models for the updated global model in the next round.
- The MultiKrum algorithm mandates all the clients to broadcast their local modes and calculate their next-round model by dropping a predetermined number $(F)$ of Byzantine clients whose Euclidean distances to the majority of clients are ranked top $F$.


### 4.5.3 Effectiveness of Gaussian Attacks

Figure 4.2 reports the loss function changes of the standard FedAvg [78] FL aggregation on four data sets when $49 \%$ of clients launch Gaussian attacks with different variances ranging between 0.0 and 0.64 . When variance $\operatorname{Var}=0$, the performance is reported for the FL system with no Gaussian attack-the baseline case. Unsurprisingly, when no Gaussian attack is launched, the loss drops significantly within 10 global rounds. It is also expected to observe a slower dropping with a larger variance: the model is more poisoned. The
point here is, however, that with small Gaussian variances added to the model, such as Var $\in\{0.04,0.16,0.36\}$ for MNIST/F-MNIST and Var $\in\{0.04,0.16,0.64\}$ for SVHN, the plots exhibit no noticeable difference between the baseline and the poisoned models after 10 rounds of FL training. This implies that the FL system cannot detect the existence of Gaussian attacks and would grant tokens to malicious clients.


Figure 4.2: FedAvg loss under Gaussian attacks for MNIST, Fashion-MNIST, CIFAR-10, and SVHN

Figure 4.3 reports the loss function changes of the MultiKrum [20] FL aggregation on the same set of four data sets when $49 \%$ of clients launch Gaussian attacks with different variances ranging between 0.0 and 0.64 . We observe that even if MultiKrum seems more resilient than FedAvg, the former can be nonetheless compromised with a smaller Gaussian variance, e.g., Var $=0.04$. The FL system cannot effectively detect such attacks and would nonetheless distribute disproportionate tokens to those malicious clients, making those honest clients worse off and hurting the retention of clients to participate in FL.


Figure 4.3: MultiKrum loss under Gaussian attacks for MNIST, Fashion-MNIST, CIFAR-10, and SVHN


Figure 4.4: FedAvg accuracy under Gaussian attacks for MNIST, Fashion-MNIST, CIFAR-10, and SVHN


Figure 4.5: MultiKrum accuracy under Gaussian attacks for MNIST, Fashion-MNIST, CIFAR-10, and SVHN
Figures 4.4 and 4.5 report the accuracy performance of FedAvg and MultiKrum on four data sets when $49 \%$ of clients launch Gaussian attacks with different variances ranging between 0.0 and 0.64 . The accuracy is highly correlated to the (inverse of) loss function, as expected. Both figures demonstrate that when Var is small, the overall accuracy performance of the FL system is about the same with or without Gaussian attacks.

### 4.5.4 Cost of Blockchain Auditing

Figure 4.6 reports the computational and I/O costs incurred by the proposed blockchainbased auditing. Our experiments are carried out between 50 and 500 clients. Because


Figure 4.6: Computational and I/O costs of blockchain auditing for MNIST, Fashion-MNIST, CIFAR-10, and SVHN
we are conducting a strong-scaling experiment, i.e., fixing the overall size of data and increasing the number of resources, we observe decreased overall running time on more clients.

We report the time overhead incurred by the two main components of the proposed blockchain auditing: local model hashing (SHA-256 Hashing) and consensus protocol (Blockchain Consensus). On the one hand, we observe that the hashing overhead is negligible: the hashing time is more than two orders of magnitude lower than the overall wall time. On the other hand, however, the blockchain overhead is more significant: it takes about $10 \%$ out of the overall running time. While it can be argued that a $10 \%$ training overhead is acceptable to trade for the highly desired auditing property, we believe that one
of the most interesting future research directions in this area is to explore more efficient consensus protocols.

### 4.6 Summary

This chapter demonstrates that the collusion of malicious clients can share a single model with deliberate Gaussian noises and launch a new type of model-poisoning attack on the recently proposed tokenized incentive for federated machine learning systems. We show that the new attack cannot be effectively detected and in fact, can be proven to converge with well-controlled parameters. To that end, we propose a blockchain-based protocol to efficiently track the local models submitted by the clients to audit their suspicious activities. We have evaluated the new attack and the auditing protocol with four popular data sets, two widely used FL aggregation algorithms, and on a broad range of clients between 50 and 500. Experimental results show that the proposed Gaussian attacks are effective with reasonable variance and the overhead of blockchain-based auditing services incurs acceptable training overhead at about $10 \%$.

