IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

MSc Computing Individual Project Report

Author:
Barry Quek Jing Jet

Supervisor:
Dr Michael Huth

Submitted in partial fulfillment of the requirements for the MSc degree in MSc Computing of Imperial College London

August 2022

Abstract

This project considers the application of protocols developed in Federated Analytics to a hypothetical use case: a 'mood app' that offers users a way to track their mental and emotional state as well as access to wellness resources. It contributes a consideration of key techniques for secure computation like differential privacy and homomorphic encryption, as well as some more recently introduced methods. It also contributes a consideration of how to specify the product desidereta for such a use case. Moreover a way to make a particular protocol, SAFE, more secure, is given.

Acknowledgments

I would like to thank Dr Michael Huth and Dr Jonathan Passerat-Palmbach for their invaluable discussion and help. I have learnt much from them.

Contents

Chapter 1

Introduction

In this introduction, I explain the motivation behind the project, as well as its objectives. At the end, I provide an overview of the report.

1.0.1 Motivation: The Privacy-Utility Tradeoff

Privacy is seen by many as a human right. Consider, for instance, how the GDPR describes one of its purposes as protecting ”fundamental rights and freedoms of natural persons and in particular their right to the protection of personal data” [22]. Yet, data collection is everywhere. Even if one tries their hardest to stay off the radar, this is simply not possible in most places.

A big reason for this is that data collecting technology is everywhere. Where the devices are; there the information collection will be. Most of us have little control over the data that is stored, analysed and sold about us.

Yet, not all of this is bad. Or, at least the fact that data is collected and put to some use is not necessarily bad. So much of the products and services we regularly enjoy are the result of data-informed processes. Additionally, we often want just information about a host of things to be more informed– where would that information come from if not via these technologies?

This all presents us with a tension to reckon with: balancing our desire and need for privacy with the substantial benefits brought about by information flow. In the market context, this can be framed as a trade-off between user privacy and product utility. If no information is collected, then how can products and services be improved to match user preferences and needs? But, if (or rather since) a lot of information is collected, how can we reasonably hope to be as private as we like?

1.0.2 A Concrete Case: Mood Apps

This project will consider this broader question with respect to a particular use case: some mobile app that is made available to some population of users. Is it possible to collect data about users whilst letting users remain private? It is the aim of the project to answer in the affirmative.

We begin with a general description of the kind of app that would be deployed. Suppose we developed a mobile app that prompted users to respond to questions about their mood on a regular basis. This could involve prompting them to select from a set of possible responses to ’survey-like’ questions (e.g. describing on a scale of 0-10 how one is feeling). Or it could offer prompts and a text editor to write text prose reflections. It may seem strange that users would consider responding to such questions, but it is not altogether unlikely. Maybe users want to have a convenient way to track their mood for greater self-awareness.

Additionally, suppose that this app not only probed users for input but also offered a selection of resources for improving wellness. For example, the app could provide meditation guides or articles advising on how to manage stress or navigate difficult situations.

Such apps exist. For instance, the app Stoic advertises itself on the App Store as a ”self-care journal mood tracker”. Stoic offers prompts for users to journal in response to (e.g. ”What has been wearing you out lately?”) as well as meditation guides.

For sake of convenience, let us refer to this kind of app as a ’**mood app**’. However users interact with the app, one thing is certain: their input would be highly personal. It is likely that users would want to use the app whilst minimizing or even eliminating the possibility of anyone knowing about how they are using it.

It is also, however, likely that those offering the app or some other parties would have interest in learning things about their users. Service providers generally rely on data about their users to personalize and improve their products. Moreover, we can imagine they might just want to learn facts about their user population to aid decision-making.

For example, say that a mood app was used by some particular community of moderate (hundreds or few thousands) size. Perhaps a religious community or a school. The user population for the app would be restricted to a specific community. Certain figures (e.g. leaders) could have a legitimate interest in learning about users. They may want to know how people in the community re generally feeling, or what resources are being used most. For simplicity, let us assume that the service providers of the mood app have these sort of analytical ambitions.

These interests from both the user and analyst’s perspectives creates some tension: how can service providers respect user privacy whilst learning from their inputs?

1.0.3 Objectives

The objective of this project is to consider how user privacy and analytical insight in this kind of use-case might be realised togetherr.

More specifically, the goal will be to discover methods that realise **input** and **output privacy** in the use case. Ideally, it should do this in a computationally efficient way, whilst also catering to a variety of scale settings. These concepts will be further explained in 3.0.2, but essentially, the aim is to allow for computations to be run on user data whilst letting that user data remain inaccessible to all others except the users themselves.
To this end, we will consider a variety of **secure computation method**s that might help us achieve this aim.

1.0.4 Overview of Report

Chapter 2 will provide technical background on the methods to be considered. Chapter 3 further explicates the concepts of input and output privacy within the framework of **structured transparency**, and assesses the applicability of the methods discussed in Chapter 2 for achieving them. Chapter 4 features the main contributions of the project: focus is placed on the SAFE Protocol due to some unique advantages it has over the other methods for the use case. A few experiments are run to (1) illustrate the protocol, (2) consider some privacy vulnerabilities, (3) explore how these might be addressed, (4) assess what trade-offs have to be made. Chapter 5 and 6 close the report with comments on future work that could be done.

