Privacy-preserving Data Analysis Techniques for Biomedical Data

by

Md. Monowar Anjum

A thesis submitted to
The Faculty of Graduate Studies of
The University of Manitoba
in partial fulfillment of the requirements
of the degree of

Master of Science

Department of Computer Science
The University of Manitoba
Winnipeg, Manitoba, Canada
May 2022

Thesis advisor

Noman Mohammed

Author

Md. Monowar Anjum

Privacy-preserving Data Analysis Techniques for Biomedical Data

Abstract

Privacy is a fundamental aspect of modern distributed systems. The data collection mechanism and subsequent analysis often reveals private information about individuals. This is especially true when designing contact tracing systems to combat a pandemic. Contact tracing systems collect vital information about individuals such as their social interaction graph, their frequently visited places, and other sensitive information. Majority of the proposed systems use centralized architecture and population wide deployment. Such macro-level design perspective is prone to privacy and scalability issues. In the first part of the thesis, we address the problems in recently proposed contact tracing systems. We propose a micro-level system design instead of a macro level system design. We propose a system that can be implemented at organizational level and can be scaled without any steep infrastructure cost. Privacy considerations are baked into the system design. The system only stores strictly necessary information from the user and the data never leaves the organization premises. Our proposed system can be scaled up rapidly without the requirement of population wide adoption.

Subsequent data analysis from the aggregate statistics of the raw data collected by our proposed system is performed in a privacy-preserving manner. In the field of epidemiology and clinical modeling, summary of raw biomedical data are used to fit or train disease-specific specialized models. Generalized Linear Mixed Model is one such widely used model. Training such models on sensitive data in a collaborative setting often entails privacy risks. Standard privacy preserving mechanisms such as differential privacy can be used to mitigate the privacy risk during training the model. However, experimental evidence suggests that adding differential privacy to the training of the model can cause significant utility loss which makes the model impractical for real world usage. Therefore, it becomes clear that generalized linear mixed models which lose their usability under differential privacy requires a different approach for privacy preserving model training.

In the second part of the thesis, we propose a value-blind training method in a federated setting for generalized linear mixed models. In our proposed training method, the central server optimizes model parameters without ever getting access to the raw training data or intermediate computation values. Intermediate computation values that are shared by the collaborating parties with the central server are encrypted using homomorphic encryption. We formally prove the security of our proposed model. Experimentation on multiple datasets suggests that the model trained by our proposed method achieves very low error rate while preserving privacy. To the best of our knowledge, this is the first work that performs a systematic privacy analysis of generalized linear mixed model training in federated setting.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to begin by thanking my advisor, my committee, my parents, my wife, and all the people who have supported me along the way. I started my graduate study at the early onset of COVID-19 pandemic and it was an incredibly difficult time for me. I am eternally grateful for the guidance by my advisor and mentors to help me navigate the academia during the difficult time.

This thesis is dedicated to parents and my wife. Without them, I would not have come this far.

Chapter 1

Introduction

Proliferation of distributed systems have given rise to a data-centric society. A lot of activities we perform on day-to-day basis rely on digital platforms and data collection is an inherent feature of these platforms. The usual working principles of these systems are that they collect a lot of data from users by their collection systems and afterwards, they analyze that data to extract value. Therefore, broadly speaking, majority of commercial distributed systems can be segmented into two sections: data collection and data analysis. To prevent sensitive personal identifiable information (PII) leak and corporate surveillance, it is imperative that privacy is taken into account in the design of both phases of the system. Consequently, privacy-aware data collection and privacy-preserving data analysis are two of the most prominent research directions in recent times. In the first part of this thesis, we propose a privacy-aware data collection system for indoor contact tracing. In the later part, we design a privacy-preserving data analysis model to learn insights from the collected data.

Motivation for Privacy-Preserving Data Collection. Pandemics often spread by contact events and therefore, require contact tracing by government or public health officials to mitigate the spread of the pandemic. Contact tracing system is an umbrella term for a range of techniques that are used by health-care officials to trace the individuals who came in contact with an infected person within a specific time frame. Digital contact tracing is the latest addition to these techniques. It refers to the idea of tracing people's interaction (i.e., contact events) with infected person via digital devices (i.e., smartphones, tabs etc.). Such systems require huge investments in infrastructure (i.e., storage and software) and widespread adoption to be successful. Basic working principles of these systems follow the pattern of large scale real-time data collection and subsequent post processing. Singapore was the first nation to adopt such technology to curb the spread of Covid-19. Some other prominent instances are South Korea, China and India. In most of these deployed systems, the users were to exchange some kind of pseudo-random tokens (i.e., strings of unicode characters) via Bluetooth of their smartphone or mobile devices. These tokens would be later used to determine whether the user in question have been in contact with someone who were positively diagnosed for the disease.

Development of a large scale data collection system such as contact tracing comes with its own set of challenges. Especially, the privacy of the individual data contributors must be preserved. Any leakage of information can lead to severe consequences for the individual involved. For instance, the data leak from the South Korea's contact tracing system led to people being ostracized from their close friends and had

adverse effect on their interpersonal relationships. In the words of Lee Su-Young, A Psychiatrist at Myongji Hospital in South Korea, "*[Some of my patients] were more afraid of being blamed than dying of the virus*" [1]. This quote portrays that social stigma due to perceived privacy loss can have adverse effect on interpersonal dynamics. Therefore, it is imperative to bake privacy in the design of any contact tracing system. The privacy requirements are two fold in this case. Firstly, the system will collect only strictly "necessary and sufficient" information about the users. It will not collect irrelevant but available information to extract additional knowledge from the interaction graph of the users. Secondly, the collected data should go through privacy-preserving data analysis. At any point of the data analysis, sensitive information about individuals should not be revealed to any stakeholder. The second privacy requirement brings us to the motivation of the second part of this thesis.

Motivation for Privacy-Preserving Data Analysis. The data collected by any contact tracing system is protected by HIPAA and therefore, can not leave the premises of the organizations. However, statistics from the raw data can be leveraged for analysis. Therefore, we require to design an analysis scheme for geographically distributed summary statistics of raw data. This is achieved by leveraging Federated Learning (FL) process.

Federated Learning is a machine learning (ML) training paradigm which allows multiple parties to train a ML model without having to share their data. A central server synchronizes this collaboration among parties. Models trained in federated setting

are more robust compared to centrally trained models. However, FL is not completely private [2, 3]. Prior works have shown that malicious collaborating party or server or can compromise the data privacy of honest collaborating parties. Therefore, federated training often takes place in conjunction with different other privacy-preserving mechanism (i.e., differential privacy, homomorphic encryption) [4, 5, 6]. This is an active research area and a large body of work is available in literature which investigates various aspects and use cases of privacy-preserving federated learning. We noted during literature review that majority of the prior works in privacy-preserving federated learning demonstrated the effectiveness of their methods on a few specific class of models such as deep neural network, support vector machine, decision tree and regression [7, 8, 9]. This trend is understandable from a comparison perspective as it simplifies the benchmarking process among the proposed methods. Consequently, some popular machine learning models such as distribution modeling by maximum likelihood estimation, Bayesian optimization are mostly absent from the current literature. Some of these absent models are widely used in domain specific cases such as infectious disease modeling in epidemiology [10]. Therefore, privacy analysis of these overlooked models is an interesting research question.

One such overlooked model is Generalized Linear Mixed Model (GLMM). It is one of the most common statistical models used in medical and epidemiology research. For instance, if a new diagnostic test is designed for rapid Covid-19 testing, the efficacy of the test is determined by a GLMM over the data from clinical trials [11]. The wide adoption of GLMM can be attributed to the interpretability of maximum likelihood

estimation [12]. Although prevalent in medical and epidemiology research, GLMM has wide applicability in other disciplines as well [12]. To the best of our knowledge no prior work have performed a systematic privacy analysis of GLMM models. This acted as the primary motivation for second part of this thesis.

Objective. Based on the background and motivation, our objective in this thesis is to design an end-to-end privacy preserving data collection and analysis pipeline for structured biomedical data. The research challenges to overcome for achieving this objective are:

- How to design a contact tracing system that preserves the data locally? Every action in this system should go through privacy-preserving primitives such as private set intersection, secure multi-party computation or similar cryptographic primitives.
- How to formulate a federated learning algorithm that works on private data? By private data, we refer to homomorphically encrypted data. Subsequently, how to design a training system to implement and evaluate the algorithm performance?

Contribution. In the first part of this thesis we propose the design of a privacy-aware indoor contact tracing system. It can be implemented at organizational level without substantial change in the existing infrastructure. It ensures the data privacy in two ways. Firstly, the collected data never leaves the storage. Secondly, the system can never know about data collected from the user unless the user self-reports the case of being infected. Our key design contributions are listed as follows:

- We designed PAARS (Privacy Aware Access Regulation System), an efficient indoor contact tracing system that provides strong privacy guarantee while considering existence of semi-honest adversary in our threat model. The semi-honest entity in our model does not have access to any sensitive information that will allow it to deduce social interaction graph of any user or link any of the information available to itself to any particular user without the user's explicit consent.

- Our system design for PAARS requires minimal infrastructure addition. It works on existing infrastructure available within the organization.

- We designed a novel probability calculation scheme to determine the positive diagnosis probability of the user. Our scheme uses the state of the art epidemiological models available in literature to calculate the disease exposure score of an user which is then used to calculate the probability of the user being positively diagnosed for the disease. The whole process for calculation of probability and query result release to the user is differentially private which preserves each user's privacy from any other users or attackers.

It is important to understand that due to the pandemic and lockdown, it was not feasible for us to implement PAARS and deploy in a closed setting to evaluate the performance. Therefore, we focused on the privacy-preserving data analysis part of contact tracing system design. Longitudinal literature review suggested that privacy-preserving data analysis requires the sharing of aggregate information from locally-stored data. Raw data should never leave the premises of the organizational boundary. This insight led us to work on privacy preserving data analysis on aggre-

gate data (i.e., summary statistics of raw data). For this purpose we chose Federated Learning as our privacy-preserving data analysis method.

In the second part of the thesis, we proposed and evaluated a framework for privacy-preserving GLMM training. The reason of choosing GLMM is two-fold. Firstly, GLMM is widely used in medical domain and therefore, has direct relevance to epidemiological data analysis. Given that aggregate statistics from contact tracing data is primarily used for epidemiological modeling, GLMM is a natural choice. Secondly, to the best of our knowledge, the privacy aspect of GLMM training is not investigated in literature. Therefore, it is an interesting research question. The summary of the contributions from the second part is listed below:

- To the best of our knowledge, this is the first work that performs a systematic privacy analysis of federated training of GLMM.
- We propose and implement FedGLMM, a privacy-preserving scheme for GLMM training in federated setting. We leveraged homomorphic encryption and grid based parameter search technique for implementing FedGLMM.
- We prove the security of our proposed scheme in semi-honest trust setting.
- We evaluate the performance of FedGLMM over multiple datasets. Experimental results show that FedGLMM achieves less than 5% error-rate for distribution modeling tasks in both real world and synthetic datasets.

Organization. The thesis is organized in the following manner. Chapter 2 contains the background of privacy preserving techniques used in this thesis. Chapter 3 de-

scribes the literature review, design, and implementation strategy of PAARS. This is the indoor contact tracing system we designed for privacy preserving data collection. Chapter 4 describes FedGLMM, a federated learning algorithm for generalized linear mixed model training. We present a literature review, system design, security analysis and experimental evaluation for FedGLMM. Chapter 5 concludes the thesis.

Two publications have resulted from this work:

- **Anjum, Md Monowar**, and Noman Mohammed. "PAARS: Privacy aware access regulation system." 2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON). IEEE, 2020.
- **Anjum, Md Monowar**, Noman Mohammed, Wentao Li, and Xiaoqian Jiang. "Privacy preserving collaborative learning of generalized linear mixed model." Journal of Biomedical Informatics 127 (2022): 104008.

Chapter 2

Background

2.1 Differential Privacy

Differential privacy is a framework proposed by Dwork *et al.* to measure the privacy leakage of randomized algorithms [13]. It requires that two datasets differing by at most one data point produce statistically indistinguishable query result to an external viewer. An adversary observing the outputs of the queries can guess about the properties of the data points within a bounded probability. Formally, we have the following definition.

Definition *(Differential Privacy).* A randomized mechanism $\mathcal{M}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ satisfies (ϵ, δ) differential privacy if for any subset $S \in \mathcal{R}$ and for datasets $d, \bar{d}$ where $|d - \bar{d}| \leq 1$ the following inequality holds:

$$Pr[\mathcal{M}(d) \in S] \leq e^{\epsilon} Pr[\mathcal{M}(\bar{d}) \in S] + \delta \tag{2.1}$$

In our experimentation, we used the Laplace mechanism to ensure differential privacy.

Definition *(Laplace Mechanism).* Laplace mechanism is one of the most popular mechanism for adding differentially private noise to query results. Consider a function f defined on a dataset $\mathcal{X}$, which can be written as $f : \mathbb{N}^{|\mathcal{X}|} \longrightarrow \mathbb{R}^k$. This function f essentially maps queries on dataset $\mathcal{X}$ to k real numbers. Given such a function f, the Laplace mechanism $\mathcal{M}_L$ is defined as follows:

$$M_L(x, f(.), \epsilon) = f(x) + (Y_1, Y_2, \cdots, Y_k) \tag{2.2}$$

where $(Y_1, Y_2, \cdots, Y_k)$ are independently drawn random variables from a Laplace distribution with scale $(\frac{\Delta f}{\epsilon})$. Here Δf stands for the l_1-sensitivity of the function.

2.2 Homomorphic Encryption

Homomorphic encryption is a form of encryption that allows algebraic computation on encrypted data without the need to decrypt them first. Let us assume that S is a set of cleartext messages. $\mathcal{E}$ is an encryption scheme and $\mathcal{D}$ is the corresponding decryption scheme. E is considered *homomorphic* if for any message $m_1, m_2 \in S$ the following holds: $\mathcal{D}(\mathcal{E}(m_1) \odot \mathcal{E}(m_2)) = m_1 \odot m_2$ where $\odot$ is an algebraic operation. A lot of such homomorphic encryption scheme have been proposed and used for different kind of applications.

There are two distinct types of homomorphic encryption scheme: partially homomorphic encryption (PHE) and fully homomorphic encryption (FHE). In partial homomorphic encryption scheme only a few specific algebraic operations are possible

on encrypted data. For instance unpadded RSA and El-Gamal cryptosystem can compute multiplication over encrypted data. Paillier cryptosystem can compute addition operation over encrypted data. On the other hand, in fully homomorphic encryption scheme addition, multiplication as well as bitwise operations can be computed directly on encrypted data. Gentry *et al.* proposed GSW cryptosystem [14] which is a faster FHE scheme compared to the previous ones. In this work, we use Paillier cryptosystem described in [15] which facilitates addition and subtraction during the model training period.

2.3 Generalized Linear Mixed Model

This section contains the elements of statistical modeling that are required to understand the problem formulation and subsequent solutions presented to solve the problem. In this work, we will focus on a simplified but highly practical GLMM. This model determines whether a diagnostic test can be used to either confirm or rule out a disease. For the rest of the paper, we will use the term GLMM and model interchangeably. This subsection consists of the objectives, terminologies, assumptions and training operations (pre, during, post) of the model. Additionally, a high level summary of the model is presented at the end to help the reader gain an overview of the model.

Objective of the GLMM. In a clinical setting, the utility of a diagnostic test to confirm a disease or health condition is determined by Positive Predictive Value (PPV). The higher the PPV, the more confidence the clinicians have to confirm the

disease or the health condition based on that test's result. Similarly, the utility of a diagnostic test to rule out a disease or health condition (e.g. detecting covid-19 with a new test) is determined by Negative Predictive Value (NPV). The higher the NPV, the more confident the clinicians are in using that test's result to rule out that disease or health condition. The GLMM proposed by Chen *et al.* in [11], determines the PPV and NPV of a diagnostic test from the experiment data of multiple parties. In this paper, we will use this GLMM as reference to demonstrate our contributions.

Terminologies. Before we describe the GLMM specifications, let us define the terminologies used in the subsequent discussion.