## CHAPTER 5

## CONCLUSIONS AND FUTURE WORKS

### 5.1 Conclusions

In this dissertation, we have focused on investigating the impact of efficient distributed computing concepts to long-standing issues in data-intensive scientific applications, applying homomorphic encryption to outsourced databases and improving the security of tokenized incentive mechanisms for Federated Learning (FL) systems. The advances provided in this dissertation make a substantial contribution to the discipline, paving the way for improved data processing, enhanced privacy and security, and the creation of novel solutions to deal with the ever-increasing data explosion.

In Chapter 2, we took an algebraic-topological approach to model the parametrization of microelectrode arrays (MEAs) that involves computationally intensive Kirchhoff laws. We then developed a system prototype called Parma which is based on the algebraictopological model and evaluated the performance in a high-performance computing setting. Experimental results showed that the Parma the state-of-the-art in time, scalability and memory usage.

In Chapter 3, we introduced the concept of parallel caching of secure aggregation to mitigate the performance overhead incurred by the homomorphic encryption module when applied to data-intensive workloads that are very common for outsourced databases, or database-as-a-serve in cloud computing. We design a new algorithm called RHE, conduct
a thorough analysis of its parameterization and design a full-fledged-protocol called Rache which adopts RHE to securely encrypt a large volume of data and data streams. Experimental results on six workloads show that the proposed caching significantly improves the performance of state-of-the-art homomorphic encryption schemes.

In Chapter 4, we addressed an important issue in federated learning: the security of tokenized incentive mechanisms for participating clients. A blockchain-based auditing framework was designed to minimize Gaussian assaults and protect the fairness and reliability of FL systems. The protocol's effectiveness was proven by extensive evaluation with several FL aggregation algorithms, diverse datasets, and varying scales. FL systems can assure the integrity and trustworthiness of tokenized reward mechanisms by leveraging the security features provided by blockchain, thereby promoting a more safe and resilient distributed learning environment.

### 5.2 Future works

Our future work along the line of MEA research is threefold. Firstly, we will extend the proposed approach into a cluster of heterogeneous nodes. Secondly, we plan to develop a GPU version of Parma so that the massive number of GPU cores can be exploited. Finally, we are also planing to re-implement both the baseline system and the proposed parallelization techniques with low-level programming language like C or $\mathrm{C}++$ in order to explore more opportunities for performance improvement.

In addtion, our future work will focus on integrating the developed radix-based caching into scientific blockchains [3, 4] such that sensitive scientific data can be shared and ver-
ified among the collaborators confidentially. One orthogonal optimization in this context will be to exploit the specific data format used in scientific workflows [79, 110] and array databases [97, [104]. We also plan to apply radix caching in federated learning [78] to improve the performance of encoding local gradient updates.

Finally, our future work will focus on exploring more efficient consensus protocols to alleviate the significant overhead incurred by the proposed blockchain-based auditing. We will also focus on extending our proposed blockchain-based auditing algorithm or propose new algorithms to defend against a broader range of FL attacks.