Chapter 2

Technical Background

2.0.1 Introduction

The aim of this chapter is to provide background knowledge on the various tools and techniques that will be considered later in the report. I begin with an explanation of the broader field this project is concerned with: Federated Analytics. Then, various secure computation methods typically employed in this space are explained.

2.0.2 Federated Analytics

To better contextualize the development of Federated Analytics, we must consider a closely-related field: Federated Learning.

Federated Learning

Most generally, Federated Learning involves the collaborative training an Machine Learning model by many clients (e.g. mobile devices) that are orchestrated by some coordinator (e.g. a central server) whilst training data is kept decentralized [32, p4]. Client data is stored on client devices, and clients should not have access to the data of other clients. The coordinator merely coordinates the training of a central model and should never have access to raw client data.

Federated Learning can occur in many forms. Broadly speaking, the two main settings in which it can occur are cross-device or cross-silo learning [32, p6]. The main difference between the two consists in who the ’clients’ or ’users’ contributing training data are. In cross-device learning, the users are typically edge devices like smartphones used by individuals. In cross-silo learning, the users are independent institutions, each contributing their own datasets which they wish to keep private.

The Link Between Federated Learning and Federated Analytics

One issue that arises in this kind of learning setting is how models can be evaluated: how do we assess the performance of the central model without access to the raw data it is indirectly trained on? Conventional performance metrics like Precision or

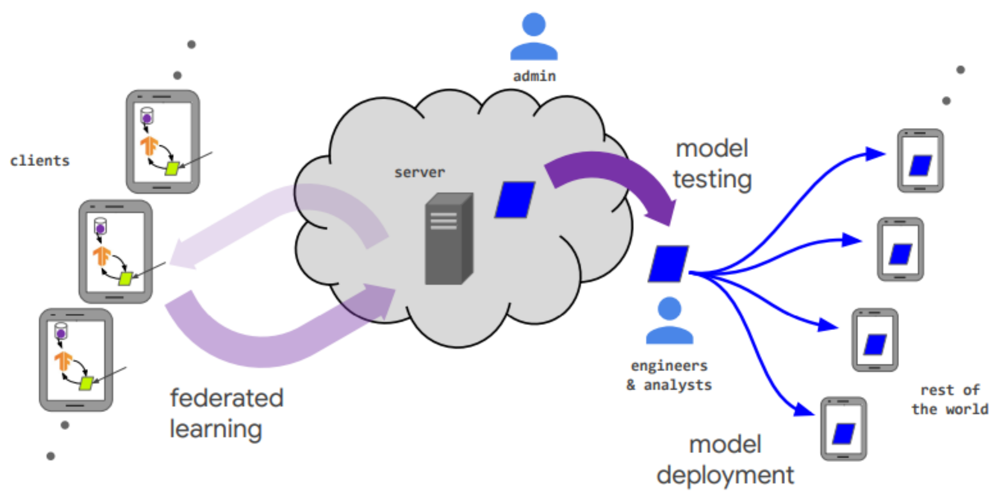


Figure 2.1: Lifecycle of a model trained via FL and the various actors. [32, Figure 1]

Accuracy cannot be used since these rely on access to the raw data (e.g. to know the number of true positives in a classification problem).

Federated Analytics was introduced to address this problem. [2] It ”uses the same infrastructure as federated learning but without the learning part” [31, p2]. The aim is not to train a model but to enable metric computation via functions that compute over decentralized user input. As with Federated Learning, user data remains private and inaccessible to the analyst, and users should not be able to access the data of other users.

For instance, engineers at Google used federated analyitcs to assess the quality of next-word recognition prediction models deployed onto users' phones [2]. Code that computes the evaluation metric was included in the definition of the model deployed on phones, and this code could be used to compute the metric for assessment.

2.0.3 Secure Computation Methods

However, Federated Analytics has since evolved to capabilities beyond mere model evaluation [2]. While still a young field, it has developed to achieve other aims like computing descriptive statistics. For instance, in [2] it is described how Google engineers have used Federated Analytics to detect which songs are most often recognised by users' phones without disclosing which songs are recognised by any specific phone.

These developments are driven by the use of various privacy-enhancing technologies or **secure computation methods**. Most generally, secure computation methods seek to allow for evaluations on distributed inputs without revealing any additional information other than the results to intended parties [32, p40]. For example, Google's song recognition protocol utilises a secure aggregation protocol [21] to allow for the central server to compute the total number of songs recognised across engaged devices, *without* the ability for it to know how many songs each device recognised (nor what those songs were).

In the rest of this background chapter we will consider some secure computation methods to better understand the current state of Federated Analytics. This burgeoning field is too massive to comprehensively cover in a single chapter, so instead I will focus on describing two broad classes of secure computation methods: Secure Multi-Party Computation (MPC) techniques, and Differentially-Private techniques. The coverage of these fields will be relatively cursory and general. In the next chapter, we will engage in a more fine-grained analyses of some techniques and then consider their application to the aforementioned use case.