- D_0, D_1 stand respectively for the event of not having the disease and having the disease (ground truth label).
- T_0, T_1 stand respectively for the event of being diagnosed negative and positive for the disease by the diagnostic test.
- π is the prevalence of the population sample on whom the diagnostic test is conducted ($\frac{D_1}{D_1+D_0}$).
- Se is the sensitivity of the population sample on whom the diagnostic test is conducted ($\frac{T_1 \bigcap D_1}{D_1}$).
- Sp is the specificity of the population sample on whom the diagnostic test is conducted ($\frac{T_0 \bigcap D_0}{D_0}$).
- n is the number of participants in the study.

Before Training GLMM. Let us assume that multiple parties intend to determine the level of effectiveness of a clinical diagnostic test T for detecting or ruling out the disease D. In other words, the parties want to know the PPV and NPV of test T for D. There are N parties involved in this process, each denoted as P_i where $i \in (1, 2, \cdots, N)$.

Each party P_i performs a study (i-th study) which consists of n_i participants. In each of these studies, the party uses diagnostic test T to detect the state of the disease D in each participant. After each party finishes their respective studies, they have access to the local information presented in table 2.1. This local table contains information regarding the state of the disease distribution in the test population. Previous studies have shown that information from such local table can be exploited to perform reconstruction attack that can leak personal information [16, 17]. Consequently, the information presented in this table is sensitive and parties do not want to share this information with the server or the other parties.

Table 2.1: Possible Data Outcomes for Study i

	D_1 (Disease)	D_0 (Non-Disease)
T_1 (Positive)	$n_{i_{11}}$	$n_{i_{10}}$
T_0 (Negative)	$n_{i_{01}}$	$n_{i_{00}}$
Total	n_{i_1}	n_{i_0}

Training the GLMM. Now, all the parties collaboratively want to compute the parameters they need to determine the PPV and NPV of the test T for disease D. In order to accomplish this without any local bias, they need to collaboratively build

a statistical model M which maximizes the log likelihood function in Equation 2.3.

$$\log\{L_C(\theta_0, \theta_1, \theta_2)\} = \log\{L_0(\theta_0)\} + \log\{L_1(\theta_1)\} + \log\{L_2(\theta_2)\} \tag{2.3}$$

where,

$$\log\{L_0(\theta_0)\} = \sum_{i=1}^{N} \int \binom{n}{n_1} {\pi_i}^{n_1} (1-\pi_i)^{n-n_1} \phi(\pi_i; \theta_0) d\pi_i \tag{2.4}$$

$$\log\{L_1(\theta_1)\} = \sum_{i=1}^{N} \int \binom{n_1}{n_{11}} {Se_i}^{n_{11}} (1-Se_i)^{n_1-n_{11}} \phi(Se_i; \theta_1) dSe_i \tag{2.5}$$

$$\log\{L_2(\theta_2)\} = \sum_{i=1}^{N} \int \binom{n_0}{n_{00}} {Sp_i}^{n_0} (1-Sp_i)^{n_0-n_{00}} \phi(Sp_i; \theta_2) dSp_i \tag{2.6}$$

Interested readers can refer to [11] for the derivation of these equations. For the current discussion purpose, let us denote the left hand side of the Equation 2.3 as final likelihood (FL) value. Each component of the right hand side of the Equation 2.3 is denoted as partial likelihood (PL) value. Let us annotate them as PL_0, PL_1 and PL_2 respectively. It is the central server's objective to find a set of parameters that maximize the value of PL_0, PL_1 and PL_2. Let us call these parameters as $\theta_0, \theta_1, \theta_2$[1]. In each training iteration, the server selects a new set of $\theta_0, \theta_1, \theta_2$ that aims to get higher PL values than previous iterations.

Equation 2.3 represents the core essence of any GLMM training. PL_0, PL_1 and PL_2 represents the log-likelihood of three distributions. For example, PL_0, PL_1 and PL_2 can represent the likelihood of a normal distribution, laplacian distribution and binomial distribution respectively. The GLMM is adding them up in Equation 2.3 which makes this a **linear** combination where data distributions are being **mixed**.

Let us show a concrete example of the training process. For the sake of simplicity,

[1] $\theta_0, \theta_1, \theta_2$ stands for prevalence, sensitivity and specificity respectively.

Table 2.2: Illustrative example of GLMM training

Private Data of Alice and Bob

	Alice		Bob	
	D_1	D_0	D_1	D_0
T_1 (Positive)	19	3	95	5
T_0 (Negative)	3	62	0	150
Total	22	65	100	150
Private Value (π)	22/87 = 0.2528		100/250 = 0.4	

Partial Log-Likelihood Equations Used by Alice and Bob

Alice Computes	Bob Computes
$C_A = f(a) \cdot \mathcal{N}(a; \theta_0)$; a stands for the private value of Alice	$C_B = f(b) \cdot \mathcal{N}(b; \theta_0)$; b stands for the private value of Bob
$f(a)$ = Binomial distribution parametrized by a.	$f(b)$ = Binomial distribution parametrized by b
$\mathcal{N}(a; \theta_0)$ = Logit-Normal distribution parametrized by a and θ_0. θ_0 contains the mean (β_0) and standard deviation(τ_0) of $\mathcal{N}$.	$\mathcal{N}(b; \theta_0)$ = Logit-Normal distribution parametrized by b and θ_0. θ_0 contains the mean (β_0) and standard deviation(τ_0) of $\mathcal{N}$

Training iterations

Iteration Number	Server Sends	Alice Computes	Bob Computes	Server Receives	Increase/ Decrease Compared to previous iteration
n	θ_0	C_A	C_B	$C_A + C_B$	$+/-$
1	0.0, 0.1	0.02781	0.071874	0.099684	+
2	0.25, 0.1	0.01259	0.05563	0.06822	-
3	-0.25, 0.1	0.04612	0.0187	0.06482	-
4	-0.75, 0.05	0.1887	0.5795	0.7682	+
5	-0.75, 0.06	0.1371	0.6495	0.7866	+
6	-0.8, 0.06	0.2100	0.9195	1.1295	+
7	-0.85, 0.06	0.3901	1.4333	1.8234	+
8	-0.9, 0.06	0.4026	1.6184	2.0210	+
⋮	⋮	⋮	⋮	⋮	⋮

we shall present an example where the server is trying to optimize the parameter θ_0 for PL_0 and the process involves two parties (Alice and Bob) only. Table 2.2 shows the training process iteration by iteration. It can be seen that after a few initial iterations, the server is consistently updating the parameter to increase the $C_A + C_B$ value compared to the previous iteration (fifth column). The server continues updating the parameters in this manner until the $C_A + C_B$ value stops increasing (reached global

optimum) or the iteration limit is reached.

One detail that we have not described yet is the process through which the server selects the next set of parameters in each iteration. The server uses Bayesian optimization algorithm to select the parameters for the next iteration [18]. For this, the server first creates a surrogate function. Then it samples parameters and calculates the expected values of the actual objective function from the surrogate function. The parameter associated with highest expected value is selected for the next iteration and shared with the parties.

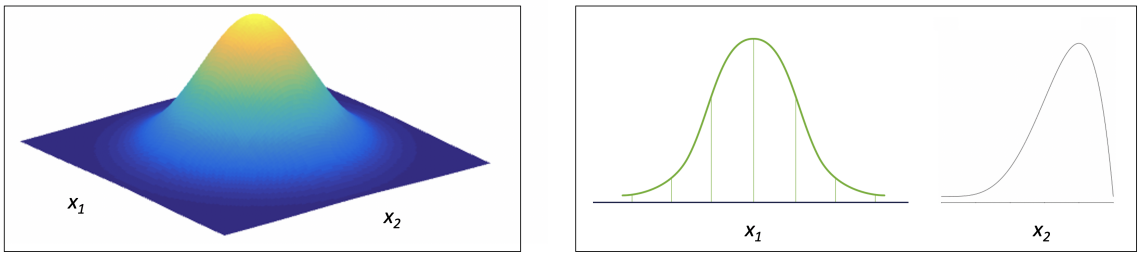
2.4 Understanding the GLMM

Let's consider logistic regression as a Linear Model. Lets assume the input points have 2 predictor features $X = \{x_1, x_2\}$. The output is a single scalar response $Y = \{y\}$. The final equation of the logistic regression will be:

$$y = \frac{1}{1 + \exp^{-\Sigma(\beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2)}} \tag{2.7}$$

Therefore, logistic regression models the **joint** distribution of X. In other words, linear models such as logistic regression does not have any way to model if predictor variables are actually independent and follows different distributions from each other.

Now consider the case of GLMM. Instead of assuming that all predictor variables follow the same distribution and can be modeled by a joint distribution, GLMM makes the assumption that each predictor variable or confounding variables follow their own distributions. Therefore, the final output should be a linear combination of variable distributions.



Logistic Regression Model Generalized Liner Mixed Model

Figure 2.1: Difference Between Linear Model and Generalized Linear Mixed Model. No matter how many predictor variables you have, linear models always model the optimization space as a single distribution. Generalized Liner Mixed Models treat each predictor variable with respect to their own distributions.

For example, consider the predictor variable x_1 in the example above follows a Gaussian distribution and predictor variable x_2 follows a skewed Cauchy distribution. A generalized linear mixed model can model two variables with two different distributions and then optimize based on the **linear** aggregation result of them. Figure 2.1 shows the difference of optimization process between logistic regression (LM) and GLMM.

Getting back to Equation 2.3, it is evident how the GLMM we described follows this principle. Log-Likelihood can be interpreted the measure of a goodness of fit of observations to a distribution. So, Each party collaboratively computes the log-likelihood of a single predictor variables. The server selects the final distribution for that predictor variable which has the highest log-likelihood value. Since there are 3 predictor variables (π, Se, Sp), the server follows the same process 3 times to find out the parameters of 3 distributions associated with 3 predictor variables. If there were n number of predictor variables, the server would have followed the process n times to determine the parameters of n distributions.

This is why GLMM's have been adopted widely in real world scenarios. In real world use cases predictor variables often **do not conform** to the assumption of i.i.d (independent and identically distributed) which is made by standard linear models. This robust modeling capacity of GLMM combined with collaborative nature of training GLMM's over distributed data inspired us to take a look at it from privacy perspective. Moreover, during our literature review we found out that in majority of the use cases GLMM is trained over clinically sensitive data. This was another reason for us to perform a systematic privacy analysis of GLMM.

2.5 Federated Learning

Federated learning is a ML technique by which a model can be trained on multiple hosts or devices using their own dataset. FL allows multiple parties to build a shared ML model without sharing the data and thereby addressing issues such as data privacy, data security and data access control. Each host runs the model on their own dataset and shares the parameters or intermediate results.

Chapter 3

Privacy preserving Indoor Contact Tracing

3.1 Literature Review

Recent works on contact tracing have focused heavily on privacy issues of the underlying principles of contact tracing. Epione [19] is a lightweight contact tracing system proposed by Trieu et. al. which relies on faster private set intersection cardinality method to achieve efficiency over other similar methods. Their work specifically focuses on the case of matching between large scale contact database and small input queries. DP-3T [20] is one of the most prominent one among the recent works on proximity tracing. This proposed system uses a decentralized model and provides robust privacy guarantees. PEPP-3T [21] is another similar protocol with centralized design instead of decentralized. Centralized systems offer more useful data which can be used by health authorities to make effective decisions however, numerous scholars

believe that this approach might be harmful as it can become a mass surveillance tool for governments. EPIC [22] is another similar scheme which provides a proximity tracing scheme by using hybrid wireless and localization technology. However, this system has scalability issues as noted in [23].

There are multiple real world implementations of the contact tracing protocols. South Korean Government implemented their own contact tracing system which is reported to have widespread privacy issues [24]. Tracetogether [25] is an adoption of the contact tracing protocol by the government of Singapore. It is a centralized system which stores the user phone number, identifying information and a randomized token. It does not store gps locations of the user. However, being a centralized system it is vulnerable to multiple security and privacy risks. Moreover, the slow adoption rate among the population also hinders it's applicability and effectiveness to a certain extent. In addition, Bluetooth based systems suffer from security vulnerabilities and privacy issues. Sophisticated GPS based solutions exist as noted in [26] by Berke et. al. However, the computational overhead of this system prevents implementation by hindering scalability due to the fact that MPC (multi party computation) is used to preserve privacy of this system.

Private Kit [1] is another protocol proposed by Raskar et. al. which is a privacy-first contact tracing protocol by design. It provides a mix of voluntary sharing and unicasting which eliminates the need of a central monitoring entity. PACT [27] is another example of similar protocol with improvements over the earlier versions of contact tracing protocols available at the beginning of the pandemic. Technology industry leaders Apple and Google proposed their joint collaborative contact tracing

protocol for smartphone devices which ensures strong privacy [28]. However, we are yet to witness any major government taking their services to combat the pandemic. COVI [29] is another contact tracing framework which attracted a lot of attention from academia. This work is different from the aforementioned ones in the sense that it contains discussion and mitigation strategy for ethical and privacy issues. Moreover, it also contains guidelines for secure data collection as well as using them for machine learning models which makes it the most complete and robust privacy preserving data collection and analysis framework for contact tracing data till date.

3.2 PAARS design

While designing PAARS our key motivation was to ensure maximum data security and privacy. In the meantime the utility of the system was also not to be compromised by design issues or slow adoption rate among users. Keeping these in mind, we took a novel approach in PAARS system design principles compared to already implemented systems. Instead of token exchange between users to detect proximity, we use the existing wireless network present in an organization for token exchange between users and the server which can be later used to detect contact between users. In addition to that, our system does not require any additional infrastructure or collaboration with any third party. Therefore, the scalability and adoption among users are easier than the existing approaches.

3.2.1 Detection of Contact Event between Two User

We assume that a network consisting of wireless network access points (e.g. routers) is present in the organization environment. We denote this network as a set of access points $R = \{R_1, R_2, R_3, \cdots, R_n\}$. We assume that at any given time all elements in R simultaneously broadcast a single pseudo-random number which we denote as K_{net}. This pseudo-random K_{net} changes over time. The time difference between each new pseudo-random number broadcasted by the central network is denoted by the τ.

According to Center for Disease Control guideline, a contact event between two individuals is defined as the individuals being in 2 metres of each other for more than 15 seconds [30]. Taking this definition into account, our system divides the system environment in GPS co-ordinate blocks. A GPS co-ordinate block in our system contains co-ordinates that are within maximum 2 meters. Every co-ordinate block is assigned an unique identifier by the system. When an user enters into the system environment, the smartphone of the user connects to the network of the environment and starts exchanging tokens with the system. In order to generate a token and exchange it with the system, the user performs the following computation on the smartphone. Let's assume that the current time interval is τ_k. Firstly, it takes the unique identifier of the GPS co-ordinate block it is currently located in. We call it $C_{gps}^{\tau_k}$. Secondly, it takes the pseudo-random number generated by the wireless access point covering that GPS co-ordinate block at that time which we denote as $K_{net}^{\tau_k}$. The token ID generation process consists of concatenating these two and then passing them through SHA-256 hash function.

$$\mathrm{ID} \leftarrow \mathrm{SHA}_{256}(C_{\mathrm{gps}}^{\tau_k} || K_{\mathrm{net}}^{\tau_k})$$

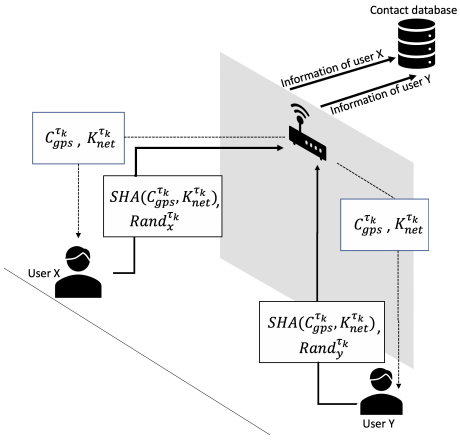


Figure 3.1: $User_x$ and $User_y$ is in the same GPS co-ordinate block. They connect with the same wireless access point which provides them $C_{gps}^{\tau_k}$ and $K_{net}^{\tau_k}$. $User_x$ generates a random number $Rand_x^{\tau_k}$ and $User_y$ generates a random number $Rand_y^{\tau_k}$. Both users concatenate the parameters sent by the server with the network address of the access point and pass them to a hash function to generate a hashed token. Both users upload their generated random numbers and hashed tokens to the server. The server stores the hashed tokens and the random values sent by the users in a database to later use them for contact detection.