## BIBLIOGRAPHY

[1] Abdullah Al-Mamun, Jun Dai, Xiaohua Xu, Mohammad Sadoghi, Haoting Shen, and Dongfang Zhao. Consortium blockchain for the assurance of supply chain security. In 27th Annual Network and Distributed System Security Symposium (NDSS), 2020.
[2] Abdullah Al-Mamun, Haoting Shen, and Dongfang Zhao. Dean: A lightweight and resource-efficient blockchain protocol for reliable edge computing. In IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2022.
[3] Abdullah Al-Mamun, Feng Yan, and Dongfang Zhao. BAASH: Lightweight, efficient, and reliable blockchain-as-a-service for hpc systems. In International Conference on High Performance Computing, Networking, Storage and Analysis (SC), 2021.
[4] Abdullah Al-Mamun, Feng Yan, and Dongfang Zhao. SciChain: Blockchainenabled lightweight and efficient data provenance for reproducible scientific computing. In IEEE 37th International Conference on Data Engineering (ICDE), 2021.
[5] Scott Alfeld, Xiaojin Zhu, and Paul Barford. Data poisoning attacks against autoregressive models. In Proceedings of the AAAI Conference on Artificial Intelligence, 2016.
[6] Md Azahar Ali, Chunshan Hu, Sanjida Jahan, Bin Yuan, Mohammad Sadeq Saleh, Enguo Ju, Shou-Jiang Gao, and Rahul P Panat. Sensing of covid-19 antibodies in seconds via aerosol jet printed three dimensional electrodes. medRxiv, 2020.
[7] R. Amirulah, S. Z. M. Muji, M. H. Jabbar, R. A. Rahim, and M. H. F. Rahiman. Digitalization of linear back projection algorithm for fpga implementation. In 2016 IEEE Conference on Systems, Process and Control (ICSPC), 2016.
[8] Prabhanjan Ananth, Aloni Cohen, and Abhishek Jain. Cryptography with updates. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, Advances in Cryptology EUROCRYPT 2017, pages 445-472, Cham, 2017. Springer International Publishing.
[9] Sanjeev Arora and Boaz Barak. Computational Complexity: A Modern Approach. Cambridge University Press, USA, 1st edition, 2009.
[10] Augustus: Gene prediction. https://github.com/Gaius-Augustus/Augustus, Accessed 2022.
[11] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics, pages 2938-2948, Palermo, Sicily, Italy, August 2020. PMLR.
[12] Anatoly Bakushinsky and A. Goncharsky. Ill-posed problems: theory and applications. Springer Science and Business Media, 2012.
[13] Mihir Bellare, Oded Goldreich, and Shafi Goldwasser. Incremental cryptography: The case of hashing and signing. In Yvo Desmedt, editor, Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings, volume 839 of Lecture Notes in Computer Science, pages 216-233. Springer, 1994.
[14] Mihir Bellare, Oded Goldreich, and Shafi Goldwasser. Incremental cryptography and application to virus protection. In Frank Thomson Leighton and Allan Borodin, editors, Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing (STOC), 1995.
[15] Ayoub Benaissa, Bilal Retiat, Bogdan Cebere, and Alaa Eddine Belfedhal. Tenseal: A library for encrypted tensor operations using homomorphic encryption, 2021.
[16] Bernard Marr. https://bernardmarr.com/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read/, Accessed 2023.
[17] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. In International Conference on Machine Learning, pages 634-643. PMLR, 2019.
[18] Tarunpreet Bhatia, A.K. Verma, and Gaurav Sharma. Towards a secure incremental proxy re-encryption for e-healthcare data sharing in mobile cloud computing. Concurrency and Computation: Practice and Experience (CCPE), 32(5):e5520, 2020. e5520 CPE-18-0794.R1.
[19] Bitcoin Trade History. https://www.blockchain.com/charts/trade-volume, Accessed 2022.
[20] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In Proceed-
ings of the 31st International Conference on Neural Information Processing Systems, NIPS' 17, page 118-128, Red Hook, NY, USA, 2017. Curran Associates Inc.
[21] Alessio Paolo Buccino, Sheng-Hsiou Hsu, and Gert Cauwenberghs. Real-time spike sorting for multi-electrode arrays with online independent component analysis. In IEEE Biomedical Circuits and Systems Conference, 2018.
[22] W. Chang, C. Lin, S. Mu, L. Chen, C. Tsai, Y. Chiu, and M. C. . Chao. Generating routing-driven power distribution networks with machine-learning technique. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 36(8):1237-1250, 2017.
[23] Subodha Charles and Prabhat Mishra. Securing network-on-chip using incremental cryptography. In 2020 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), pages 168-175, 2020.