Secure Multi-Party Computation (MPC)

Description

As noted in [32, p40], Secure MPC is best regarded as a field of secure computation techniques, rather than a technique per se. Most generally, secure MPC techniques allow for multiple parties to compute the output of some agreed-upon function of their private inputs without the disclosure of those private inputs [32, p40].
One particular kind of technique secure-MPC technique that has emerged relatively recently are homomorphic encryption schemes.

Homomorphic Encryption

Homomorphic Encryption (HE) is a kind of encryption scheme. One way to understand it's significance and design is to contrast it with classical asymmetric public key encryption.

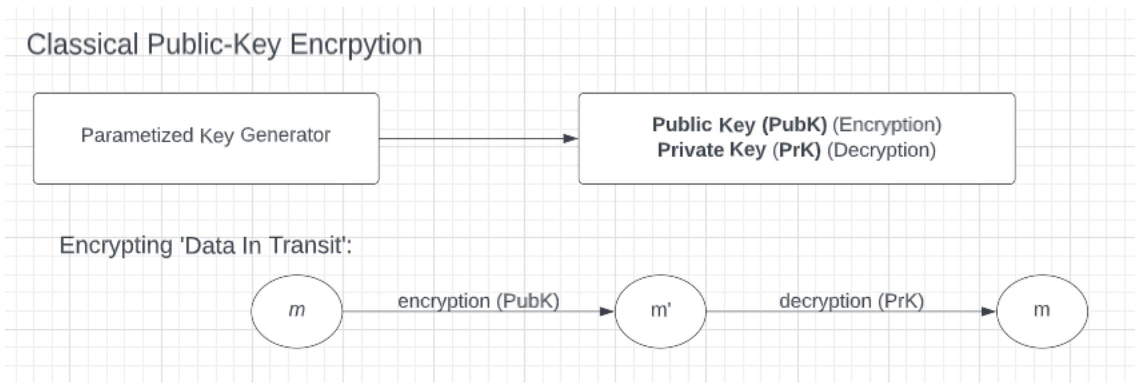

Classical Public-Key Encrpytion
Parametized Key Generator
Public Key (PubK) (Encryption)
Private Key (PrK) (Decryption)
Encrypting 'Data In Transit':
m
encryption (PubK)
m'
decryption (PrK)
m

Figure 2.2: General Flow of Classical Public Key Encryption Schemes.

One purpose of asymmetric encryption schemes is to facilitate the transfer of information between parties in a way that only those they intend to understand it can.

In classical public key encryption, some parametized key generator is used to generate a public and private key pair for each communicating party. Each individual is to keep their private key secret, whilst their public keys are typically disclosed to whomever they wish to communicate with. If one party wishes to send a message m to another, they would use the receiving party's public key to encrypt their m, resulting in an ecrypted message m' that sent to the receiving party. Using their private key, they can decrypt m' to get m.

This kind of encryption is sometimes referred to as encryption for 'data in transit' or 'data at rest'. By contrast, Homomorphic Encryption seeks to achieve encryption for 'data in use' [20]– allowing for computations to be performed on encrypted data *without* decrypting the data.

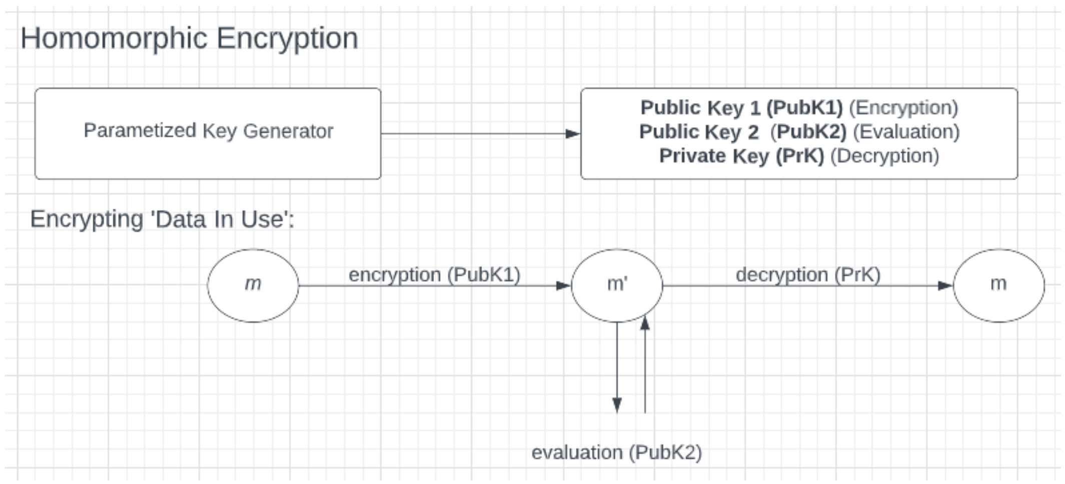

Homomorphic Encryption
Parametized Key Generator
Public Key 1 (PubK1) (Encryption)
Public Key 2 (PubK2) (Evaluation)
Private Key (PrK) (Decryption)
Encrypting 'Data In Use':
m
encryption (PubK1)
m'
decryption (PrK)
m
evaluation (PubK2)

Figure 2.3: General Flow of Asymmetric Homomorphic Encryption Schemes.