Therefore, it can be ensured that at a given time τ_k, if two users are in the same block of GPS co-ordinates, they will generate the same token which will indicate a contact event between them. The user also generates a pseudo-random number $Rand_{\text{user}}^{\tau_k}$. The user then sends this pseudo-random number along with the hashed token to the server. The server stores the hashed token, pseudo-random number , time , current infection status of this user and the probability value of being infected (initialized with value 0) in a database table. The server does not store any other persistent information about the user that might compromise the anonymity of the user. The user also stores the hashed tokens and the pseudo-random numbers sent to the servers on his/her device.

In the server database, a contact event between two users will be registered as

Table 3.1: A sample contact event in the contact database. Both entries have same hashed token value but different user generated pseudo-random number entry.

Hashed Token	$Rand_{\text{user}}$	Time	Status	Probability
FF56182345FA6BF7	18728712789	6:32	N/A	0
FF56182345FA6BF7	74826390017	6:32	N/A	0

having same hash token values with different $Rand_{\text{user}}^{\tau_k}$ value.

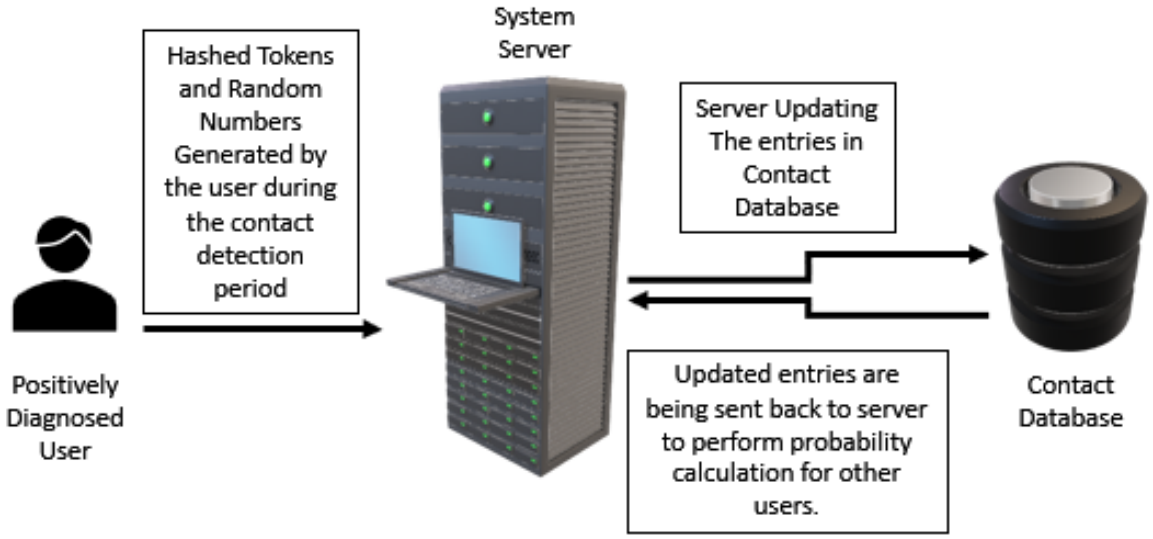


Figure 3.2: Workflow of the system once an user reports to be positively diagnosed and chooses to share the hashed tokens and pseudo-random numbers generated during the contact detection period of the system.

Once an user tests positive for the infectious disease, that user can **choose to share** the hashed tokens and the random tokens generated during the contact detection phase with the system server. The integrity of the claims of positive test diagnosis can be ensured by secure methods. For instance, public health authority may provide each positively diagnosed user with an unique identifier or secure login methods to verify the claim of the user being positively diagnosed. After the authenticity of the claim of being positively diagnosed is confirmed, the user can voluntarily share the hashed tokens and the random numbers generated by the user with the

Table 3.2: Updated contact database entries when an user reports being infected and chooses to share relevant information with the server.

Hashed Token	$Rand_{\text{user}}$	Time	Status	Probability
FF56182345FA6BF7	18728712789	6:32	TBD	TBD
FF56182345FA6BF7	74826390017	6:32	Infected	1

system server. The system server performs a series of operations after receiving the tokens and random numbers from the user. The system updates the matching entries for the contact database where it sets the value of Status as "Infected" and value of probability as 1. Moreover, for the matching entries of same hash tokens of other users we set the status column and probability value as "TBD". This workflow is shown in figure 3.2.

Referring back to table 3.1, let's assume that an user whose claim of being positively diagnosed has been validated by the system, chose to share his/her hashed tokens and pseudo-random numbers generated with the system server. The server received all these and updated the database accordingly. Moreover, let's assume that among the contact event entries uploaded by the user, there is a contact event entry with hashed token value of "FF56182345FA6BF7" and $\text{Rand}_{\text{user}}$ value of "74826390017". After the database is updated, the relevant contact entries will be as in table 3.2.

After updating the database entries, the server retrieves the updated entries and calculates probability of being positively diagnosed for the users who have had a contact event with the infected user. To calculate this probability we formulated a novel probability calculation model which depends on state of the art epidemiological models available in the literature such as variations of Susceptible-Infected-Removed (SIR) model [31] while taking in account factors such as temporal shedding nature

of the viral transmission [32]. The details of this probability calculation model is described in the following subsection.

3.2.2 Mathematical Model for Probability of an User Being Positively Diagnosed

Let's assume the event of being tested positive at time t is denoted by $\lambda(t)$ which is a binary random variable which takes the value 0 if the user is tested negative and 1 if tested positive. We denote the contact database entries for an user u up to time t as C_u^t. We intend to find out $\mathbf{P}(\lambda(t) = 1|C_u^t)$ which is the probability of being positively diagnosed of an user for the disease given the contact database entries of the user. Let's assume that C_u^t is a set of contact entries which contains contact event entries that coincides with both infected and non-infected users. Therefore, this can be written as a set, $C_u^t = \{C_v^+ + C_v^-\}$ where $v \in V$. V denotes the set of users who had contact events with user u and the $+$ and $-$ superscript denotes whether the user v is diagnosed as being positively infected or not. The set of all positively diagnosed users are denoted as D^+. Therefore, we can write the intended probability function of being infected as:

$$\mathbf{P}(\lambda(t) = 1|C_u^t) = \mathbf{f}(\mathbf{P}(\lambda(t) = 1|C_v^+)); \forall v \in (V \cap D^+) \tag{3.1}$$

To calculate our intended probability function $\mathbf{f}$, we need to find the average of the probability of each individual contact event with user v who were positively diagnosed for the disease. This is due to the fact that each contact event with a positively diagnosed user $v \in V$ can contribute to the user u in question to be positively diagnosed.

Moreover, we assume that each contact event between two users are independent of any prior contact events between any two users. Therefore, it is possible to view each individual contact events between users as independent events. Therefore, the equation 3.1 can be written as:

$$\mathbf{P}(\lambda(t) = 1|C_u^t) = \frac{1}{N}\sum_{i=0}^{N}\mathbf{P}(\lambda(t) = 1|C_v^+); \forall v \in (V \cap D^+) \quad (3.2)$$
$$where, N = |(V \cap D^+)|$$

In order to understand how our system calculates the individual probability of each contact event contributing to the total probability we would like to draw attention to the [31] which shows how a variant of widely used Susceptible-Infectious-Removed (SIR) model was used to mathematically model the recent COVID-19 pandemic outbreak in Italy. In this work, the authors noted that transmission of the disease from one individual to another can depend on multiple factors. However, the most important factors are the time duration of a contact event between two individuals and the viral shedding factor of the infected individual. The later one can be described as the rate at which the infected individual is spreading the virus within a certain distance [30]. We take these two important aspect into account and design our probability calculation method accordingly.

We define $E_u^v(t)$ as the exposure score of the contact event between user u and v at time t which we calculate from the work described in [33]. In this work, the authors design multiple models to determine the severity of a contact event between two individuals. In our particular case, we will use the sigmoid model developed by the authors to determine the severity of a contact event. This model takes the

duration of a contact and the distance between two individuals during the contact as input of the model and produces a scalar value which is the severity score of the contact event.

We define $S^v(t)$ as the viral shedding rate of the infected user at time t. This can be directly determined by using the work described in [32]. In this work, the authors drew the temporal relation between the days of infection and viral shedding. According to this work, for the first few days after being infected, the viral shedding of an infected person is significantly higher than the subsequent days. We can get $S^v(t)$ from this model by inputting the number of days when the contact event took place.

Therefore, we can derive the probability of being positively diagnosed from each contact event by using the following formulation:

$$\mathbf{P}(\lambda(t) = 1|C_v^+) = \frac{E_u^v(t) * S^v(t)}{\max(E_u^v(t) * S^v(t))} \tag{3.3}$$

Using equation 3.3, we can calculate the probability of being positively diagnosed for each contact events and average them to get absolute probability score for an user being positively diagnosed. However, there is a caveat in the above formulation which is privacy breach. A malicious user may track the differences between the updates of his/her probability score and might attempt to infer which contact event caused the update of the probability score. Therefore, we attempt to mitigate this problem by using differential privacy. After calculation of the left hand term in 3.3, we add a noise to it which is from Laplace distribution. Then equation (3.3) becomes:

$$\mathbf{P}(\lambda(t) = 1|C_v^+) = \frac{E_u^v(t) * S^v(t)}{\max(E_u^v(t) * S^v(t))} + \mathcal{L}(0, \frac{\alpha}{N}) \tag{3.4}$$

where α is the regularization parameter for the noise and the mechanism is $\frac{N}{\alpha}$ differentially private. The accuracy analysis of this noise addition mechanism will be discussed in the utility analysis section.

Using equation 3.4 for all the contact events between the user in question u and the infected users v, we can derive noisy probability scores for each contact event. We input these noisy probability scores in equation 3.2 and calculate the final probability score of an user being positively diagnosed given the contact events in the contact database.

3.2.3 Privacy Aware User Alerting System

The server has a set of hashed tokens H_{sys} which contains the hashed tokens that have positive probability scores. Let's assume user u wants to access his/her probability score from the system. In order to do that, the user uses H_u which is the relevant hashed tokens within the time period which is generated by the user and have been shared to the server. The server and the user performs a PSI (private set intersection) as described in the work of [19]. The result of $PSI(H_{sys}, H_u)$ is shared to the user u. If the result returns a non-empty set, the user u makes a request to the server which contains the result of the $PSI(H_{sys}, H_u)$ and the $Rand_{user}^{\tau_k}$ associated with each elements in the result of $PSI(H_{sys}, H_u)$. The server receives the request and sends back the probability scores associated with the query. The user receives the result and based on the result the user can decide to self-isolate, quarantine or seek medical help from professionals. It is important to note that the user gets the probability scores associated with the entries in the result of $PSI(H_{sys}, H_u)$ and his/her

generated $Rand_{user}^{\tau_k}$ only. Therefore, he/she does not have access to the probability scores calculated for the other users with the same hashed token values.

3.3 Utility Analysis

3.3.1 Building Occupancy Level Determination

PAARS can automatically calculate the building occupancy level in real time by analyzing the number of active connections sending in hashed tokens in every wireless access point within the system environment. Based on the level of occupancy at a given time, it can alert the authorities if maximum occupancy level set by health professionals are being violated.

3.3.2 Error Estimation of the Probability Calculation Model

In order to calculate the accuracy of the noisy probability scores in subsection 3.2.2, we adopt the accuracy analysis method presented in the work of [34] and [35]. Let's assume P_{actual} is the actual probability and P_{noisy} is the calculated noisy probability. According to [35], the error of a differentially private mechanism output is determined by the variance $V(P) = \mathbb{E}[(P_{noisy} - P_{actual})^2] - \mathbb{E}[(P_{noisy} - P_{actual})]$, where $\mathbb{E}$ is the mathematical expectation. Referring to equation 3.4, $(P_{noisy} - P_{actual})$ in this case is $\mathcal{L}(0, \frac{\alpha}{N})$ which has variance of $\frac{2\alpha^2}{N^2}$. Therefore, each term on the right hand side in equation 3.2 is expected to add $\frac{2\alpha^2}{N^2}$ error to the final output. Accordingly, the

final expected accumulated error is:

$$error = \frac{1}{N} \sum_{i=0}^{N} \frac{2\alpha^2}{N^2} = \frac{2\alpha^2}{N^2} \tag{3.5}$$

3.4 Security Analysis

To analyze the security of the system firstly we consider a set of parties and a few protocols. We assume that in our proposed system there is a set of parties who have agreed to perform some computation. Moreover, the parties have consented to release the final result of the computation to a specific party and nothing else will be released from any other parties, not even the computational process used. There are two classical security models in this case:

- Semi-Honest Model: This type of adversary is someone who is presumed to follow the execution protocol. However, it attempts to obtain extra information from the execution protocol.

- Malicious Model: This type of adversary may attack the system using any possible strategy i.e. brute force attack, side channel attack, supplying inconsistent output or trying to infer information about system data or execution process of the computation involved.

In this work, we define three individual entities: The user group U, the semi-honest system server S and malicious adversary A who would try to attack the system using any possible method that is executable in polynomial time. Attacker A can be a malicious user or someone outside the system environment. For simplicity, we

assume that the no entity is colluding with one another. Moreover, we assume that all communications between the user U and the server S are securely authenticated (i.e. TLS).

The robustness of all proposed decentralized solutions so far relies on the fact that individual user identities are not linkable to the pseudo-random tokens they generate. However, the exchange of these pseudo random tokens between users still poses some security threats such as possibility of identifying positively tested users in case of a single contact event in the given time period. In the following paragraphs we will show that our proposed system prevents majority of the security risks of the previous approaches while ensuring privacy.

- **Positively tested user identification attack**: In this attack a malicious user tries to identify a positively tested user by matching the tokens exchanged between them. Most of the proposed contact tracing/proximity monitoring systems have token exchange as a core part of their functionality. However, in our proposed system there is no token exchange between two users. Therefore, this attack is not feasible in our system.

- **False positive reporting attack by user**: In this attack a malicious user U_{mal} tries to generate false alarm, by reporting him/her self as positively diagnosed with the disease to the server S. This attack could cause erroneous computation on the server and result in confusion and panic among users. However, in our proposed system, this attack is not possible as every claim made by any user of being positively diagnosed for the disease is verified with the proper authority.

- **Impersonation attack by user:** This attack is aimed at faking a person's

presence when that person was actually not present there. The attacker A gathers exchanged tokens from other users and broadcasts them. If any of those tokens are later marked as tokens from an individual who has been positively diagnosed with the disease, a number of users may be falsely alerted of being in contact with that person while in reality they never had any contact with that positively diagnosed person. In our proposed system this attack is not possible as individual users do not exchange tokens with each other.

- **Exposure of Social Interaction Graph of the user by the server:** One of the major privacy issues for any contact tracing or access monitoring/regulations system is that there are inherent risks of the server S associating an user U with a set of users $U_{contacts}$ where each member of the set $U_{contacts}$ is the ones who were in proximity of U. In our proposed system, the server S does not keep any persistent identifier of any user U and the user U never shares any persistent identifier of him/her to the server S. The server S only receives the securely hashed values from the user U which can not be linked back to individual users. Therefore, in our proposed system this attack is not possible.

- **Track user's location by the server:** This attack is performed by the server to use the shared tokens to track the location of any individual user. Theoretically speaking, any system where the user U exchanges some kind of token with the server S , it is possible for the server S to track the user by using passive packet sniffer. In our proposed system we assumed the server to be semi-honest entity which does not employ any such malicious tactics. Therefore, it is not possible for the server S in our proposed system to track the users movement

since the exchanged tokens contain no persistent identifier and they change over time based on user's location.

3.5 Discussion of Design Decisions

3.5.1 Decentralized Data Storage

One of the key differences between PAARS and the existing approaches is the fact that PAARS enables decentralized data storage. Each organization implements their own version of PAARS and stores the data locally. Therefore, it is not feasible for an adversary to compromise all different data storages residing in different organizations at once and get a complete picture. On the contrary, majority of the existing approaches that rely on ephemeral token exchange between users store these tokens from all users in large central databases. Therefore, a data breach in these systems from the inside or the outside can bear catastrophic effect in terms of privacy.