[24] Dawei Chen, Choong Seon Hong, Li Wang, Yiyong Zha, Yunfei Zhanga, Xin Liu, and Zhu Han. Matching-theory-based low-latency scheme for multitask federated learning in mec networks. IEEE Internet of Things Journal, 2021.
[25] Hang Chen, Syed Ali Asif, Jihong Park, Chien-Chung Shen, and Mehdi Bennis. Robust blockchained federated learning with model validation and proof-of-stake inspired consensus. arXiv preprint arXiv:2101.03300, 2021.
[26] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yong Soo Song. Homomorphic encryption for arithmetic of approximate numbers. In Tsuyoshi Takagi and Thomas Peyrin, editors, Advances in Cryptology - ASIACRYPT 2017-23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I, volume 10624 of Lecture Notes in Computer Science, pages 409-437. Springer, 2017.
[27] Covid-19 Data. https://covidtracking.com/data/download/nationalhistory.csv, Accessed 2022.
[28] Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. Multiparty computation from threshold homomorphic encryption. In Birgit Pfitzmann, editor, Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding, volume 2045 of Lecture Notes in Computer Science, pages 280-299. Springer, 2001.
[29] Ivan Damgård and Jesper Buus Nielsen. Universally composable efficient multiparty
computation from threshold homomorphic encryption. In Dan Boneh, editor, $A d$ vances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings, volume 2729 of Lecture Notes in Computer Science, pages 247-264. Springer, 2003.
[30] DataStax Java Driver. https://github.com/datastax/java-driver, Accessed 2022.
[31] M. Daum, B. Haynes, D. He, A. Mazumdar, and M. Balazinska. Tasm: A tile-based storage manager for video analytics. In 2021 IEEE 37th International Conference on Data Engineering (ICDE), pages 1775-1786, Los Alamitos, CA, USA, apr 2021. IEEE Computer Society.
[32] Leo de Castro, Rashmi Agrawal, Rabia Yazicigil, Anantha Chandrakasan, Vinod Vaikuntanathan, Chiraag Juvekar, and Ajay Joshi. Does fully homomorphic encryption need compute acceleration?, 2021.
[33] Elena della Valle and James D. Weiland. Simultaneous impedance measurements of the utah electrodes array: A finite element method analysis. In 9th International IEEE/EMBS Conference on Neural Engineering (NER), pages 819-822, 2019.
[34] Daniel Demmler, Thomas Schneider, and Michael Zohner. ABY - A framework for efficient mixed-protocol secure two-party computation. In 22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2015. The Internet Society, 2015.
[35] Yongheng Deng, Feng Lyu, Ju Ren, Yi-Chao Chen, Peng Yang, Yuezhi Zhou, and Yaoxue Zhang. Fair: Quality-aware federated learning with precise user incentive and model aggregation. In IEEE INFOCOM 2021-IEEE Conference on Computer Communications, 2021.
[36] Yarkın Doroz, Erdinc Ozturk, and Berk Sunar. Accelerating fully homomorphic encryption in hardware. IEEE Transactions on Computers, 64(6):1509-1521, 2015.
[37] David S. Dummit and Richard M. Foote. Abstract algebra. John Wiley, 1999.
[38] Dmitry Duplyakin, Robert Ricci, Aleksander Maricq, Gary Wong, Jonathon Duerig, Eric Eide, Leigh Stoller, Mike Hibler, David Johnson, Kirk Webb, Aditya Akella, Kuangching Wang, Glenn Ricart, Larry Landweber, Chip Elliott, Michael Zink, Emmanuel Cecchet, Snigdhaswin Kar, and Prabodh Mishra. The design and operation of CloudLab. In Proceedings of the USENIX Annual Technical Conference (ATC), pages 1-14, July 2019.
[39] T. Elgamal. A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory, 31(4):469-472, 1985.
[40] Federico Esposti, Jacopo Lamanna, and Maria Gabriella Signorini. A new approach to the spatio-temporal pattern identification in neuronal multi-electrode registrations. In Proceedings of Neuroscience Today, pages 21-24, 2007.
[41] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Paper 2012/144, 2012. https: //eprint.iacr.org/2012/144.
[42] Yann Fraboni, Richard Vidal, and Marco Lorenzi. Free-rider attacks on model aggregation in federated learning. In International Conference on Artificial Intelligence and Statistics, pages 1846-1854. PMLR, 2021.
[43] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing, STOC '09, pages 169-178, 2009.
[44] Peter Giblin. Graphs, Surfaces and Homology. Cambridge University Press, 2010.