As in classical public key encryption, some parametized key generator is used to generate public and private keys. However, in this case two separate public keys are generated– one meant for message encryption and the other for message evaluation (i.e., executing computations on the encrypted data) [20]. These two public keys are still paired to the private key, which parties keep secret. Encryption of m and decryption of m' happen as in classical public key encryption, however processing may be done on m' via use of the public encryption key of the receiving user.

In classical public key encryption, one cannot perform computations on the encrypted data without losing the ability to decrypt the original data [20]. Moreover, such computations would not be informative– they would not yield results that pertain to the original contents of the message m. In Homomorphic Encryption schemes, one can apply operations to m' such that, once m^prime is decrypted by the receiving party, one ends up with the result of those computations *as though applied to the raw data*.

Different Homomorphic Encryption Schemes

The general discussion of this section relies on [18].

The kind of operations that are permitted on encrypted data is one factor in which the various Homomorphic Encryption (HE) schemes differ from one another . These operations are *evaluation circuit operations* [20]. Evaluation circuits are directed acyclic graphs of inputs, operations, and outputs. These inputs and outputs can take various forms (e.g. reals, integers), but are usually decomposed into bit form at the lowest level. The scheme specifies the particular evaluation circuit at work.

Some HE schemes are **partial**– they allow for only one kind of operation to be applied to ciphertexts. One example is the Paillier (Pallier) Scheme which allows for just addition to be done between two encrypted messages decomposed into separate input bits [4].

Other HE schemes are **leveled** HE schemes– they allow for both multiplication and addition operations on ciphertexts but only up to a certain circuit depth i.e., a certain number of times. This is because the ciphertexts accumulate noise as a feature of their security [19], and there is a limit to the number of operations that can be applied before the encrypted message can no longer be decrypted to the actual result. One example is the CKKS scheme (CKKS) that allows for addition, multiplication, and subtraction of real values [3].

Some HE schemes employ *bootstrapping*. This is a technique for eliminating noise introduced by previous operations. Though it introduces noise of its own, it paves the way for **fully** HE schemes that have unlimited evaluation circuit depth– so long as one more operation can be applied before bootstrapping is needed again, we effectively have the ability to apply as many operations as we like without the ciphertext becoming too noisy [19].

Assessment of these kinds of schemes for our use case is deferred to 3.0.3. Now we shall turn to consider a secure computation method of a rather different flavour, but no less vogue: Differential Privacy.

Differential Privacy

Besides the host of computation techniques that involve some sort of encryption, there is what one paper has termed ”state-of-the-art model for quantifying and limiting information disclosure about individuals” [32, p44]: Differential Privacy. As with computation schemes that involve encryption, there are many different flavours of differential privacy.

Most generally, differential privacy is a formal privacy guarantee for individuals when releasing statistical information about a population they are a part of [33, p3]. The aim is to disentangle information about a population from information about specific individuals in that population. As a result, anyone viewing the released results can learn new things about the population. However, they would not be able to learn any information about the individuals in the population that would not be implied by what they learn about the population [29, p4].

We can illustrate this with the use case we introduced in 1.0.2. Say that the mood app is being used by some specific community, and the analysts want to conduct a study to determine the proportion of the community that has depression. A differentially private study would provide a certain guarantee to participants that their participation in it would not be inferrable from the released results of the study.

The significance of this is most easily seen in light of researchers' demonstration of the insufficiency of mere anonymization. Researchers managed to de-anonymized a significant portion of a database that Netflix had released some time ago through a **linkage attack** [5]. differential privacy has since been an active area of research and implementation– even reaching global-scales in terms of industry roll-out [29, p2].

Formal Definition

The formal guarantee of differential privacy can be understood in terms of its formal definition. To understand this definition, we must introduce some terminology.

Let D and D' be some **adjacent dataset** pair that can be drawn from some **universe** X of possible datasets. This adjacency can be construed in different ways: in a **bounded** version of differential privacy, D and D' are different in terms of their contents though they have the same cardinality. In an **unbounded** version of differential privacy, the datasets differ in their cardinality– the adjacent dataset D' is generated via addition or removal of some record. [35]

Following [30, p3], we will assume that at least for our use case, if metrics were released, the size of the dataset used would also be known. (This seems to be a reasonable possibility to account for, rather than assuming that the database size would be hidden.) So, we will construe adjacency in bounded terms: D and D' are adjacent datasets that differ in terms of their contents. Let D contain the record of an individual x in the population that contributes to the dataset universe X, and D' contains some dummy value in place of their record instead.

Let M be a mechanism that returns some output that is in the space of its possible outputs S given a dataset from the universe X. ε-differential-privacy is a property of M, such that:

$$\Pr[M(D) \in S] \leq e^{\varepsilon} \cdot \Pr\left[M\left(D'\right) \in S\right]$$

where $\varepsilon > 0$. An intuitive reading of this definition is as follows: the guarantee of differential privacy is that the probability of any output M might generate is within a multiplicative factor e^{ε} regardless of whether x's record is included in the input dataset or not [30, p3].

Key to the privacy guarantee then is the paramater ε. If ε is close to 0, then e^{ε} is close to 1, meaning that the probabilities will be very similar. The larger that ε is, the more that the probabilities can differ [17].