3.5.2 Differential Privacy

To the best of our knowledge, no work in literature yet addressed the user side privacy issues in contact tracing/access monitoring systems by incorporating differential privacy. Most of these work rely on using homomorphic encryption or private set intersection (PSI) method [19] which is computationally costly. In our work, we use efficient PSI for once in the user query only. The rest of the privacy is guaranteed by differential privacy.

3.5.3 Infrastructure Requirement

PAARS do not require any significant addition in the infrastructure available in the environment. This is a major advantage over existing systems or the ones being implemented. For instance, every organization has wireless network now a days and majority of the population has access to smartphones. Therefore, deploying and maintaining PAARS is easy and adoption should be fast. On the contrary, implementing PAARS architecture in city-wide scale would require a singular network which has city-wide coverage. Cell service providers are capable of providing such network. However, to collaborate with them to build and deploy a system like this would require a lot of time and effort, whereas, PAARS can be deployed quickly in organizational level.

3.5.4 GPS accuracy

GPS accuracy can be perceived as a technical issue in PAARS system design. However, using other sensors (i.e. magnetometer and accelerometer and IMU) available in smartphones, the accuracy issue can be effectively resolved [36] .

Chapter 4

Privacy-Preserving Generalized Linear Mixed Model

4.1 Literature Review

4.1.1 Homomorphic Encryption for Privacy Preserving ML model Training.

Privacy preserving machine learning or statistical model training is one of the most widely investigated topic in literature. Homomorphic Encryption (HE) is one of the most popular privacy preserving tool used for this purpose. Prior works in [37, 38, 39, 40, 41] have focused on training logistic regression model in a federated setting using HE. Logistic Regression model is a linear model. However, it does not take random effects into account and therefore, can not be directly termed as a GLMM.HE have been used in deep learning model training [6, 42, 43, 44]. However, efforts to use HE

in federated learning over large datasets are rare. Majority of the work in this domain have focused on making deep learning on encrypted data more robust and efficient.

Apart from logistic regression and deep neural networks, HE have found applications in a myriad of statistical modeling in medicine and genomics (e.g. Naive Bayes Classifier, Decision Tree etc.) [45]. These algorithms are computationally less intensive compared to deep learning models when the input dataset is encrypted. Although direct comparison is not possible for the GLMM used in this paper, our proposed method also appears to be computationally expensive than HE based Naive Bayes or Linear Regression models. The reason behind this is Naive Bayes or Linear Regression models usually take place in a single round while our proposed method performs optimization on multiple rounds.

4.1.2 Statistical Attack On Private Dataset.

Statistical Attack for dataset reconstruction is widely investigated in the literature. Homer *et. al.* proposed a famous attack where individual identity could be resolved from sensitive genomic data microarrays [46]. Subsequent works in this domain extended on Homer's attack to propose newer and more sophisticated attacks [47].

Another well known attack is membership inference attack on deep learning models [3]. In this attack, an attacker tries to infer whether a data point was part of the training set that is used to train the model. Membership inference attack heavily relies on gradient information. This served as the inspiration in our attack described in Section 4.3.

4.1.3 Generalized Linear Mixed Model in Medical Domain.

In the introduction, we mentioned that GLMM is one of the most widely adopted class of statistical models in medical domain. GLMM is first introduced as a likelihood based approach in [48]. Maximum Likelihood estimation is the most common objective function for GLMM [49]. In epidemiology research, GLMM is regularly used to model specificity and sensitivity [50]. Clinical trial results of new drugs or treatment methods are also modeled using GLMM [51]. An extensive analysis of use cases of GLMM in medical domain can be found in [52]. The robustness and the predictive power of GLMM in medical use cases is presented in [53].GLMM is also used widely in data modeling tasks involving sensitive genomic data. Wang *et al.* used GLMM to study the problem of genome wide association [54]. Recent works on genome wide association mapping have focused on the use of GLMM to tackle the scalability issues associated with the problem [55]. Song *et al.* introduced a GLMM based ensemble predictor in [56] for cancer genome prediction. Similar approaches have been used for predicting late stage Alzheimer disease based on Bayesian GLMM [57].

4.2 Problem Formulation

4.2.1 Threat Model

We propose a scheme where n parties jointly train a GLMM model M using FL. There is a central server and a crypto service provider in this scheme. The

central server is called the *aggregator* and the crypto service provider is called *csp*. Our proposed scheme is designed to withstand four type of potential adversaries: 1. aggregator 2. csp 3. collaborating parties 4. outsiders

Honest-But-Curious Aggregator and CSP

Honest but curious setting (also known as semi-honest adversarial model) is widely adopted in secure multiparty computation protocol design since it's inception [58, 59]. In this setting, the adversary follows the protocol correctly but will try to infer private information from the collaborating parties. Therefore, in our case, the aggregator or the csp will not deviate from the predetermined ML model execution. However, they will try to infer additional private information from all the data that can be accessed during the scheme execution. Our proposed scheme does not consider collusion between aggregator and csp.

Honest-But-Curious Collaborators

Our proposed scheme also considers the threat from honest but curious collaborating parties. Honest but curious collaborating parties stay honest during the ML model execution scheme. After the completion of the training process, they try to gain additional private information regarding the other honest parties by using the final model and the data collected during the execution. We consider collusion among collaborating parties themselves. However, our scheme does not consider collusion among parties, aggregator and csp.

Malicious Outsiders

We designed our proposed scheme to be resilient against outsiders. We consider any user of the final model as potential adversary who would want to infer private information about the data used to train the model. Moreover, any observer who is monitoring the communication of the parties, csp and aggregator during the training scheme execution time is also considered as potential adversary. We assume that all communication among parties, aggregator and csp are encrypted which prevents outsiders from injecting their own input in the execution process.

4.2.2 Problem Statement

Let's consider that there are n parties in the execution environment. Moreover, an aggregator $\mathcal{A}$ and a csp λ is also present. Parties $P_1, P_2, \cdots, P_n$ has access to dataset $D_1, D_2, \cdots, D_n$. They want to use a distribution learning algorithm f_M which will produce a final output model M where $f_M(D_1 \oplus D_2 \oplus \cdots \oplus D_n) = M$ and $\oplus$ indicates concatenation. The problem is to propose a solution model that can protect against the following threats:

Inference During the Learning Process

Inference during the learning process refers to any $P_i, \mathcal{A}$, or λ learning about other parties private dataset using the data exchanged (i.e. gradients, log-likelihood, weights) for training (execution of f_M). Let us assume that f_M is a collection of steps s. Each step is a computational operation which is preceded by a query Q_s from $\mathcal{A}$. During the execution of f_M, each party P_i must respond to the query Q_s by

performing some computation on the private dataset D_i. The types of these queries are f_M specific in nature. For instance, in the case of building a decision tree, the Q_s will be number of rows in the D_i which matches a certain criteria. In case of a distribution modeling, Q_s will be the partial log-likelihood values computed on rows of D_i. To build a privacy-preserving FL system, one must account for the risk of inference over the responses of query Q_s provided by the parties.

Privacy-preserving FL systems usually mitigate this risk by using Secure Multi Party Computation (SMPC) protocols. SMPC allows multiple parties to obtain the output of a function without revealing the party-specific input data. SMPC also prevents the leakage of any other information apart from the output. However, prior works have shown that the output can also be used to make inference about private dataset [3]. Therefore, we must also consider the possibility of inference over outputs.

Inference over the Output of the Learning Process

This refers to the predictive model M which is the output of f_M. Recent works have shown that black-box access to model M is enough for an attacker to infer private training data. Any privacy-preserving FL system must consider this kind of outside threat and take steps to mitigate these. Prior works have used differential privacy to mitigate this risk [4]. In this work, we will *not* adopt differential privacy to mitigate the risk of inference over output. The rationale for this decision is discussed in Section 4.7.

4.3 Privacy Leakage of GLMM training in Federated setting

Let us demonstrate how the server can compromise the private information of collaborating parties by a motivating example. Referring to Table 2.2, we can see that Bob holds private value 0.4. In this example, we shall demonstrate how the server can reach the conclusion that Bob's private value is 0.4. Before diving into the example, let us formalize a few notations first. We assume that during the training, the server and Bob exchanged K intermediate computation values (i.e. the server updated the parameters K times and Bob shared the computed value in response for K times). We also assume that server can guess with reasonable confidence that it is maximizing log-likelihood for a GLMM.

In order to gain access to the private information, the server leverages the widely used concept of "Rainbow Table" which was proposed by Oechslin *et al.* in [60]. This concept is widely used in cryptanalysis to crack passwords. This table contains billions of password and salt combinations hashed in all possible hash formats such as MD5 and SHA-256. In order to crack a password, a malicious actor just has to map the salted hash of the password to an entry value in the table. We borrow this concept of rainbow table and modify it according to our attack policy.

Let us assume that on training iteration 1, the server sends the parameter $\theta_0 = \{0.0, 0.5\}$ to Bob. Bob receives the parameter and computes $\mathcal{N}(0.4; \theta_0) = 0.3720$[1]. Bob also computes $f(0.4) = 0.05144$. In both computations, Bob uses his private value 0.4. After that, Bob computes $C_B = \mathcal{N}(0.4; \theta_0) \times f(0.4) = 0.01914$. Bob

[1]For Definition of $\mathcal{N}$ and f, refer to Table 2.2

shares this C_B value with the server. The server divides the received C_B value with $\mathcal{N}(0.1;\theta_0), \mathcal{N}(0.2;\theta_0), \mathcal{N}(0.3;\theta_0), \cdots, \mathcal{N}(0.6;\theta_0)$ and stores the division results in a row of the table.

On training iteration 2, the server sends the parameter $\theta_0 = \{0.1, 0.4\}$ to Bob. Bob then follows same pattern as the previous iteration and computes $C_B = \mathcal{N}(0.4;\theta_0) \times f(0.4) = 0.00508$. The server receives the shared C_B value from Bob and divides the received value with $\mathcal{N}(0.1;\theta_0), \mathcal{N}(0.2;\theta_0), \mathcal{N}(0.3;\theta_0), \cdots, \mathcal{N}(0.6;\theta_0)$. The server stores the division results in the table. The server continues this process in each training iterations. There are two important aspects to be noted here:

- The server tries to perform the division process for as many parametrized distributions as possible (i.e. Gaussian, Laplacian, Cauchy, Gamma etc.)
- The server tries to use as many sample values as possible for each distributions. In the example above, the server uses only 6 sample values (0.1 to 0.6). In practical scenario, the server can use hundreds of sample values to improve accuracy.

The stored values by server are shown in the table 4.1. After the training iterations are complete, the server computes standard deviation for each column of the table. The sample value that is represented by the column with least standard deviation value is the private value of Bob. Referring to table 4.1, it is evident that the column that represents 0.4 has the least standard deviation. Therefore, the server concludes that the private value being used by Bob is 0.4. This indeed turns out to be the case (Referring to table 2.2).

Table 4.1: Division Results computed by Server to determine the private value held by Bob

Iter. count	Distribution sample values used by server					
	0.1	0.2	0.3	0.4	0.5	0.6
1	1.06E+101	2.73E+39	1.205E+13	5.14E-2	1.1E-10	3.64E-16
2	1.20E+23	1.95E+8	1.13E+2	5.14E-2	5.46E-4	3.36E-5
3	8.76E+8	4.65E+2	1.18E+0	5.14E-2	8.53E-3	2.96E-3
4	7.51E+8	4.31E+2	1.14E+0	5.14E-2	8.74E-3	3.1E-3
5	3.47E+8	2.93E+2	9.74E-1	5.14E-2	9.9E-3	3.88E-3
6	3.37E+5	1.85E+1	3.68E-1	5.14E-2	1.78E-2	1.01E-2
7	2.58E+3	2.57E+0	1.81E-1	5.14E-2	2.77E-2	2.1E-2
8	1.00E+2	6.95E-1	1.14E-1	5.14E-2	3.69E-2	3.37E-2
9	5.07E+1	4.93E-1	9.86E-2	5.14E-2	4.18E-2	4.12E-2
10	6.04E+0	2.13E-1	7.35E-2	5.14E-2	4.9E-2	5.46E-2
Std. Dev.	3.2E+100	8.2E+38	3.62E+12	**7.60E-18**	1.67E-2	1.87E-2

We experimented with the algorithm on a real-world dataset. The secret value in this case was the disease prevalence value of the test population of the parties. The experimental results are shown in the Figure 4.1. The server determines the prevalence of the parties with high accuracy when the number of samples tried by the server is comparatively higher. Moreover, the server can also perform this division process multiple times while each time computing on a stricter boundary with low standard deviations. This is an interesting avenue of approach which we intend to pursue further in the future.

4.4 Proposed Solution

We propose FedGLMM, a novel GLMM training method in FL setting that protects against both type of security threats specified in the problem statement. In our proposed method, the aggregator receives homomorphically encrypted log-likelihood values from the collaborating parties. While this preserves the privacy of the collab-

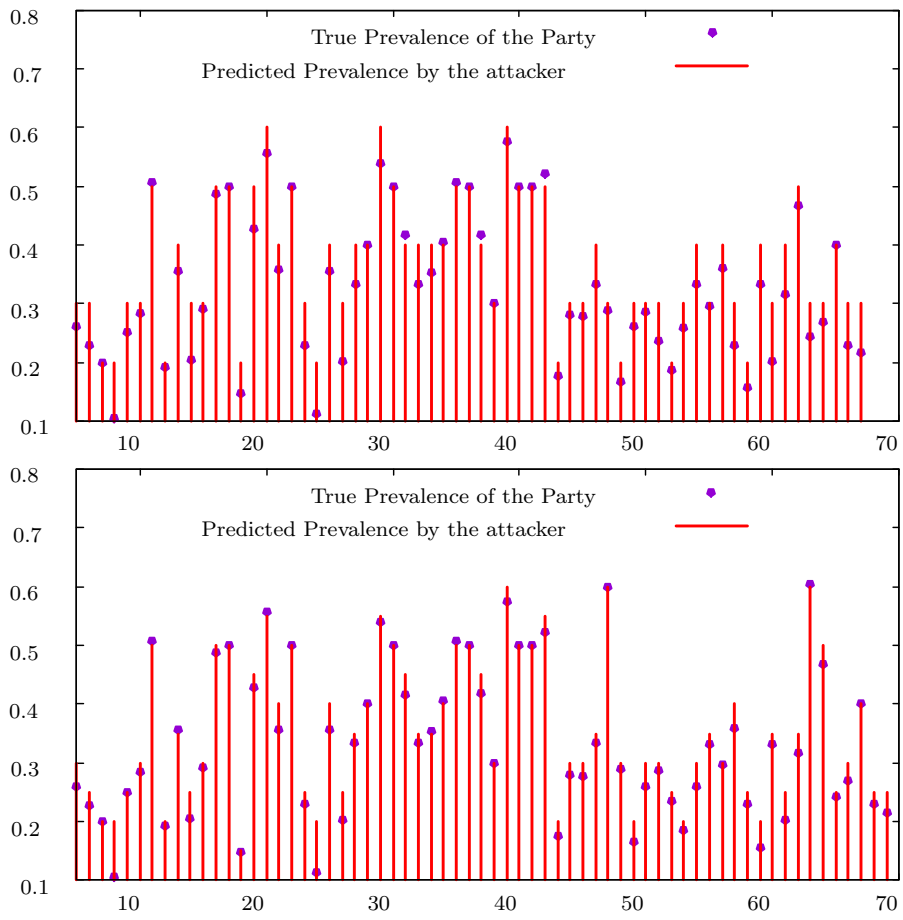


Figure 4.1: The top figure shows our attack algorithm's performance when the rainbow table has only 5 sample values per distribution. Some of the prediction points have high error margin. The bottom figure shows the performance of same attack algorithm when the rainbow table has 10 samples per distribution. The attack accuracy improvement is clearly visible in this occasion.

orating parties, it opens up the aggregator to a particularly challenging problem.[2]

How to optimize on Encrypted Data? Consider a maximization objective function $\mathcal{F}(\theta, x)$ where θ is the learnable parameters of the function and x is the input.