[45] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing, STOC '87, page 218-229, New York, NY, USA, 1987. Association for Computing Machinery.
[46] Rachid Guerraoui, Maurice Herlihy, and Bastian Pochon. A topological treatment of early-deciding set-agreement. Theor. Comput. Sci., 410(6-7):570-580, 2009.
[47] Hakan Hacigümüş, Bala Iyer, Chen Li, and Sharad Mehrotra. Executing sql over encrypted data in the database-service-provider model. In Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, SIGMOD '02, page 216-227, New York, NY, USA, 2002. Association for Computing Machinery.
[48] Shai Halevi and Victor Shoup. Bootstrapping for helib. J. Cryptol., 34(1), jan 2021.
[49] Allan Hambley. Electrical Engineering: Principles and Applications. Pearson; 7th edition, 2017.
[50] Jingoo Han, Ahmad Faraz Khan, Syed Zawad, Ali Anwa, Nathalie Baracaldo Angel, Yi Zhou, Feng Yan, and Ali R. Butt. Tokenized incentive for federated learning. In Proceedings of the AAAI Conference on Artificial Intelligence, 2022.
[51] Allen Hatcher. Algebraic topology. Cambridge Univ. Press, Cambridge, 2000.
[52] Brandon Haynes, Maureen Daum, Dong He, Amrita Mazumdar, Magdalena Balazinska, Alvin Cheung, and Luis Ceze. Vss: A storage system for video analytics. In Proceedings of the 2021 International Conference on Management of Data, SIGMOD/PODS '21, page 685-696, 2021.
[53] Chaoyang He, Songze Li, Jinhyun So, Mi Zhang, Hongyi Wang, Xiaoyang Wang, Praneeth Vepakomma, Abhishek Singh, Hang Qiu, Li Shen, Peilin Zhao, Yan Kang, Yang Liu, Ramesh Raskar, Qiang Yang, Murali Annavaram, and Salman Avestimehr. Fedml: A research library and benchmark for federated machine learning. Advances in Neural Information Processing Systems, Best Paper Award at Federate Learning Workshop, 2020.
[54] HElib. https://github.com/shaih/HElib, Accessed January 18, 2016.
[55] Maurice Herlihy and Nir Shavit. The asynchronous computability theorem for tresilient tasks. In Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing (STOC), pages 111-120, 1993.
[56] Maurice Herlihy and Nir Shavit. A simple constructive computability theorem for wait-free computation. In Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing (STOC), pages 243-252, 1994.
[57] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In Joe Buhler, editor, Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998, Proceedings, volume 1423 of Lecture Notes in Computer Science, pages 267-288. Springer, 1998.
[58] W. Kuan Hon and Christopher Millard. Banking in the cloud: Part 3 - contractual issues. Computer Law E Security Review, 34(3):595-614, 2018.
[59] Human Genome Databases. http://hgdownload.soe.ucsc.edu/goldenPath/ hg38/database/, Accessed 2022.
[60] Hyperledger. https://www.hyperledger.org/, Accessed 2020.
[61] Vandana Jain and K. Muralidhar. Electrowetting-on-dielectric system for covid-19 testing. Transactions of the Indian National Academy of Engineering, May 2020.
[62] Tehsin Kanwal, Adeel Anjum, and Abid Khan. Privacy preservation in e-health
cloud: taxonomy, privacy requirements, feasibility analysis, and opportunities. Clust. Comput., 24(1):293-317, 2021.
[63] Gang Ke, Shi Wang, and Huan-huan Wu. Parallel incremental attribute-based encryption for mobile cloud data storage and sharing. Journal of Ambient Intelligence and Humanized Computing, pages 1-11, 012021.
[64] Marcel Keller, Emmanuela Orsini, and Peter Scholl. Mascot: Faster malicious arithmetic secure computation with oblivious transfer. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS '16, page 830-842, New York, NY, USA, 2016. Association for Computing Machinery.
[65] Latif U. Khan, Shashi Raj Pandeya, Nguyen H. Tran, Walid Saad, Zhu Han, Minh NH Nguyen, and Choong Seon Hong. Federated learning for edge networks: Resource optimization and incentive mechanism. IEEE Communications Magazine, 2020.
[66] Louiza Khati and Damien Vergnaud. Analysis and improvement of an authentication scheme in incremental cryptography. In Carlos Cid and Michael J. Jacobson Jr., editors, Selected Areas in Cryptography - SAC 2018-25th International Conference, Calgary, AB, Canada, August 15-17, 2018, Revised Selected Papers, volume 11349 of Lecture Notes in Computer Science, pages 50-70. Springer, 2018.
[67] P. Kiele, A. Kohler, C. Pasluosta, and T. Stieglitz. Robust and precise alignment monitoring of electrode arrays for capacitive energy supply and signal transmission. In 9th International IEEE/EMBS Conference on Neural Engineering (NER), pages 686-689, 2019.