Seen from the perspective of a privacy guarantee, that means a smaller ε guarantees more privacy, and a larger ε means less privacy. We will consider a number of important issues related to this parameter in 3.0.3.

$\varepsilon - \delta$ Differential Privacy

Another important generalisation of the DP mechanism is $\varepsilon - \delta$ differential privacy. It's formal definition adds another parameter δ:

$$\Pr[M(D) \in S] \leq e^{\varepsilon} \cdot \Pr\left[M\left(D'\right) \in S\right] + \delta$$

Intuitively, this definition gives more 'leeway' for a DP-mechanism by allowing a δ "probability of failure where the mechanism could violate privacy". [30, p21]. In this paper, unless stated otherwise, we will mainly focus on ε differential privacy.

Adding Noise

ε-DP mechanisms achieve their privacy guarantee through the addition of some random noise to the original output, such that a particular output of M depends not only on the input data but also randomness. There are various kinds of noise that can be added, but the the standard approach adds noise taken from a Laplace distribution parametized by $\frac{s}{\varepsilon}$ where s is the sensitivity of M [16]. For the mechanisms we will consider, the sensitivity of a mechanism is the amount its output changes when its input changes by 1.

General Properties of DP Mechanisms

To better understand the nature of this privacy guarantee, it would be helpful to also consider of some salient properties of DP mechanisms. The following points are taken from [30, p4].

Firstly, differential privacy makes no assumptions about the amount of knowledge an adversary has, and this allows its guarantee to hold even in worst-case scenarios. This is significant because it can not only be difficult to reason about the knowledge an adversary might have, but reasoning wrongly about this is often the cause of privacy breaches [30, p1].

Secondly, the guarantee of a differentially-private mechanism will still hold under arbitrary post-processing. If one applies some function f to the output of a DP-mechanism M, the result is still differentially private. This makes DP a flexible framework.

Thirdly, DP mechanisms are compositional. For example, running k ε DP mechanisms (either in succession or in parallel) results in a $k\varepsilon$- DP mecahnism. This privacy degredation over multiple runs makes it easy to track the way privacy guarantees evolve as more complex algorithms are built from simpler primitives.

Privacy Budget

This last point about the compositional nature of DP brings us to an important consideration for implementation of DP-mechanisms. The cumulative privacy loss that occurs over multiple runs of a DP-mechanism calls for the clear definition of a **privacy budget**– ”the maximum acceptable privacy loss allowed before no more queries are permitted” [29, p6]. Once exhausted, the dataset *should* be retired, and not queried again to maintain the specified privacy guarantee [29, p6].

Different Models of Differential Privacy

We come to the last feature of DP models which we must consider, which are the different settings in which they can be applied.

Actual implementations of differential privacy can differ in terms of *who* adds the noise. In a **central** model of differential privacy, some trusted third-party applies the noise to the computed result [32, p44]. In the case of federated analytics, if the results are to be disclosed to just the analysts, this would involve some other orchestrating third-party. Alternatively, it could be that the aggregators/analysts themselves apply the perturbations, and release a public noisy result. In either case, there is some party other than the users themselves that have access to the raw data. One example of this kind of implementation is LinkedIn’s PriPearl protocol [33]), which adds pseudorandom laplace noise to actual query results before releasing it for members to view.

A **local** model of differential privacy seeks to avoid the need for a trusted party by instead having each user apply perturbations to their own data before sending noisy inputs to the server. In this case, the adjacent datasets D and D' in the formal definition are part of the universe X of possible local datasets. One example of this is Microsoft’s protocol for collecting telemetry data on user devices [25].

2.0.4 Summary

We have seen the link between Federated Analytics and Federated Learning, and also surveyed briefly two broad classes of secure computation methods. In the next chapter, we assess and explore these in greater detail in relation to our use case.

Chapter 3

Initial Assessment of Secure Computation Methods

3.0.1 Introduction

The aim of this chapter is to relate the various secure computation methods discussed in the previous chapter to our use case. A more detailed explanation of what is involved in **input** and **output** privacy is first given. Following this discussion of the 'product desiderata', we will consider in greater detail some of the methods we discussed in the previous chapter. Motivation for focusing on the SAFE Protocol in particular will be offered at the end.

3.0.2 Product Desiderata In Greater Detail

We've mentioned that for our use case, we want both for there to both be user privacy and the potential for analytical insight. It would be good to be more specific about what this actually involves. In other words, we want to consider the product desiderata in greater detail. We will consider this from both the user and service provider perspective.

User Perspective

We begin with the user's perspective. To do this, we will make use of a framework introduced in a paper done by some researchers at OpenMined [24]. In this subsection, I explicate this framework.

Background To Framework

First, we must understand what an information flow refers to. The authors of this paper offer a taxonomy of information flows in order of increasing complexity [24, p3-4]:

1. **Messaging Flows**: Information flows that consists simply in the transfer of some data from one party to another. (e.g., the exchange of text messages between two parties)

2. **Service Provider Flows**: information flows that the reception of some result that requires computation over information sent. (e.g., personalized services like automatic product recommendations)

3. **Aggregation Flows**: like service provider flows with one addition: the provider seeks not only to compute some result and return it to the client, but also learn some fact (e.g., the aggregate mean of the data a business receives on it's product user's habits.)