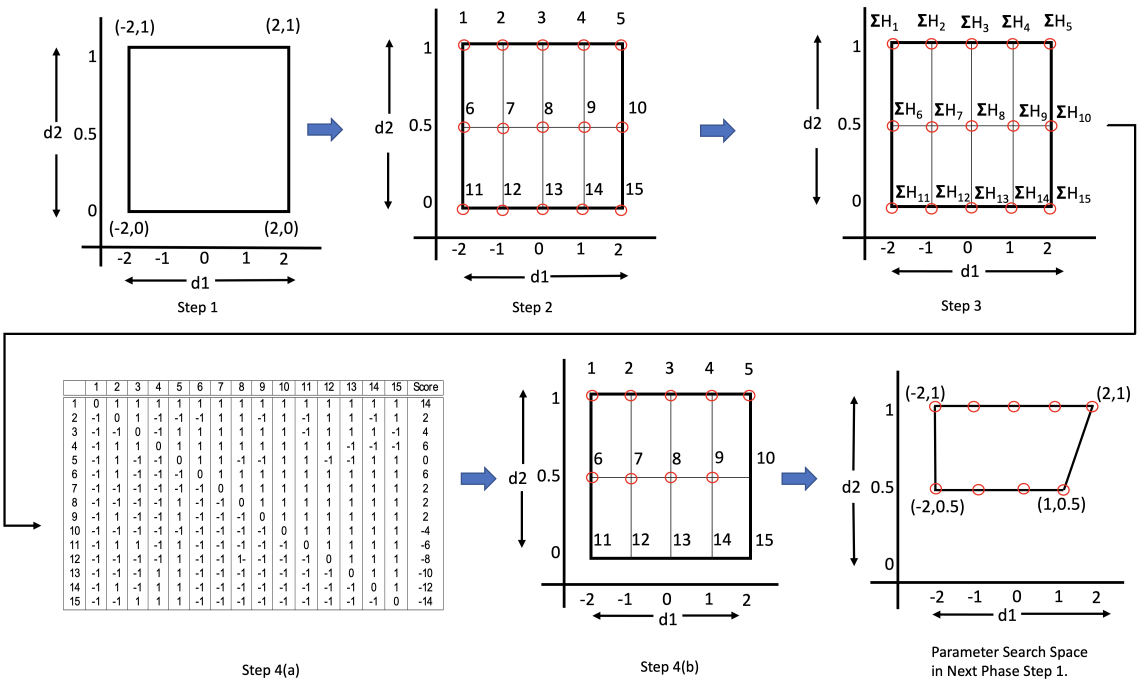
[2]Like majority of the distributed statistical model training process, GLMM training in CL setting is a server-side objective function optimization (maximize or minimize) process. Therefore, for the rest of this work we use the term **optimization** and **training** interchangeably to mean the same process.

Any aggregator that tries to iteratively maximize $\mathcal{F}$, will have to compare the output between iterations to choose the parameters (θ) for further optimization. Specifically, an optimizer has to derive a numeric value for $\mathcal{F}(\theta_i, x) - \mathcal{F}(\theta_{i-1}, x)$ to choose θ_{i+1}. Let us call this numeric value $\Delta\mathcal{F}_i$.

Generally, the aggregator chooses the parameters of next training iteration based on the scale and sign of $\Delta\mathcal{F}_i$. In a federated setting, if the aggregator receives $\Delta\mathcal{F}_i$ in encrypted form, then it can not directly choose the parameters as the scale or sign of $\Delta\mathcal{F}_i$ can not be inferred from its encrypted form. This is where FedGLMM comes in. FedGLMM performs optimization on encrypted $\Delta\mathcal{F}$ by incorporating both an aggregator $\mathcal{A}$ and a csp λ. $\mathcal{A}$ synchronizes the training and selects parameters in each training step while λ assists $\mathcal{A}$ with the optimization.

We will describe the FedGLMM design now. During the initialization of the FedGLMM scheme, collaborating parties and $\mathcal{A}$ agree on modeling a distribution based on their private datasets (i.e., normal, cauchy, laplacian etc.). The csp λ generates a public key, private key pair (P_k, S_k) for Paillier cryptosystem. λ shares the P_k with the collaborating parties and $\mathcal{A}$ and keeps the S_k to itself.

FedGLMM divides the training process on aggregator in **phases**. Each phase contains these steps: ① construction of parameter space, ② select parameter samples and send to parties, ③ aggregation of encrypted result from parties and ④ select next set of candidates for parameter space. We describe these steps with illustrative examples.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Score
1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	14
2	-1	0	1	-1	-1	-1	1	1	-1	1	-1	1	1	-1	1	2
3	-1	-1	0	-1	1	1	1	1	1	1	-1	1	1	1	-1	4
4	-1	1	1	0	1	1	1	1	1	1	1	1	-1	-1	-1	6
5	-1	1	-1	-1	0	1	1	-1	-1	1	1	-1	-1	1	1	0
6	-1	1	-1	-1	-1	0	1	1	1	1	1	1	1	1	1	6
7	-1	-1	-1	-1	-1	-1	0	1	1	1	1	1	1	1	1	2
8	-1	-1	-1	-1	1	-1	-1	0	1	1	1	1	1	1	1	2
9	-1	1	-1	-1	1	-1	-1	-1	0	1	1	1	1	1	1	2
10	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	1	1	1	1	1	-4
11	-1	1	1	-1	1	-1	-1	-1	-1	-1	0	1	1	1	1	-6
12	-1	-1	-1	-1	1	-1	-1	1-	-1	-1	-1	0	1	1	1	-8
13	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	0	1	1	-10
14	-1	1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	0	1	-12
15	-1	-1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-14

Figure 4.2: Aggregator's view of phase execution of FedGLMM in the case of 2 dimensional parameter optimization. Note that the parameter space shrinks phase by phase (comparison between the first figure and the last one).

4.4.1 Construction of Parameter Space

Each phase begins with construction of the n-dimensional parameter space. GLMM is essentially a distribution modeling problem. Therefore, the parameter θ of the objective function $\mathcal{F}$ has two components in this case (i.e., mean and standard deviation). We refer to each component as a dimension. Therefore, θ is a parameter with 2 dimensions (a row vector with length 2). Let's denote the first dimension of θ as d_1 (mean) and the second one as d_2 (standard deviation). The range of d_1 and d_2 (the limit, in which the aggregator will search for suitable values) is provided to the aggregator $\mathcal{A}$ during the initialization of the FedGLMM. Assume that d_1's range is $[-2, 2]$ and d_2's range is $[0, 1]$. Therefore, in phase 1, $\mathcal{A}$ has 4 points to construct a

parameter space, namely $(-2, 0), (2, 0), (-2, 1), (2, 1)$. $\mathcal{A}$ uses these points to form a convex hull[3]. The points within this convex hull represents a parameter space (step 1 in Figure 4.2). In FedGLMM scheme, $\mathcal{A}$ samples candidate parameters for optimization from this parameter space and shares with the collaborating parties. It is to be noted that the formation of parameter space by leveraging d_1 and d_2's range is only limited to phase 1 (initial phase of the training). Starting from phase 2, the most promising parameter samples from the previous phase are used to form the convex hull. This means that phase 2 will use the most promising points from phase 1 to construct the parameter space, phase 3 will use the most promising points from phase 2 and so on[4].

4.4.2 Selecting Parameters in Parameter Space

$\mathcal{A}$ selects parameters within the parameter space by sampling along each dimension in specific intervals (red circles in step 2 of Figure 4.2). After that, $\mathcal{A}$ shares the selected parameters with the collaborating parties. The parties use these parameters to compute the log-likelihood values based on their privately held dataset. After the computation, the parties encrypt the log-likelihood values using P_k from csp λ and send it back to $\mathcal{A}$.

To illustrate this. step, let us continue the example from 4.4.1. In that example,

[3]Imagine that there are n points in 2-d space. Now, the smallest convex shape that holds all the n points within the perimeter of the shape is drawn. This perimeter is called the convex hull of the points.

[4]One can think of the parameter space as the collection of probable parameters (points) which can result in the optimization of the objective function. So the aggregator searches for parameters in that space. In each phase the aggregator reduces the size of that space. Therefore, at the end of all the phases, the aggregator is left with a very small parameter space where all the parameters have very high likelihood of best optimizing the training objective.

d_1 has a range of $[-2, 2]$. Let us assume, sampling interval for d_1 is 1. Therefore, d_1 can take 5 possible values in phase 1. They are $\{-2, -1, 0, 1, 2\}$. Similarly d_2 has range of $[0, 1]$. Let's assume that sampling interval for d_2 is 0.5. Therefore, d_2 can have 3 possible values $\{0, 0.5, 1\}$. Combining d_1 and d_2 creates $5 \times 3 = 15$ possible values for parameter θ. They are marked as red circles in the step 2 of Figure 4.2. Each possible parameter value is assigned a unique index by $\mathcal{A}$. Referring to step 2 of Figure 4.2, $\theta = (-2, 1)$ is assigned index 1, $\theta = (-2, 0.5)$ is assigned index 6, $\theta = (0, 0.5)$ is assigned index 8 and so on.

It is important to note that the sampling interval for any specific dimension is a **hyperparameter** of FedGLMM scheme. Sampling interval controls how many points are sampled and tested by $\mathcal{A}$ in each phase. Therefore, sampling interval also indirectly determines how much time a phase would require to be completed. For example, consider d_2's sampling interval as 0.25 in the previous paragraph. Then, the possible values of d_2 will be $\{0, 0.25, 0.5, 0.75, 1\}$. The possible number of values for θ will be $5 \times 5 = 25$. This means in phase 1, parties will have to compute 25 log-likelihood values instead of 15. This will result in increase of computation time for both the collaborating parties and the aggregator.

4.4.3 Aggregating Encrypted Computation from Parties

In this step, $\mathcal{A}$ aggregates the Paillier encrypted results sent by the collaborating parties for each possible parameter samples selected by $\mathcal{A}$ in the prior step. Let's assume, a possible value of the parameter θ is mapped to index i. Then the sum of the encrypted log-likelihood values received from the parties is stored in the server as

ΣH_i. This is illustrated in the step 3 of Figure 4.2. For instance, step 2 of Figure 4.2 shows that $\theta = (0, 0.5)$ is mapped to index 8. Consequently, the encrypted sum of the log-likelihood values computed by parties for this particular value of θ is stored as ΣH_8 (step 3 of Figure 4.2).

4.4.4 Selecting Next Set of Candidates for Parameter Space

In this step, $\mathcal{A}$ compares the output of all the parameters it sent to the collaborating parties. After comparison, $\mathcal{A}$ chooses the parameters with the highest outputs to construct the parameter space of the next phase.

Let's continue with the example from Figure 4.2. In $\mathcal{A}$'s view, parameter $\theta = (0, 0.5)$ is mapped to 8 and $\theta = (-2, 1)$ is mapped to 1. Now, $\mathcal{A}$ has to figure out whether ΣH_8 is greater than ΣH_1 or the opposite. Therefore, $\mathcal{A}$ subtracts ΣH_8 from ΣH_1. $\mathcal{A}$ sends this subtraction result to the csp λ and queries about the sign of the subtraction result. λ decrypts the result (since λ has private key S_k) and responds to the query with the sign of the decrypted subtraction result. Based on the query response from λ, aggregator $\mathcal{A}$ formulates an optimization strategy. It is important to note that this strategy is **value-blind**. $\mathcal{A}$ only knows about the sign of the decrypted subtraction result. $\mathcal{A}$ does not know anything about the numerical value of the subtraction result of ΣH_8 and ΣH_1. The detailed execution of this step is described in the following paragraphs.

Creating Comparison Matrix. Let us assume that $\mathcal{A}$ selected K possible candidate parameters for θ in a particular phase. $\mathcal{A}$ initializes a $K \times K$ matrix. We denote

this matrix as comparison matrix, C. For the phase shown in Figure 4.2, $\mathcal{A}$ selected 15 possible candidate parameters for θ. Therefore, the size of the comparison matrix is 15×15 for this phase which is shown in step 4(a) of Figure 4.2. This comparison matrix allow the server to identify which are the most promising points for construction of the parameter space in the next phase. The entries of the comparison matrix are determined by Equation 4.1.

$$C_{ij} = \begin{cases} 1, & \text{if } i \neq j \text{ and } \Sigma H_i \geq \Sigma H_j \\ -1, & \text{if } i \neq j \text{ and } \Sigma H_i < \Sigma H_j \\ 0, & \text{if } i = j \end{cases} \tag{4.1}$$

Populating Comparison Matrix. To populate an entry C_{ij} in the comparison matrix, first $\mathcal{A}$ subtracts ΣH_i from ΣH_j. $\mathcal{A}$ sends the subtraction result to csp λ and queries about the sign of the subtraction result. λ decrypts the subtraction result and responds to $\mathcal{A}$'s query with the sign (positive or negative) of the decrypted subtraction result. Based on the response and Equation 4.1, $\mathcal{A}$ populates the C_{ij}. $\mathcal{A}$ performs this operation for all entries in C.

For example, $\mathcal{A}$ wants to populate C_{12}. $\mathcal{A}$ computes $(\Sigma H_1 - \Sigma H_2)$. $\mathcal{A}$ makes a query to λ regarding the sign of $(\Sigma H_1 - \Sigma H_2)$. λ responds that the sign is positive. $\mathcal{A}$ correlates the response with Equation 4.1 and sets C_{12} to 1 (Since positive sign means ΣH_1 is greater than ΣH_2). Next, $\mathcal{A}$ wants to populate C_{34}. $\mathcal{A}$ computes $(\Sigma H_3 - \Sigma H_4)$ and repeats the query to λ with $(\Sigma H_3 - \Sigma H_4)$. λ responds with negative. $\mathcal{A}$ then sets C_{34} to -1. $\mathcal{A}$ repeats this process for all entries in C.

Algorithm 1 FedGLMM: Secure Federated Training of GLMM

Require: D_i is dataset held by party i. (P_k, S_k) is the Paillier encryption key generated by csp λ at the initialization.

function SECURETRAINING(Ω)
  $f, P_k, S_k, n, d, \Theta_0, \chi \leftarrow$ initialize ▷ χ is the finishing condition
  $i \leftarrow 1$ ▷ n is # of parties, d is sampling interval, f is distribution to model
  while χ is false **do** ▷ Ω is adaptive shrinking parameter
    $\bar{\Delta} \leftarrow$ SAMPLEPARAMS(Θ_{i-1}, d)
    $\bar{\Sigma} \leftarrow$ SECUREAGGREGATE($\bar{\Delta}, n$)
    $\bar{C} \leftarrow$ CREATECOMPARISONMATRIX($\bar{\Sigma}$, length($\bar{\Sigma}$))
    $\Theta_i \leftarrow$ NEXTPARAMCANDIDATES($\bar{C}$, length($\bar{\Sigma}$), Ω)
    $i \leftarrow i + 1$
    if $\Omega \geq 0.1$ **then**
      $\Omega \leftarrow \Omega - 0.1$ ▷ Reducing parameter space shrink rate by 10%
    else
      $\Omega \leftarrow 0.1$ ▷ Keeping the shrink rate fixed otherwise
    end if
    $\chi \leftarrow$ check finishing condition
  end while
end function

function SECUREAGGREGATE(Θ, N)
  $\Sigma \leftarrow \emptyset$ ▷ Θ is the set of candidate parameters
  for all i, θ in Θ **do** ▷ N is number of parties.
    $S \leftarrow 0$ ▷ i is candidate parameter index.
    for all j in N **do**
      $S \leftarrow S +$ ENCRYPTEDLIKELIHOOD(f, θ) ▷ Party #j
    end for
    $\Sigma[i] \leftarrow S$
  end for ▷ This function is called by $\mathcal{A}$
  return Σ
end function

Selecting Points for Next Phase. $\mathcal{A}$ needs to find the candidate parameters which resulted in higher objective function output among the all candidates of θ selected for a phase. In order to do that, $\mathcal{A}$ adds up the values in each row after populating the comparison matrix (score column of step 4(a) in Figure 4.2). Each

Algorithm 2 FedGLMM: Secure Federated Training of GLMM (Continued)

function SAMPLEPARAMS(Θ, d)
    $\square \leftarrow$ Construct a convex hull from points in Θ
    $h, l \leftarrow$ get max and min of all dimensions in $\square$
    $\Delta \leftarrow$ Sample points evenly for d interval between h and l
    return Δ
end function

function CREATECOMPARISONMATRIX(Σ, K)
    $C \leftarrow \emptyset$
    for all i in K **do**
        for all $j \neq i$ in K **do**
            $C[i][j] = \text{sign}(\Sigma[i] - \Sigma[j])$ ▷ Resolved by λ
        end for ▷ This function is called by $\mathcal{A}$
    end for
    return C
end function

function NEXTPARAMCANDIDATES(C, K, k)
    $S \leftarrow \emptyset$
    for all i in K **do** ▷ K is # of parameters in C
        $S[i] \leftarrow \text{sum}(C[i])$ ▷ Scores for candidate parameters.
    end for
    $S \leftarrow \text{sort}(S)$
    $S \leftarrow$ top k% of S ▷ Keeping the top k% adaptively.
    return S ▷ This function is called by $\mathcal{A}$
end function

function ENCRYPTEDLIKELIHOOD(f, θ)
    $l \leftarrow 0$ ▷ f is the pdf of the distribution
    for all d_i in D **do** ▷ θ are the parameters of pdf
        $l \leftarrow l +$ `likelihood`$(f, \theta | d_i)$ ▷ D is the private dataset
    end for ▷ This function is called by parties
    return ENCRYPT(l, P_k)
end function

row represents a candidate parameter. For example, row 1 represents the candidate parameter mapped to index 1, row 2 represents the candidate parameter mapped to index 2 and so on.