[68] Eun Kim, Geza Erdos, Shaohua Huang, Thomas W. Kenniston, Stephen C. Balmert, Cara Donahue Carey, V. Stalin Raj, Michael W. Epperly, William B. Klimstra, Bart L. Haagmans, Emrullah Korkmaz, Louis D. Falo, and Andrea Gambotto. Microneedle array delivered recombinant coronavirus vaccines: Immunogenicity and rapid translational development. EBioMedicine, 55:102743, 2020.
[69] Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. Blockchained ondevice federated learning. IEEE Communications Letters, 24(6):1279-1283, 2019.
[70] Yejin Kim, Jimeng Sun, Hwanjo Yu, and Xiaoqian Jiang. Federated tensor factorization for computational phenotyping. In Proceedings of the 23rd ACM SIGKDD International conference on knowledge discovery and data mining, pages 887-895, 2017.
[71] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.
[72] Avinash Lakshman and Prashant Malik. Cassandra: A decentralized structured storage system. SIGOPS Oper. Syst. Rev., 44(2), April 2010.
[73] M. M. Lavrentev, V. G. Romanov, and Serge P. S. Ill-posed problems of mathematical physics and analysis. American Mathematical Society, 1986.
[74] H. Li, Z. Tian, J. Xu, R. K. V. Maeda, Z. Wang, and Z. Wang. Chip-specific power delivery and consumption co-management for process-variation-aware manycore systems using reinforcement learning. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 28(5):1150-1163, 2020.
[75] Li Li, Yuxi Fan, Mike Tse, and Kuo-Yi Lin. A review of applications in federated learning. Computers $\mathcal{E}$ Industrial Engineering, 149:106854, 2020.
[76] Fenglin Liu, Xian Wu, Shen Ge, Wei Fan, and Yuexian Zou. Federated learning for vision-and-language grounding problems. In Proceedings of the AAAI Conference on Artificial Intelligence, pages 11572-11579, 2020.
[77] M. H. Loke, P. B. Wilkinson, and J. E. Chambers. Fast computation of optimized electrode arrays for 2d resistivity surveys. Computers $\mathcal{E}$ Geosciences, 36(11), November 2010.
[78] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In Aarti Singh and Xiaojin (Jerry) Zhu, editors, Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA, volume 54 of Proceedings of Machine Learning Research, pages 1273-1282. PMLR, 2017.
[79] Parmita Mehta, Sven Dorkenwald, Dongfang Zhao, Tomer Kaftan, Alvin Cheung, Magdalena Balazinska, Ariel Rokem, Andrew Connolly, Jacob Vanderplas, and Yusra AlSayyad. Comparative evaluation of big-data systems on scientific image analytics workloads. In Proceedings of the 43rd International Conference on Very Large Data Bases (VLDB), 2017.
[80] Gonzalo E. Mena, Lauren E. Grosberg, Sasidhar Madugula, Paweł Hottowy, Alan Litke, John Cunningham, E. J. Chichilnisky, and Liam Paninski. Electrical stimulus artifact cancellation and neural spike detection on large multi-electrode arrays. PLoS computational biology, 13(11), November 2017.
[81] Ralph C. Merkle. A digital signature based on a conventional encryption function. In Carl Pomerance, editor, Advances in Cryptology - CRYPTO '87, pages 369-378, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg.
[82] Ilya Mironov, Omkant Pandey, Omer Reingold, and Gil Segev. Incremental deterministic public-key encryption. In David Pointcheval and Thomas Johansson, editors, Advances in Cryptology - EUROCRYPT 2012, pages 628-644, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
[83] MNIST Dataset. http://yann.lecun.com/exdb/mnist/, Accessed 2020.
[84] Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacypreserving machine learning. In 2017 IEEE Symposium on Security and Privacy (SP), pages 19-38, 2017.
[85] MPI. https://www.mpi-forum.org/docs/, Accessed 2019.
[86] MPI4PY. https://mpi4py.readthedocs.io/en/stable/intro.html, Accessed 2021.
[87] National Institute and Technology of Standards. Advanced encryption standard. NIST FIPS PUB 197, 2001.
[88] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In NIPS Workshop on Deep Learning and Unsupervised Feature Learning, 2011.
[89] Y. Niu, A. Al-Mamun, H. Lin, T. Li, Y. Zhao, and D. Zhao. Toward scalable analysis of multidimensional scientific data: A case study of electrode arrays. In IEEE International Conference on Big Data, 2018.
[90] Ye Niu, Lin Qi, Fen Zhang, and Yi Zhao. Geometric screening of core/shell hydrogel microcapsules using a tapered microchannel with interdigitated electrodes. Biosensors and Bioelectronics, 112:162-169, 2018.