We are concerned with the third kind of information flow in this project.

Now, we can explain the framework which we can use to describe our ideal product's privacy guarantees: **structured transparency**. Structured Transparency is a feature of a system in which there is some information flow. Systems possess structured transparency to the degree that they enforce some ”desired information flow” [24, p3]. This ’desired information flow’ is specified in terms of two components: Firstly, the specific constraints on parties involved. Secondly, the specific ways these constraints will be upheld.

This first component (the specific constraints) is informed by the notion of Privacy as Contextual Integrity. The second component (the means to enforce those constraints) is determined by the specific secure computation a particular use-case employs [24, p3].

We will deal with the secure computation methods for our use-case in later sections. Here, we want to specify the constraints. For this, the authors of the paper on Structured Transparency draw on the work of the legal philosopher Helen Nissenbaum [34]. In particular, her notion of Privacy as Contextual Integrity underlies their notion of what a ’desirable information flow’ consists of. They write, ”In the framework of contextual integrity, an ideal information flow is one that would enable us to collaborate over information while ensuring that information is used only for the *context-relative* ‘approved’ purposes.” (emphasis mine).

What are the ’context-relative’ approved purposes for our use case? In general, app providers are required to be explicit about the way that they store and handle user data. The context-relative approved purposes are at minimal specified by the specific privacy guarantees they specify in their privacy policies; information flows that run contrary to those guarantees would constitute as improper information flows.

For our case, let us add another assumption: that those providing the app wish to do so whilst maximizing user privacy. So, they are keen to not only specify nice-sounding guarantees but committed to achieving them in practice. Given the framework of contextual integrity, these service providers need to ensure that the information flows that result from the use of their app are in line with these commitments. In short, the framework dictates that service providers are constrained to achieve their analytical goals in a privacy-preserving way.

Components of Structured Transparency

What would this look like in practice? A helpful aspect of Structured Transparency is the way that it specifies various metrics to consider for information flows that occur in the system. Moreover, these metrics are two-sided coins: they can be considered from both the user and analyst perspectives: [24, p4-5]

- **Input Privacy:**
  - User Perspective: able to allow processing on their information without exposing that information
  - Aggregator Perspective: able to run computations on information hidden to them
- **Output Privacy:**
  - User Perspective: able to contribute to an information flow without the fear that someone can reverse engineer from the results your original inputs
  - Aggregator Perspective: able to read results of computation without being able to reverse engineer from the results to the original inputs.
- **Input Verification:**
  - User Perspective: able to verify that the output provided by the service providers used your inputs
  - Aggregator Perspective: able to verify that received inputs are really from purported senders
- **Output Verification:**
  - This component is mostly relevant from the user perspective. In essence, a system with output verification "allows you to verify attributes of any information processing (computation) within an information flow." [24, p4]. The most salient kind of output verification that parties using the app would be interested in is verifying that the privacy-preserving methods were used to generate the results that analysts are interested in.
- **Flow Governance:**
  - This component pertains to the kinds of guarantees that are in place over whether an "information flow will preserve...intended use" [24, p5] of the information. In other words, what assurance can users have that the stated privacy policy of service providers will be upheld?

Given that this project focuses on federated analytics, some metrics will be less relevant than others. Moreover, some of these simply do not apply in our use case. For instance, for input and output verification, users will not be provided the outputs of the analysts' computations, so there is no output for them to verify. Moreover, issues of verification are beyond project scope. While, in an ideal world, all five metrics would be realised for any given app, only input and output privacy will concern us here.

To summarize, from the user perspective, the product should be one that allows for input contribution in a way that analysts cannot see their raw data nor reverse engineer from the output to their raw data.

Analyst Perspective

Now we turn to a discussion of what analysts desire of the product from an analytical perspective.

Often, protocols in Federated Analytics aim to compute metrics that are computed in standard Data Analytics– just in a way that preserves user-privacy. Moreover, it is not clear that for this particular use case, potential analysts would have interest in computing any non-standard metrics.

So, we can assume that in an ideal world, analysts and service providers would be able to compute results and perform analytic work of the sort described in e.g. Google's own description of good data analysis practices. Some of these practices would not be applicable because they would involve handling raw data (e.g. data cleaning; slicing datasets). However, many of these (e.g. computing summary metrics like the mean, generating histograms etc.) would. Broadly speaking, the product desiderata from the service provider/analyst's perspective is just robust data analytics over user data.

The key would be to compute these various metrics and distributions in a way that is close to if not exactly identical to what would be computed in the non-federated case. We do not want to be constrained by a privacy-utility trade-off, but instead get to have our cake and eat it.

Other General Product Desidereta

So far we have focused on product desidereta with a focus on privacy and data utility. However, there are presumably other things that users and service providers would want that do not have to do with privacy or analytics directly.

These are are still worth mentioning for the reason that they serve as further factors to consider when we assess what techniques we can use to achieve both robust input and output privacy.

Some example desideretas include things like minimal software complexity, efficient resource management, and a smooth user experience. Moreover, we want to factor in that the mood app envisioned could be deployed in a variety of scale settings (e.g. small versus large user populations, 'small' versus 'large' service providers...) so we would want techniques that serve those specific settings well.