The rows are then sorted by $\mathcal{A}$ based on the score column (descending sort). $\mathcal{A}$ selects the top $k\%$ of the candidate parameters for construction of parameter space in the next phase. For example, let's refer to the step 4(b) of Figure 4.2. $\mathcal{A}$ selects top 60% parameters from these phase for parameter space construction in next phase. The parameters that are selected are $(-2, 1), (-2, 0.5), (-1, 0.5), (-1, 1)$, $(0, 1), (0, 0.5), (1, 0.5), (1, 1), (2, 1)$. The resulting parameter space in the next phase is shown in the last step of Figure 4.2.

It is important to understand the role of k as a **hyperparameter** in FedGLMM. k controls how the parameter space shrinks phase by phase. If k is too small (e.g. 10%), then the parameter space shrinks too rapidly. This can cause poor generalization and high error rate. On the other hand, if k is too large (e.g. 80%), then the parameter space shrinks too slow. This can result in slow convergence and higher computation time and cost.

In order to mitigate these effects, we propose a phase by phase adaptive update of k. We linearly reduce k in each phase until k reaches a threshold value. For example, in phase 1, we select $k = 60\%$ candidate parameters for phase 2. In phase 2, we select $k = 50\%$ candidate parameters for phase 3 and so on. We stop reducing k once it reaches 10%. For the remaining phases, k remains at 10% until the training process terminates.[5]

[5]To get a high level overview, consider this: FedGLMM is a multi phase process. In each phase, a grid search is performed (The grid is a convex hull in our case). However, instead of plain numerical values, the grid search is performed on encrypted values. Therefore, FedGLMM is a **value-blind optimization** process.

4.4.5 Termination

There are two terminating conditions in FedGLMM scheme.

- The training process found the optimum parameter (An indication of this is when the l_2 distance between candidate parameters selected for next phase is close to zero.).
- The parameter space has become very small. (e.g., less than 1% of the parameter space of the initial phase)

In both cases, $\mathcal{A}$ follows the procedure described in 4.4.4. First, $\mathcal{A}$ creates a comparison matrix. Then $\mathcal{A}$ populates it and sorts them. Instead of selecting a set of parameter samples, $\mathcal{A}$ selects a singular parameter sample that is at the top of the sorted order. After that, $\mathcal{A}$ shares this selected candidate parameter with the collaborating parties and the training terminates. Algorithm 1 presents the complete execution process of FedGLMM scheme.

4.5 Security Analysis

We can prove the security of our proposed scheme by designing a soundness game which can be shown to have negligible probability (in a security parameter) of being won by an adversary.

Definition 1 *(Soundness).* Let us define a game SOUND[ρ] between adversary A and challenger C. The role of A can be assumed by any *one* of the following: aggregator, csp, subset of collaborating parties. The role of C is assumed by the

honest collaborating parties. ρ is a security parameter. The steps of the game are as follows:

- The csp generates homomorphic public key-private key pair (P_k, S_k) and shares P_k with aggregator and the parties.
- Adversary A stores all the intermediate data during the execution of f_M and the final model M.

There are two ways where the adversary wins the game.

- If the adversary A is either the aggregator or the csp, then A can win the game if it can infer any point or statistics of the dataset D_i held by party P_i.
- If the adversary A is a subset of the collaborating parties, then A can win the game if it can infer any any point or statistics of the dataset D_i of P_i where P_i does not belong in the subset of the parties who are adversary.

We say a training scheme is *sound* if no probabilistic polynomial time adversary can win the SOUND[ρ] game described above with greater than negligible probability in the security parameter ρ.

Claim. *FedGLMM scheme is sound.*

Proof. The soundness proof follows closely from the construction of the FedGLMM protocol. We consider two cases. The first case is the execution period of f_M in FedGLMM.

- If the adversary A is aggregator $\mathcal{A}$, then the adversary receives the encrypted intermediate log-likelihood values from the parties. Adversary also receives the sign of the values of comparison matrix from the λ. Note that our threat model assumes no collusion between $\mathcal{A}$ and λ. Therefore, to retrieve the private dataset D_i or related statistics which are used to compute the log-likelihood value, $\mathcal{A}$ has to correctly guess the private key S_k (held by csp λ). The probability of correctly guessing a ρ bit private key is $2^{-\rho}$. We set the key size in FedGLMM scheme to be multiple of ρ. Let's assume the multiplication factor is $m \in \mathbb{N}$. Therefore, probability of guessing the private key correctly is $2^{-m\rho}$ which is negligible[6].

- If the adversary A is csp λ, then the adversary can decrypt the aggregated intermediate log-likelihood values sent by the parties which it receives from the aggregator $\mathcal{A}$. Recall that our threat model has no collusion between $\mathcal{A}$ and λ. Consequently, to retrieve the private dataset D_i or related statistics which are used to compute the log-likelihood value, λ has to correctly *guess* the parameters of the distribution. The probability of correctly guessing n parameters of the distribution is $2^{-n\rho}$, which is negligible[7].

Now, we move to the second case, which is after the end of f_M execution. In this case, the adversary have access to both the data collected during f_M execution and final output M. The two arguments above (adversary is either $\mathcal{A}$ or λ) also hold in this case. However, there is an additional adversary case here. We have to consider

[6] In our instantiation of FedGLMM we used 128 bit key where the $\rho = 64$ and $m = 2$.

[7] In our instantiation of the FedGLMM scheme the parameters are mean and standard deviation, which amounts to $n = 2$ and $\rho = 64$. We set ρ to the precision of the floating point number used in implementation for convenience.

the case where the adversary A is a subset of collaborating parties $\{P_1, P_2, \cdots, P_j\}$ where, $j \in [1, n-1]$ or outsiders.

- Let's assume that there are n collaborating parties and adversary A controls $(n-1)$ of them. The party P_n is honest. Based on M, adversary A has to compute the parameters of a specific probability distribution function $f(D_n)$ such that superposition of $f(D_1 \oplus D_2 \cdots \oplus D_{n-1})$ and $f(D_n)$ satisfies M. This problem does not have any known analytical solution. Therefore, from adversarial perspective, A has to accurately guess the parameters of $f(D_n)$. The probability of correctly guessing n parameters of the distribution is $2^{-n\rho}$, which is negligible.

- Let us assume that the adversary A is a malicious outsider. A intercepted the encrypted log-likelihood values sent to $\mathcal{A}$ by P_i. To infer the data in D_i, adversary A must decrypt the encrypted log-likelihood values. To accomplish this, A must guess the S_k held by λ. The probability of A accurately guessing S_k is $2^{-m\rho}$ which is negligible.

Thus the adversary can only win the game during execution of f_M or after the execution of f_M with a negligible probability in ρ. Therefore, the soundness of FedGLMM scheme is proven.■

4.6 Experimental Results

In this section, we investigate the experimental performance of FedGLMM. It is important to understand that GLMM is universally used in clinical and biomedical domain. Therefore, our analysis will take clinical utility into account. Clinical utility is different than statistical utility which we discuss more in Section 4.7. We aim to address the following research questions:

Q1. Does FedGLMM achieve clinically acceptable low error-rate under different hyperparameter settings? How do different hyperparameter settings affect error rate of FedGLMM execution?(section 4.6.2)

Q2. What is the distribution of execution time of FedGLMM? How do different hyperparameter settings affect the execution time?(section 4.6.3)

4.6.1 Dataset and Execution Environment

We experimented on the performance of FedGLMM on two datasets. The first one is a real world dataset. It contains the clinical trial data for testing a cancer detection method from 103 hospitals (parties). The second one is a synthetic dataset. We generated this dataset to have different distribution of prevalence, sensitivity and specificity than the real world dataset. It contains the data sample from 500 hospitals (parties). We denote the real world dataset as D_r and the synthetic dataset as D_s. Table 4.2 shows the overview of the datasets. In both cases, parties try to model a GLMM on logit-normal distribution in a distributed manner.

The machine used for experiment is a MacOS Mojave machine with intel core-i7

Table 4.2: Overview of the Datasets Used in Experiment

Data set	# of parties	prevalence		sensitivity		specificity	
		mean	std.dev	mean	std.dev	mean	std.dev
D_r	103	0.38	0.25	0.69	0.3	0.88	0.14
D_s	500	0.41	0.46	0.52	0.31	0.70	0.24

processor (clock speed 2.66 GHz) and 8 gigabytes of memory. The hospitals (parties) were simulated by python clients. Since the server and parties were situated in the same machine, the communication latency between server and parties were minimal.

4.6.2 Error Percentage

In this section we vary the hyperparameters of FedGLMM and observe the effect on the error percentage of FedGLMM output. We show that FedGLMM achieves very low error rate (less than 5%) in both datasets.

Phase Count. We observe the effect of phase count variation in FedGLMM. The results are shown in Figure 4.3. During experimenting with phase count variation, we fixed the k to adaptive setting and initial parameter space was set to optimal setting. Both of these are explained in later parts of this section.

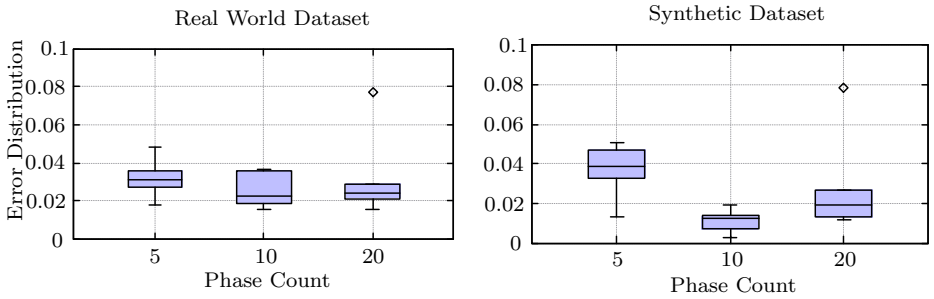


Figure 4.3: Error Distribution in our proposed training method for different number of phases

We divide our experiments in 3 classes: When the training completes within 5 phases, within 10 phases and within 20 phases. When the training finishes too early for the D_r (within 5 phases), the average error rate is around 3%. On the other hand when the training ends within 10 phases, the error rate is slightly above 2%. If the training goes up to 20 phases, the average error does not change significantly. Experiments on D_s shows similar pattern of low error rates.

Variation of k. In section 4.4.4, we stated that k is a important hyperparameter in FedGLMM. k controls the reduction of parameter space in the beginning of each phase. We experimented FedGLMM by setting very high $k = 80\%$, very low $k = 15\%$ and adaptive k adjustment starting at 50% and reducing by 10% in subsequent phases. The effect of each setting on the error percentage is shown in Figure 4.4. From the Figure 4.4, we can observe that the proposed adaptive k adjustment

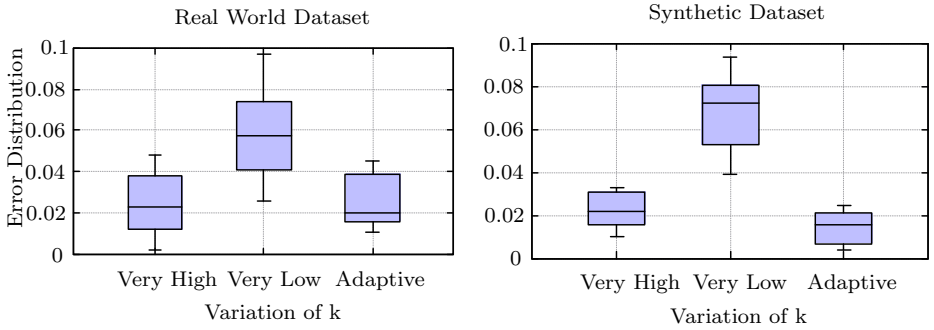


Figure 4.4: Error Distribution in our proposed training method for different settings of k

mechanism outperforms both high k and low k settings for this experiment (less than 2% error). When the k is low, FedGLMM converges prematurely for both datasets. This premature convergence results in very high error rate which is clinically not

acceptable. On the other hand, when k is high, the error is acceptable (marginally above 2%). However, it takes a long time for the training process to reach that low error rate which is computationally costly.

Initial Parameter Space Construction. An important hyperparameter of our proposed approach is initial the parameter space construction. If the initial parameter space is too big, $\mathcal{A}$ will have to run more phases to reach the optimization objective. On the other hand, if the initial parameter space is too small, $\mathcal{A}$ may generalize poorly and have high error rates.

To demonstrate this, we run our experiment with 3 different settings. In the first setting, we set the range of d_1 to $[-4, 4]$ and d_2 to $[0, 1]$. In the second setting, we set the range of d_1 to $[-2, 1]$ and d_2 to $[0, 0.5]$. Finally, in the third setting, we set the range of d_1 to $[-1, 0.5]$ and d_2 to $[0, 0.1]$. The first, second and third settings correspond to big, optimal and small initial parameter space respectively. The experiment results are shown in Figure 4.5.

Figure 4.5: Error Distribution in our proposed training method for different settings of initialization

We observe that error rate remains under 5% (clinically acceptable) for settings 1 and 2. Settings 1 required more phases than settings 2 for converging to the optimized objective function. Therefore, settings 1 required more computational cost. On the other hand, settings 3 unperformed in all of our experiments. FedGLMM generalized very poorly for settings 3 and sometimes resulted in high error rate as much as 20% which is completely unacceptable in clinical setting. We conclude that

settings 2 is the best candidate for initial parameter space construction. Note that, settings 2 is optimal only for this particular distribution modeling (logit-normal). If a different distribution was chosen, the outcome could be different. Therefore, the aggregator $\mathcal{A}$ will have to explore the initial parameter space a few times to get the idea of optimality.

Other Factors. Another hyperparameter involved in our proposed training method is the sampling interval (4.4.2). Variation in sampling interval did not have any effect on the error percentage of our experiments. However, it did have effect on the total execution time of a phase which is discussed in the next subsection.

4.6.3 Execution Time

We observed that there are two hyperparameters in our proposed training method which have effect on the total execution time of the optimization. They are phase count and sampling interval.

Phase Count. The greater the number of phases required to optimize the function, the greater the required execution time in FedGLMM. Figure 4.6 shows the experiment results when FedGLMM phase count varies. During these experiments the initial parameter space was set to setting 2 (optimal) and k was reduced linearly as described in Section 4.4.4. Figure 4.6 shows that our proposed method takes orders of magnitude more time compared to the baseline with no privacy. As number of phases increase, the time required also increases monotonically.

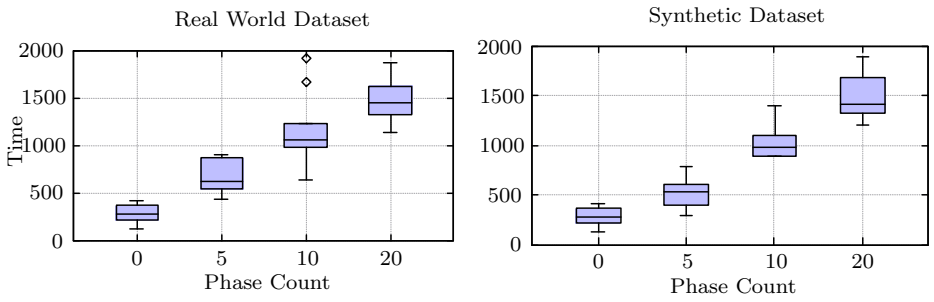


Figure 4.6: Execution Time (in seconds) required for various phase counts in our experiments. Phase count 0 stands for the time required for federated GLMM training without any privacy preserving method which is our baseline.