[91] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Proceedings of the 17th International Conference on Theory and Application of Cryptographic Techniques, EUROCRYPT'99, page 223-238, Berlin, Heidelberg, 1999. Springer-Verlag.
[92] Antonis Papadimitriou, Ranjita Bhagwan, Nishanth Chandran, Ramachandran Ramjee, Andreas Haeberlen, Harmeet Singh, Abhishek Modi, and Saikrishna Badri-
narayanan. Big data analytics over encrypted datasets with seabed. In Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation (OSDI), page 587-602, USA, 2016. USENIX Association.
[93] Rishabh Poddar, Tobias Boelter, and Raluca Ada Popa. Arx: An encrypted database using semantically secure encryption. Proc. VLDB Endow., 12(11):1664-1678, 2019.
[94] Raluca Ada Popa, Catherine Redfield, Nickolai Zeldovich, and Hari Balakrishnan. Cryptdb: protecting confidentiality with encrypted query processing. In Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles, pages 85100. ACM, 2011.
[95] PyTorch. https://pytorch.org/, Accessed 2022.
[96] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing, STOC '89, page 73-85, New York, NY, USA, 1989. Association for Computing Machinery.
[97] Rasdaman. http://www.rasdaman.org/, Accessed 2021.
[98] Dayane Reis, Jonathan Takeshita, Taeho Jung, Michael Niemier, and Xiaobo Sharon Hu. Computing-in-memory for performance and energy-efficient homomorphic encryption. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 28(11):2300-2313, 2020.
[99] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. Commиn. ACM, 21(2):120-126, feb 1978.
[100] Bita Darvish Rouhani, M. Sadegh Riazi, and Farinaz Koushanfar. Deepsecure: Scalable provably-secure deep learning. In Proceedings of the 55th Annual Design Automation Conference, DAC '18, New York, NY, USA, 2018. Association for Computing Machinery.
[101] Nikola Samardzic, Axel Feldmann, Aleksandar Krastev, Srinivas Devadas, Ronald Dreslinski, Christopher Peikert, and Daniel Sanchez. F1: A Fast and Programmable Accelerator for Fully Homomorphic Encryption, page 238-252. Association for Computing Machinery, 2021.
[102] Savvas Savvides, Darshika Khandelwal, and Patrick Eugster. Efficient confidentiality-preserving data analytics over symmetrically encrypted datasets. Proc. VLDB Endow., 13(8):1290-1303, April 2020.
[103] F. Schmuck and R. Haskin. Gpfs: A shared-disk file system for large computing clusters. In Proceedings of the 1st USENIX Conference on File and Storage Technologies (FAST), 2002.
[104] SciDB. https://github.com/Paradigm4/SciDB, Accessed 2021.
[105] Microsoft SEAL (release 3.7). https://github.com/Microsoft/SEAL, September 2021. Microsoft Research, Redmond, WA.
[106] Adi Shamir. How to share a secret. Commun. ACM, 22(11):612-613, nov 1979.
[107] Muhammad Shayan, Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. Biscotti: A blockchain system for private and secure federated learning. IEEE Transactions on Parallel and Distributed Systems, 32(7):1513-1525, 2020.
[108] Virat Shejwalkar and Amir Houmansadr. Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning. In Network and Distributed Systems Security (NDSS) Symposium 2021, 2021.
[109] Haoting Shen, Shahriar Badsha, and Dongfang Zhao. Consortium blockchain for the assurance of supply chain security. In 27th Annual Network and Distributed System Security Symposium (NDSS), 2020.
[110] Tong Shu, Yanfei Guo, Justin Wozniak, Xiaoning Ding, Ian Foster, and Tahsin Kurc. Bootstrapping in-situ workflow auto-tuning via combining performance models of component applications. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC), 2021.
[111] Julian Shun. Practical parallel hypergraph algorithms. In Proceedings of the 25th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, PPoPP '20, page 232-249, New York, NY, USA, 2020. Association for Computing Machinery.
[112] statistica. https://www.statista.com/statistics/871513/worldwide-data-created/, Accessed 2023.
[113] J. Sun, W. Tian, H. Che, S. Sun, S. Gao, L. Xu, and W. Yang. Proportional-integral controller modified landweber iterative method for image reconstruction in electrical capacitance tomography. IEEE Sensors Journal, 2019.
[114] Jiawen Sun, Hans Vandierendonck, and Dimitrios S. Nikolopoulos. A new approach to the spatio-temporal pattern identification in neuronal multi-electrode registrations.