Putting Everything Together: Describing User Flow

Let us draw the various things we have discussed together and describe a possible user flow for a mood app.

When some user downloads the app, they are prompted to give their consent to participation in rounds of Federated Analytics. This consent is necessary for use of the app, and depending on whether the target audience is the general market (and thus service providers are seeking a profit) or some specific community (in which the providers merely have some analytical aims), the app can take various pricing models (or none at all).

Some key app features would be regular prompts to respond to 'survey-like' questions and a space for users to reflect about their experiences or thought-processes in prose. Other features would be the promotion of various wellness-resources (e.g. articles, meditation guides) which users can choose to access if they wish.

As users use the app, data is stored locally on their device and federated analytic protocols are run to compute metrics that are of interest to the service providers.

The ideal result would be: (1) an app users can liberally use without fear of privacy loss and which generates (2) app usage that analysts can learn richly from. These two main desidereta will ideally be achieved along with the other desirables mentioned in the previous section.

3.0.3 Assessment of Secure Computation Methods

Having canvassed the ideal product scenario, let us now consider the means by which we may realise it: secure computation methods. Beginning with Homomorphic Encryption, comments will be made on each method regarding its possible application in the use case. The considerations will not be exhaustive in considering every variant or flavour of each secure computation method, nor the most cutting-edge instances. Rather, consideration will be made at a higher-level for whether further experimentation and probing is warranted.

Homomorphic Encryption and Secure MPC

We begin with Homomorphic Encryption (HE) and secure MPC schemes, which were described in 2.0.3. In considering the use of HE schemes, or more generally secure MPC schemes, one must consider issues surrounding **key management**.

Consider the diagram which was shown in 2.0.3:

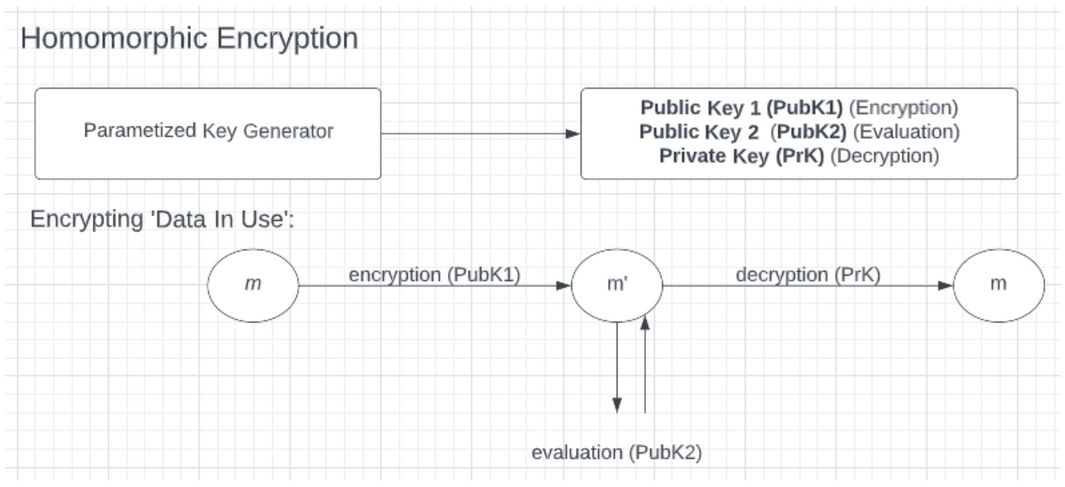

Homomorphic Encryption
Parametized Key Generator
Public Key 1 (PubK1) (Encryption)
Public Key 2 (PubK2) (Evaluation)
Private Key (PrK) (Decryption)
Encrypting 'Data In Use':
m
encryption (PubK1)
m'
decryption (PrK)
m
evaluation (PubK2)

Figure 3.1: General Flow of Asymmetric Homomorphic Encryption Schemes.

While illustrative of the basic idea, this diagram abstracts away a few things we must consider in deciding whether or not to apply HE in our use case. Firstly, whose keys are these? Presumably, these belong to the user whose data we wish to remain private, because the m that is to be encrypted is their data. So, each user will have an ecryption (public), decryption (private) and evaluation (public) key triplet. However, therein lies the problem: the analysts are presumably the ones who wish to see the result of the 'data in use', and they will not be able to do this unless they hold the decryption key. However, if they hold the decryption key, then they have the ability to access the original inputs, which would eliminate output privacy.

This problem hints at a more general issue to be addressed with using HE in federated settings, which is determining who can hold the secret key [32, p42]. However, there have been some workarounds developed. In particular, a distribution (or threshold) encryption scheme could be used to distribute the secret key among parties instead of assigned to a particular one to trust [32, p42]. A way to facilitate a threshold encryption scheme is described in [27, p362], where a given ciphertext (or m') is decryptable only by any $t + 1$ out of n intended participants, where $t + 1$ is in effect a 'threshold' for the number of parties that would be required to decrypt the ciphertext. Any less than $t+1$ parties would not be able to decrypt the encrypted message, which means that, for instance, the central server or analyst would not be able to decrypt the message by themselves.