It is interesting to note that D_s is almost 5 times bigger than D_r in terms of the number of entries (Table 4.2). However, they take almost similar amount of time to complete training process. This can be attributed to the parallelized nature FedGLMM. The calculation of log-likelihood values by parties is a parallel process. All parties receive the parameters, compute the log-likelihood and send the encrypted result back at the same time period. Additionally, the populating the comparison matrix is also a parallel process. $\mathcal{A}$ makes a single query with all encrypted subtraction results to λ and λ can reply back in a single query. $\mathcal{A}$ and λ both internally employs vectorized calculation to compute respective parts. This is the reason why two datasets have the similar distribution of execution time for different phase counts while having different number of parties.

Sampling Interval. The sampling interval (Section 4.4.2) plays an important role in determining the total time required for a phase execution. This is because sampling interval determines how many possible candidate parameters are shared with parties in a phase. To demonstrate this, we fixed a phase and a specific dimension d_1 of θ

(Section 4.4.2). After that we tried different sampling intervals for d_1 and recorded the respective time required to complete that specific phase execution. The results are presented in Figure 4.7. It shows that as sampling interval becomes smaller, the required execution time of the phase goes higher. This can be attributed to the fact that when sampling interval becomes smaller, the number of possible parameters within the parameter space becomes higher. This causes $\mathcal{A}$ requiring more time to compute the sum of encrypted partial log-likelihood values as well as populating comparison matrix [8].

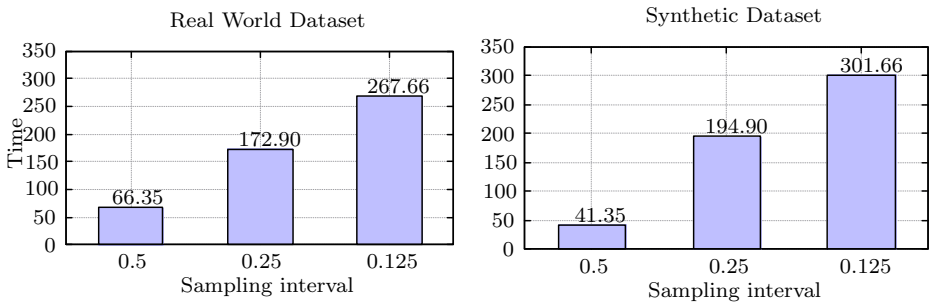


Figure 4.7: Time (in seconds) required for different sampling intervals in our experiments.

4.6.4 Additional Results

We chose 100 random subsets of 10 parties from both the synthetic and real-world datasets. The results of the experiments are shown in Figure 4.8. We can see that the results are consistent with the full dataset experiment with phase count 5 achieving higher error rate for poor generalization, phase count 10 achieving the least error and phase count 20 has comparable error margin with phase count 10. However, we

[8]Figure 4.7 is represented as a bar plot because, the data had too low standard deviation to be plotted as box plot (e.g. the time intervals for multiple runs were very close).

noticed an interesting phenomenon in the synthetic dataset experiment. The results have very low standard deviation for the synthetic dataset. The real world dataset has comparatively higher standard deviation for experimental error rate. This can be attributed to the fact that the real-world dataset has different distribution of values than the synthetic dataset. Since we are taking a small subset of just 10 parties, this distributional difference is being reflected in the experimental result.

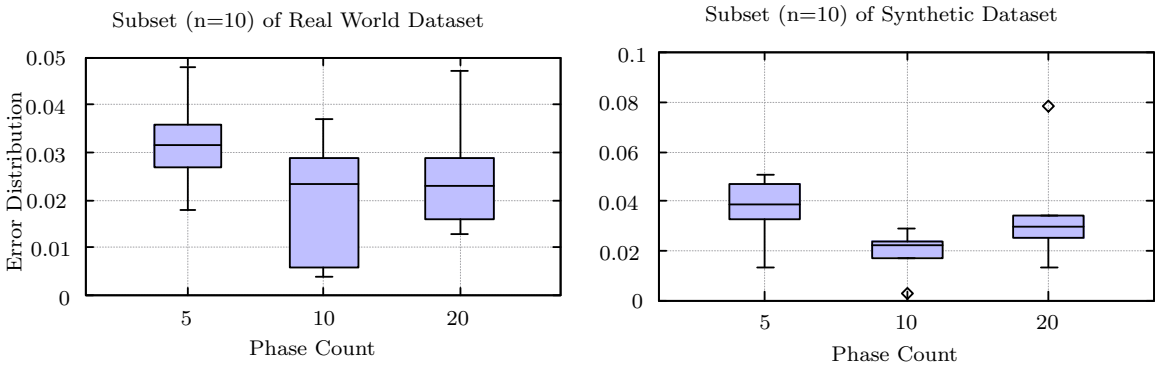


Figure 4.8: Experiment on 10 randomly selected parties from the dataset.

Similar to the above experiment, we chose 100 random subsets of 20 parties from both synthetic and real world datasets. The result of this experiment is shown in Figure 4.9. The outcomes of these experiments are consistent with the full dataset experiment. We note that experiments have a similar standard deviation. However, the synthetic dataset has a slightly higher error rate than real world dataset.

We performed the same experiment with 100 random subsets of 50 parties from both real world and synthetic datasets. The result of this experiment is shown in Figure 4.10. Similar to the prior experiments, the experiments with a subset of 50 parties have shown consistent results. In all cases, the optimal error rate is achieved

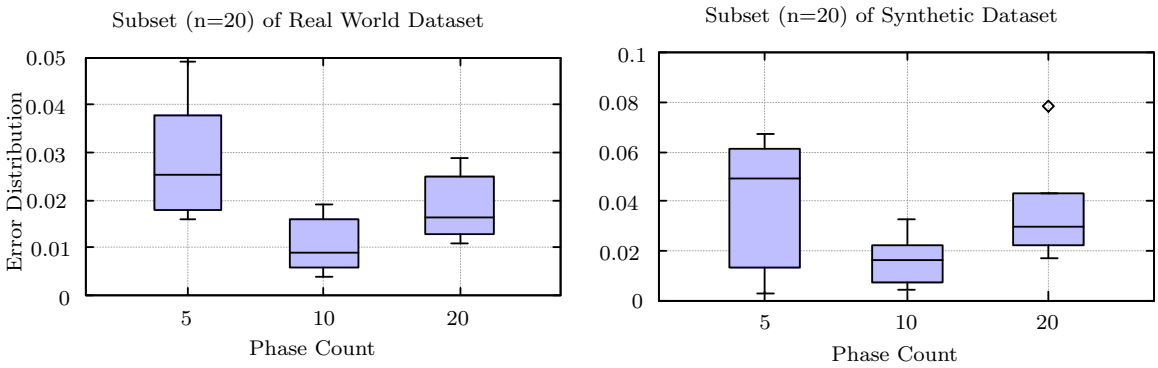


Figure 4.9: Experiment on 20 randomly selected parties from the dataset.

at phase count 10.

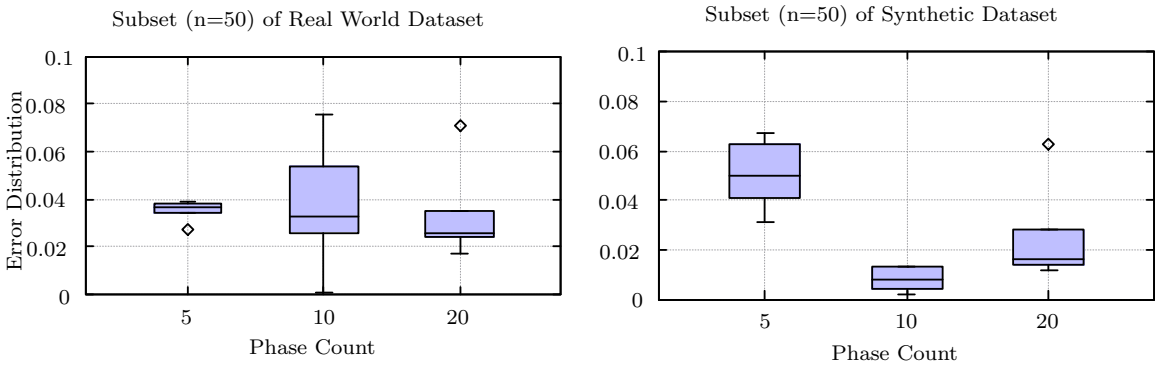


Figure 4.10: Experiment on 50 randomly selected parties from the dataset.

To summarize, the new experiments have shown that the results obtained by experimenting on a subset of the datasets are consistent with the results from the whole datasets. Therefore, our proposed method is effective for both small and large number of parties.

4.7 Discussion of Design Decisions

In this section, we further elaborate on FedGLMM design decisions. Specifically, why SMPC was chosen as underlying protocol instead of differential privacy for

FedGLMM design.

4.7.1 Statistical utility does not always translate to clinical utility

. In privacy preserving machine learning, the model training needs to be performed in a way that utility is preserved as well. The metric of utility is usually domain-specific. For instance, deep learning models trained on MNIST dataset can achieve upto 99% accuracy [61]. However, when privacy preserving mechanisms (i.e., federated learning, differential privacy) are applied on the training process, accuracy drops to some extent [62, 63]. These accuracy drops or error rate increase is acceptable in many real-life use cases such as (i.e., object-recognition, speech transcription). However, in sensitive application domains such as clinical and biomedical decision making algorithms, the acceptable margin of error induced by privacy preserving mechanisms is very low. Therefore, the utility metric of FedGLMM is to achieve as low as possible error rate to be clinically acceptable. Specifically, we consider the performance of FedGLMM to be clinically acceptable if it achieves less than 5% error rate, compared to the non privacy-preserving training scheme.

Let's elaborate this with a motivating example. Consider a GLMM for predicting the efficacy of a clinical test to determine cancer. The clinical test data are fit to distributions. Based on the distribution fit, it is decided whether the test can be used to rule out cancer or not. Let's assume that without any privacy preserving mechanism in place, the GLMM in question have 95% accuracy. If we use local

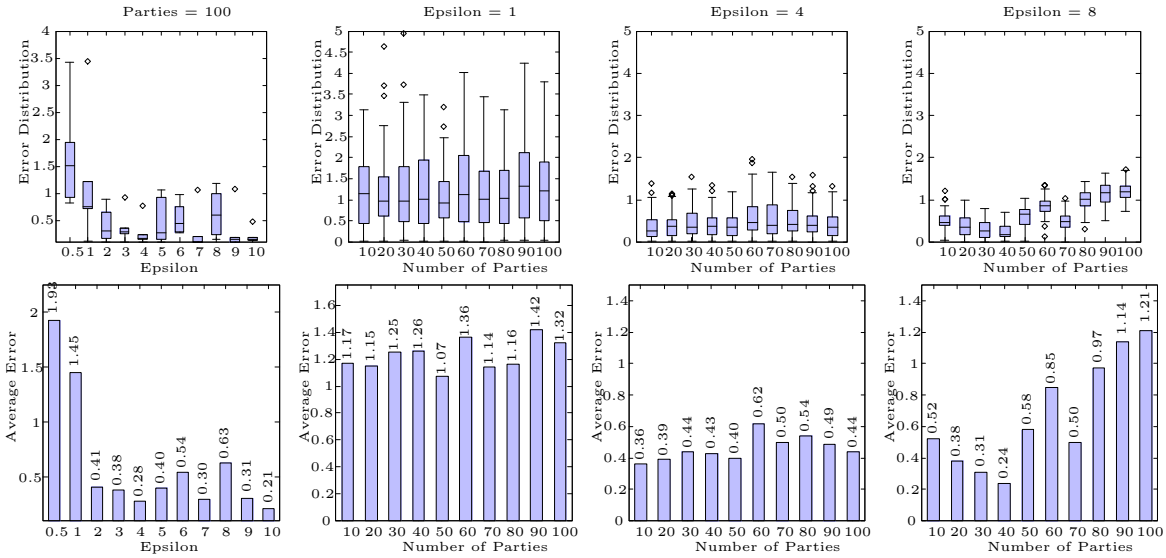


Figure 4.11: Experimentation with differential privacy as privacy preserving mechanism for FedGLMM.

differential privacy to train the same model, the accuracy can drop below 80% [64]. In other words, private GLMM modeling have 15% more error rate compared to the non-private counterpart. For highly sensitive medical applications, sometimes 15% error rate is above the acceptable margin of error.

4.7.2 Local Differential Privacy does not meet the utility requirement for federated GLMM training.

While designing FedGLMM, we had to make an important design choice. We needed to decide whether to use secure multi-party computation (SMPC) or local differential privacy (LDP) as the privacy-preserving mechanism. Initially, we experimented on FedGLMM using LDP. The error rate was very high in this case. We experimented with different $\epsilon \in (0, 10]$ values for different number of parties $n \in (0, 100]$.

The experimentation result is depicted in Figure 4.11. There are two observations that stand out in the result.

① Increase in ϵ did not result in the decrease of error-rate. As the epsilon values increased steadily, the distribution of errors did not decrease uniformly. Instead, the distribution of errors was lower at $\epsilon = \{4, 7, 10\}$ than the neighbors (Figure 4.11, bottom row, first column). Even at $\epsilon = 10$, the error rate was around 21% which is above the threshold of acceptable error for a number of clinical applications. This was one of the major reasons we preferred SMPC over LDP for FedGLMM design. This high error rate is rooted in the working principle of GLMM. In the parameter optimization part, GLMM uses bayesian optimization algorithm to find the next parameters. However, bayesian optimization technique is extremely sensitive to noise [65]. Therefore, the large noises injected by LDP quickly adds up which misleads the optimization and reduces utility of the model. Moreover, GLMM training is a communication intensive process. In our case, the synthetic dataset had very high variance and often required more than 100 rounds to converge. Therefore, the privacy budget ran out very quick. To have a GLMM training converge appropriately with LDP, the privacy budget would have to be significantly higher. At that point, LDP offers little to no meaningful privacy. This reinforces our observation that it is difficult for LDP to preserve the privacy-utility trade-off for a noise sensitive process like GLMM training.

② When greater number of parties are involved in GLMM training with LDP, the error rate steadily increase (Figure 4.11 column 2,3,4). This phenomenon can also

be attributed to the fact that increase in parties cause increase in dataset variance, which results in more communication rounds for the training to converge. More communication rounds translate to more LDP noise injected in the GLMM training and eventually the utility of the trained model is depleted. This again proves that LDP is not a suitable choice for privacy preserving GLMM training in federated setting. It is desirable that any privacy preserving GLMM training scheme should accommodate any number of collaborating parties in the training. The observations from Figure 4.11 suggest that LDP can not satisfy this property. Interestingly, [64] also faced similar phenomenon for healthcare data and drew similar conclusion.

Based on these observations, we decided to use SMPC for designing FedGLMM. It is important to note that we are not comparing SMPC and LDP as privacy-preserving approach. Both have specific use cases where they excel. We just presented our rationale for choosing SMPC as the privacy-preserving mechanism. It might be possible to construct special LDP mechanism that can achieve clinical utility as well as meaningful privacy. We will investigate this research direction in our future works.

4.7.3 Scalability of FedGLMM design.

Grid search is often prohibitively expensive when a fine grained resolution is considered [66]. Therefore, the choice of grid (convex hull) based optimization in FedGLMM can raise concern regarding scalability. We address this concern by adaptively shrinking the grid over each iteration. As iterations progress, we increase the granularity of our search. In other words, we start with large parameter search space

with sparse sampling. As phases progress, we gradually progress from sparse sampling to dense sampling. This prevents FedGLMM from getting stuck in local minima or from being computationally expensive.

4.7.4 Appropriate Use Case(s) of FedGLMM

It is important to understand that our proposed method provides rigorous security for any number of participants who want to learn a generalized linear mixed model on private datasets. However, there are cases where there is no need for such a strict security model, and in those cases, the assumptions can be relaxed. For instance, Table 1 contains aggregate statistics of clinical studies related to a specific disease. There can be two cases here.