In Proceedings of the 24th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP), pages 391-392, 2019.
[115] Jingwei Sun, Ang Li, Louis DiValentin, Amin Hassanzadeh, Yiran Chen, and Hai Li. Fl-wbc: Enhancing robustness against model poisoning attacks in federated learning from a client perspective. Advances in Neural Information Processing Systems, 2021.
[116] C. Tan, S. Lv, F. Dong, and M. Takei. Image reconstruction based on convolutional neural network for electrical resistance tomography. IEEE Sensors Journal, 19(1):196-204, 2019.
[117] TPC-H 3.0.0. http://tpc.org/tpc ${ }_{d}$ ocuments ${ }_{c}$ urrent ${ }_{v}$ ersions/ currentspecifications5.asp, Accessed 2022.
[118] M. Vauhkonen, D. Vadasz, P. A. Karjalainen, E. Somersalo, and J. P. Kaipio. Tikhonov regularization and prior information in electrical impedance tomography. IEEE Transactions on Medical Imaging, 17(2):285-293, 1998.
[119] Fenghe Wang, Junquan Wang, and Wenfeng Yang. Efficient incremental authentication for the updated data in fog computing. Future Generation Computer Systems (FGCS), 114:130-137, 2021.
[120] Jianhua Wang, Xiaolin Chang, Ricardo J Rodrìguez, and Yixiang Wang. Assessing anonymous and selfish free-rider attacks in federated learning. In 2022 IEEE Symposium on Computers and Communications (ISCC), pages 1-6. IEEE, 2022.
[121] Xinying Wang, Olamide Timothy Tawose, Feng Yan, and Dongfang Zhao. HDK: toward high-performance deep-learning-based kirchhoff analysis. In The ThirtyFourth AAAI Conference on Artificial Intelligence (AAAI), pages 997-1004, 2020.
[122] Maoqiang Wu, Dongdong Ye, Jiahao Ding, Yuanxiong Guo, Rong Yu, and Miao Pan. Incentivizing differentially private federated learning: A multidimensional contract approach. IEEE Internet of Things Journal, 2021.
[123] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.
[124] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. Dba: Distributed backdoor attacks against federated learning. In International Conference on Learning Representations, 2020.
[125] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. ACM Trans. Intell. Syst. Technol., 10(2), jan 2019.
[126] Andrew C. Yao. Protocols for secure computations. In 23rd Annual Symposium on Foundations of Computer Science, pages 160-164, 1982.
[127] Yahya H. Yassin, Catthoor Francky, Kloosterman Fabian, Jyh-Jang Sun, JoãO Couto, Per Gunnar Kjeldsberg, and Nick Van Helleputte. Algorithm/architecture co-optimisation technique for automatic data reduction of wireless read-out in highdensity electrode arrays. ACM Transactions on Embedded Computing Systems (TECS), 17(3), June 2018.
[128] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. Byzantinerobust distributed learning: Towards optimal statistical rates. In International Conference on Machine Learning, 2018.
[129] Zhebin Zhang, Dajie Dong, Yuhang Ma, Yilong Ying, Dawei Jiang, Ke Chen, Lidan Shou, and Gang Chen. Refiner: A reliable incentive-driven federated learning system powered by blockchain. Proceedings of the VLDB Endowment, 14(12):26592662, 2021.
[130] Yiyuan Zhao, Srijata Chakravorti, Robert F. Labadie, Benoit M. Dawant, and Jack H. Noble. Automatic graph-based method for localization of cochlear implant electrode arrays in clinical ct with sub-voxel accuracy. Medical image analysis, 52, February 2019.
[131] Zhaohua Zheng, Yize Zhou, Yilong Sun, Zhang Wang, Boyi Liu, and Keqiu Li. Applications of federated learning in smart cities: recent advances, taxonomy, and open challenges. Connection Science, 34(1):1-28, 2022.
[132] Xiaojie Zhu, Erman Ayday, Roman Vitenberg, and Narasimha Raghavan Veeraragavan. Privacy-preserving search for a similar genomic makeup in the cloud. IEEE Transactions on Dependable and Secure Computing, 2021.
[133] Sabine Zips, Leroy Grob, Philipp Rinklin, Korkut Terkan, Nouran Yehia Adly, Lennart Jakob Konstantin Weiß, Dirk Mayer, and Bernhard Wolfrum. Fully printed u-needle electrode array from conductive polymer ink for bioelectronic applications. ACS Applied Materials $\mathcal{E}$ Interfaces, 11(36):32778-32786, 2019.


[^0]:    ${ }^{1}$ Namely, both $\emptyset$ and $S$ are in $\mathcal{T}$, a finite number of intersections of elements in $\mathcal{T}$ is in $\mathcal{T}$, and any union of elements in $\mathcal{T}$ is in $\mathcal{T}$.

[^1]:    ${ }^{2}$ There is a more formal definition of the boundary operation in algebraic topology; we do not mention it

[^2]:    ${ }^{3}$ We use $c$ to denote a constant number.

[^3]:    ${ }^{4}$ Or, equivalently, vertices in the original array.

[^4]:    5 https://github.com/classner/pymp

[^5]:    ${ }^{6}$ https://github.com/Taotopps2006/ParMA

[^6]:    ${ }^{1}$ For simplicity, we use Rache to indicate incremental Rache in this section.