The first case is where the disease or clinical condition in question is rare, and there is no auxiliary information here to mount homer's attack on the dataset from aggregate statistics. In this case, aggregate statistics is relatively secure in terms of leakage. Therefore, in these cases, secure training methods like ours may not be necessary. However, in the second case, secure training is absolutely necessary.

The second case is where there is auxiliary information available for inferring patient level data from aggregate statistics. This is demonstrated by Wang et al. in [47] for sporadic postmenopausal breast cancer. They have shown that given auxiliary information, they can effectively extract patient-level SNP information from published aggregate statistics. This is an example of an appropriate use case of our proposed secure training method.

Chapter 5

Conclusion

In this thesis we proposed and designed an end-to-end privacy preserving contact tracing data collection and analysis framework. We designed the framework in a way that takes privacy into account in every single design decisions. Experimental evaluation on real-world and synthetic datasets suggest that our proposed framework is robust and accurate to be useful in real-world application scenario. This two step framework with privacy-aware design is widely applicable to numerous real-world problem instances. We believe that our work in this domain will inspire researchers to further work on these problem that involve privacy-aware solution design. However, there are some limitations of this thesis which requires deeper exploration. For instance, the prevalence of GLMM as an algorithm is limited to biomedical data analysis. Several other domain-specific algorithms exist. Each of these algorithms warrant their own systematic privacy analysis. This is an interesting research direction that we intend to further explore in future. Another interesting problem is the adversarial robustness of these privacy-preserving measures. How much resilient to

adversarial manipulation they are and what are the possible preventive measures? A probable way to approach this problem will be to generate adversarial perturbations to the input and quantifying the output degradation. Lastly, this thesis uses secure multi-party computation and homomorphic encryption as the building block of privacy preserving mechanisms. However, these blocks are computationally expensive. On the other hand, differential privacy is a robust privacy-preserving mechanism with possible degraded utility. It remains an interesting research question whether it is possible to design specialized differentially private mechanism to achieve the same level of utility as secure-multi-party computation. We intend to explore this particular problem in future as well.

Bibliography

[1] R. Raskar, I. Schunemann, R. Barbar, K. Vilcans, J. Gray, P. Vepakomma, S. Kapa, A. Nuzzo, R. Gupta, A. Berke *et al.*, "Apps gone rogue: Maintaining personal privacy in an epidemic," *arXiv preprint arXiv:2003.08567*, 2020. 3, 20

[2] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava, "A survey on security and privacy of federated learning," *Future Generation Computer Systems*, vol. 115, pp. 619–640, 2021. 4

[3] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 3–18. 4, 37, 41

[4] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. Quek, and H. V. Poor, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3454–3469, 2020. 4, 41

[5] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," *CoRR*, vol. abs/1712.07557, 2017. [Online]. Available: http://arxiv.org/abs/1712.07557 4

[6] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, "Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning," in *2020 {USENIX} Annual Technical Conference ({USENIX}{ATC} 20)*, 2020, pp. 493–506. 4, 36

[7] V. Nikolaenko, S. Ioannidis, U. Weinsberg, M. Joye, N. Taft, and D. Boneh, "Privacy-preserving matrix factorization," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013, pp. 801–812. 4

[8] ——, "Privacy-preserving matrix factorization," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013, pp. 801–812. 4

[9] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, and Y. Zhou, "A hybrid approach to privacy-preserving federated learning," in *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, 2019, pp. 1–11. 4

[10] W. D. Flanders and D. G. Kleinbaum, "Basic models for disease occurrence in epidemiology," *International journal of epidemiology*, vol. 24, no. 1, pp. 1–7, 1995. 4

[11] Y. Chen, Y. Liu, J. Ning, J. Cormier, and H. Chu, "A hybrid model for combining case-control and cohort studies in systematic reviews of diagnostic tests," *Journal of the Royal Statistical Society. Series C, Applied statistics*, vol. 64, no. 3, p. 469, 2015. 4, 12, 14

[12] X.-Y. Song and S.-Y. Lee, "Model comparison of generalized linear mixed models," *Statistics in medicine*, vol. 25, no. 10, pp. 1685–1698, 2006. 5

[13] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of cryptography conference*. Springer, 2006, pp. 265–284. 9

[14] C. Gentry, A. Sahai, and B. Waters, "Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based," in *Annual Cryptology Conference*. Springer, 2013, pp. 75–92. 11

[15] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *International conference on the theory and applications of cryptographic techniques*. Springer, 1999, pp. 223–238. 11

[16] P. Grubbs, M.-S. Lacharité, B. Minaud, and K. G. Paterson, "Pump up the volume: Practical database reconstruction from volume leakage on range queries," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 315–331. 13

[17] S. Garfinkel, J. M. Abowd, and C. Martindale, "Understanding database reconstruction attacks on public data: These attacks on statistical databases are no longer a theoretical danger." *Queue*, vol. 16, no. 5, pp. 28–53, 2018. 13

[18] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," *arXiv preprint arXiv:1206.2944*, 2012. 16

[19] N. Trieu, K. Shehata, P. Saxena, R. Shokri, and D. Song, "Epione: Lightweight

contact tracing with strong privacy," *arXiv preprint arXiv:2004.13293*, 2020. 19, 29, 34

[20] *DP-3T*, 2020 (accessed June 3, 2020), https://github.com/DP-3T/documents. [Online]. Available: https://github.com/DP-3T/documents 19

[21] *PEPP-3T*, 2020 (accessed June 3, 2020), https://www.pepp-pt.org. [Online]. Available: https://www.pepp-pt.org 19

[22] T. Altuwaiyan, M. Hadian, and X. Liang, "Epic: efficient privacy-preserving contact tracing for infection detection," in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018, pp. 1–6. 20

[23] A. B. Dar, A. H. Lone, S. Zahoor, A. A. Khan, and R. N. Mir, "Applicability of mobile contact tracing in fighting pandemic (covid-19): Issues, challenges and solutions." *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 484, 2020. 20

[24] *Seoul's Radical Experiment in digital contact tracing*, 2020 (accessed August 21, 2020), https://www.newyorker.com/news/news-desk/seouls-radical-experiment-in-digital-contact-tracing. [Online]. Available: https://www.newyorker.com/news/news-desk/seouls-radical-experiment-in-digital-contact-tracing 20

[25] J. Bay, J. Kek, A. Tan, C. S. Hau, L. Yongquan, J. Tan, and T. A. Quy, "Bluetrace: A privacy-preserving protocol for community-driven contact tracing across borders," *Government Technology Agency-Singapore, Tech. Rep*, 2020. 20

[26] A. Berke, M. Bakker, P. Vepakomma, R. Raskar, K. Larson, and A. Pentland, "Assessing disease exposure risk with location histories and protecting privacy:

A cryptographic approach in response to a global pandemic," *arXiv preprint arXiv:2003.14412*, 2020. 20

[27] J. Chan, S. Gollakota, E. Horvitz, J. Jaeger, S. Kakade, T. Kohno, J. Langford, J. Larson, S. Singanamalla, J. Sunshine *et al.*, "Pact: Privacy sensitive protocols and mechanisms for mobile contact tracing," *arXiv preprint arXiv:2004.03544*, 2020. 20

[28] *Apple and Google releases contact tracing API's for building mobile apps*, 2020 (accessed June 3, 2020), https://www.apple.com/newsroom/2020/04/apple-and-google-partner-on-covid-19-contact-tracing-technology. 21

[29] H. Alsdurf, Y. Bengio, T. Deleu, P. Gupta, D. Ippolito, R. Janda, M. Jarvie, T. Kolody, S. Krastev, T. Maharaj *et al.*, "Covi white paper," *arXiv preprint arXiv:2005.08502*, 2020. 21

[30] *CDC guideline*, 2020 (accessed Aug 22, 2020), https://www.cdc.gov/coronavirus/2019-ncov/prevent-getting-sick/social-distancing.html. [Online]. Available: https://www.cdc.gov/coronavirus/2019-ncov/prevent-getting-sick/social-distancing.html 22, 27

[31] G. Giordano, F. Blanchini, R. Bruno, P. Colaneri, A. Di Filippo, A. Di Matteo, and M. Colaneri, "Modelling the covid-19 epidemic and implementation of population-wide interventions in italy," *Nature Medicine*, pp. 1–6, 2020. 25, 27

[32] X. He, E. H. Lau, P. Wu, X. Deng, J. Wang, X. Hao, Y. C. Lau, J. Y. Wong, Y. Guan, X. Tan *et al.*, "Temporal dynamics in viral shedding and transmis-

sibility of covid-19," *Nature medicine*, vol. 26, no. 5, pp. 672–675, 2020. 26, 28

[33] F. Sattler, J. Ma, P. Wagner, D. Neumann, M. Wenzel, R. Schäfer, W. Samek, K.-R. Müller, and T. Wiegand, "Risk estimation of sars-cov-2 transmission from bluetooth low energy measurements," *arXiv preprint arXiv:2004.11841*, 2020. 27

[34] J. Cao, Q. Xiao, G. Ghinita, N. Li, E. Bertino, and K.-L. Tan, "Efficient and accurate strategies for differentially-private sliding window queries," in *Proceedings of the 16th International Conference on Extending Database Technology*, 2013, pp. 191–202. 30

[35] M. Hay, V. Rastogi, G. Miklau, and D. Suciu, "Boosting the accuracy of differentially-private histograms through consistency," *arXiv preprint arXiv:0904.0942*, 2009. 30

[36] A. Poulose, O. S. Eyobu, and D. S. Han, "An indoor position-estimation algorithm using smartphone imu sensor data," *IEEE Access*, vol. 7, pp. 11 165–11 177, 2019. 35

[37] Y. Aono, T. Hayashi, L. Trieu Phong, and L. Wang, "Scalable and secure logistic regression via homomorphic encryption," in *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, 2016, pp. 142–144. 36

[38] J. Li and H. Huang, "Faster secure data mining via distributed homomorphic

encryption," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 2706–2714. 36

[39] H. Chen, R. Gilad-Bachrach, K. Han, Z. Huang, A. Jalali, K. Laine, and K. Lauter, "Logistic regression over encrypted data from fully homomorphic encryption," *BMC medical genomics*, vol. 11, no. 4, pp. 3–12, 2018. 36

[40] J. L. Crawford, C. Gentry, S. Halevi, D. Platt, and V. Shoup, "Doing real work with fhe: the case of logistic regression," in *Proceedings of the 6th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, 2018, pp. 1–12. 36

[41] F. Bergamaschi, S. Halevi, T. T. Halevi, and H. Hunt, "Homomorphic training of 30,000 logistic regression models," in *International Conference on Applied Cryptography and Network Security*. Springer, 2019, pp. 592–611. 36

[42] Y. Aono, T. Hayashi, L. Wang, S. Moriai *et al.*, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1333–1345, 2017. 36

[43] A. QaisarAhmadAlBadawi, J. Chao, J. Lin, C. F. Mun, S. J. Jie, B. H. M. Tan, X. Nan, A. M. M. Khin, and V. Chandrasekhar, "Towards the alexnet moment for homomorphic encryption: Hcnn, the first homomorphic cnn on encrypted data with gpus," *IEEE Transactions on Emerging Topics in Computing*, 2020. 36

[44] K. Nandakumar, N. Ratha, S. Pankanti, and S. Halevi, "Towards deep neural

network training on encrypted data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0. 36

[45] A. Wood, K. Najarian, and D. Kahrobaei, "Homomorphic encryption for machine learning in medicine and bioinformatics," *ACM Computing Surveys (CSUR)*, vol. 53, no. 4, pp. 1–35, 2020. 37

[46] N. Homer, S. Szelinger, M. Redman, D. Duggan, W. Tembe, J. Muehling, J. V. Pearson, D. A. Stephan, S. F. Nelson, and D. W. Craig, "Resolving individuals contributing trace amounts of dna to highly complex mixtures using high-density snp genotyping microarrays," *PLoS Genet*, vol. 4, no. 8, p. e1000167, 2008. 37

[47] R. Wang, Y. F. Li, X. Wang, H. Tang, and X. Zhou, "Learning your identity and disease from research papers: information leaks in genome wide association study," in *Proceedings of the 16th ACM conference on Computer and communications security*, 2009, pp. 534–544. 37, 72

[48] R. Wolfinger and M. O'connell, "Generalized linear mixed models a pseudo-likelihood approach," *Journal of statistical Computation and Simulation*, vol. 48, no. 3-4, pp. 233–243, 1993. 38

[49] C. E. McCulloch, "Maximum likelihood algorithms for generalized linear mixed models," *Journal of the American statistical Association*, vol. 92, no. 437, pp. 162–170, 1997. 38

[50] H. Chu and S. R. Cole, "Bivariate meta-analysis of sensitivity and specificity

with sparse data: a generalized linear mixed model approach," *Journal of clinical epidemiology*, vol. 59, no. 12, p. 1331, 2006. 38

[51] E. F. Vonesh, H. Wang, L. Nie, and D. Majumdar, "Conditional second-order generalized estimating equations for generalized linear and nonlinear mixed-effects models," *Journal of the American Statistical Association*, vol. 97, no. 457, pp. 271–283, 2002. 38

[52] M. Casals, M. Girabent-Farres, and J. L. Carrasco, "Methodological quality and reporting of generalized linear mixed models in clinical medicine (2000–2012): a systematic review," *PloS one*, vol. 9, no. 11, p. e112653, 2014. 38

[53] B. Zheng and A. Agresti, "Summarizing the predictive power of a generalized linear model," *Statistics in medicine*, vol. 19, no. 13, pp. 1771–1781, 2000. 38

[54] L. Wang, P. Jia, R. D. Wolfinger, X. Chen, B. L. Grayson, T. M. Aune, and Z. Zhao, "An efficient hierarchical generalized linear mixed model for pathway analysis of genome-wide association studies," *Bioinformatics*, vol. 27, no. 5, pp. 686–692, 2011. 38

[55] W. Zhou, Z. Zhao, J. B. Nielsen, L. G. Fritsche, J. LeFaive, S. A. G. Taliun, W. Bi, M. E. Gabrielsen, M. J. Daly, B. M. Neale *et al.*, "Scalable generalized linear mixed model for region-based association tests in large biobanks and cohorts," *Nature genetics*, vol. 52, no. 6, pp. 634–639, 2020. 38

[56] L. Song, P. Langfelder, and S. Horvath, "Random generalized linear model:

a highly accurate and interpretable ensemble predictor," *BMC bioinformatics*, vol. 14, no. 1, pp. 1–22, 2013. 38

[57] X. Wang, V. M. Philip, G. Ananda, C. C. White, A. Malhotra, P. J. Michalski, K. R. M. Karuturi, S. R. Chintalapudi, C. Acklin, M. Sasner *et al.*, "A bayesian framework for generalized linear mixed modeling identifies new candidate loci for late-onset alzheimer's disease," *Genetics*, vol. 209, no. 1, pp. 51–64, 2018. 38

[58] D. Beaver, "Foundations of secure interactive computing," in *Annual International Cryptology Conference*. Springer, 1991, pp. 377–391. 39

[59] Y. Lindell and B. Pinkas, "Privacy preserving data mining," in *Annual International Cryptology Conference*. Springer, 2000, pp. 36–54. 39

[60] P. Oechslin, "Making a faster cryptanalytic time-memory trade-off," in *Annual International Cryptology Conference*. Springer, 2003, pp. 617–630. 42

[61] A. Byerly, T. Kalganova, and I. Dear, "No routing needed between capsules," *Neurocomputing*, vol. 463, pp. 545–553, 2021. 68

[62] J. Ding, G. Liang, J. Bi, and M. Pan, "Differentially private and communication efficient collaborative learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 8, pp. 7219–7227, May 2021. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/16887 68

[63] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016*

ACM SIGSAC conference on computer and communications security, 2016, pp. 308–318. 68

[64] O. Choudhury, A. Gkoulalas-Divanis, T. Salonidis, I. Sylla, Y. Park, G. Hsu, and A. Das, "Differential privacy-enabled federated learning for sensitive health data," *arXiv preprint arXiv:1910.02578*, 2019. 69, 71

[65] B. Letham, B. Karrer, G. Ottoni, E. Bakshy *et al.*, "Constrained bayesian optimization with noisy experiments," *Bayesian Analysis*, vol. 14, no. 2, pp. 495–519, 2019. 70

[66] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization." *Journal of machine learning research*, vol. 13, no. 2, 2012. 71