This can be applied in for what [27] describes as LOVE MPC– "Large-Scale One-Server Vanishing-participants Efficient MPC" [27, p364] in honest-but-curious settings. Each user can send ecrypted inputs to the server, who homomorphically combines them before disseminating the result to be decrypted by all users. So long as more than t users respond with partial decryptions, the server will be able to construct the correct result. This particular threshold encryption scheme has the advantage of not requiring correlated set up among recipients; each user simply communicates with the server. All it relies on is the set up of a basic public key infrastructure [27, p362].

Thus, there could be some settings in which a mood app of the kind we have described makes use of a HE scheme to achieve some aggregate results in a way that preserves input and output privacy, namely those with a public key infrastructure in place. However, reliance on public key infrastructure itself brings its own set of issues.

Public Key Infrastructure

A public key infrastructure is some way of managing the distribution and authentication of public keys. In order for public keys to have utility in secure communications, parties must be able to *authenticate* that public keys do map to the *principal*– the ”entity that these identities represent” [6, p3].

One way this is actually done today is through the use of public key **certificates** that are signed by Certification Authorities (CAs). For instance, X.509 certificates are used in the protocols TLS/SSL, which is the basis for HTTP [15].

One limitation with delegating the management of public keys to CAs is that they are ”corruptible central points of failure” (6, p2). Compromised CAs (for instance, by hackers) can generate fradulent certificates which allow for ’man-in-the-middle’ attacks, where adversaries intercept sensitive messages in a communication through impersonating parties to that communication [6, p2].

Overall Evaluation

Despite these concerns, HE or MPC schemes could be of use to achieving input and output privacy for mood-app use cases. Moreover, these issues are not particular to HE or MPC schemes.

However, other techniques may be preferable especially if the complexities and drawbacks of centralized key management are deemed to be unfavorable costs. We have reason then, to look elsewhere whilst acknowledging possible utility here. In light of that, let us now consider the other secure computation method we introduced in the previous chapter: differential privacy.

Differential Privacy

By itself, differential privacy (DP) seems to offer a suitable alternative to the encryption schemes considered before. Here, there is no reliance on an established public key infrastructure. Moreover, a formal guarantee on the privacy afforded to individuals by mechanisms can be given.

However, it should noted at the outset that a typical central model of DP is not going to be applicable for our use case. In the central model, the party who adds noise to the computed result has access to the raw data, and this would be a violation of input privacy. So, we will concern ourselves mainly with the local model.

There are two issues that we must consider in evaluating the applicability of a DP mechanism to our use case: the required size of the database and how to set the ε parameter.

Concerning Scale

One issue to reckon with especially in the context of a local model of DP is scale. As noted in [32, p54], in the local model, the privacy guarantee of a user depends on their own contribution of randomness to their input data, and is independent of the combined randomness of all users. Consequently, the magnitude of overall noise needs to be matched by a similar magnitude of signal from the data, and this could require combining of reports [32, p54]. LDP models therefore require either larger ε values (to account for lower noise) or larger datasets [32, p54].

Importantly, this is not so much a limitation of DP as it is a feature of it– it's what we should expect given what it aims to do. As noted in [29]

> ”Speaking informally, we expect statistical estimators run on large datasets to be more robust to the addition or deletion of a data point, as compared to the case with small datasets. In other words, adding or removing an individual from a small dataset is more likely to significantly change the value of the statistical estimator as compared to when the dataset is large. Despite lower accuracy, differential privacy is indeed working as intended...” [29, p5]

Thus when deciding whether or not to use a differentially private algorithm, one must consider the target scale. As mentioned in section 3.0.2, we would consider techniques that would work for both small and large scale settings. For a sense of what ’large’ could feasibly mean, consider theexample of the aforementioned app **Stoic**, which advertises itself (at time of writing) as having ”3 million happy users” on its website. (Incidentally, this is the size of the dataset which researchers at Microsoft had run experiments on for their local DP protocol of collecting telemetry data [25, p7], with success). Thus, while local DP may not be very suitable for smaller scale settings, it could well facilitate analytical insight whilst guaranteeing user privacy.

Choosing ε

Is there a principled way to determine how ’large’ a dataset needs to be for a DP-mechanism to realise both privacy and accuracy aims?

This question brings us to an issue that has been increasingly discussed in the space of DP: how to choose an optimal ε value.

While the ε does not tell all about the privacy guarantees afforded by a mechanism [29, p8], it does, ceteris paribus, function as a key determiner for it.

Despite many recent instances of industry roll-outs of DP protocols, there is evidence that there is no clearly defined way for setting ε among practitioners or researchers. As noted in [30] :

> ”This is evident in the literature [32, 7, 30, 24, 5, 31, 6, 28, 38, 8, 2, 41, 43, 27, 9, 12, 40, etc.], where algorithms have been evaluated with ranging from as little as 0.01 to as much as 7, often with no explanation or justification. A similar concern applies to a second parameter in (,)-differential privacy, a standard generalization of differential privacy” [30, p2]

Similarly, commenting on the findings of various interviews they conducted with firms who had rolled out implementations of DP [29],

> ”We found no clear consensus on how to choose or even how to approach this and other key implementation decisions; choices vary widely, resulting in systems that afford wildly different privacy guarantees; and, there is little collaboration, information sharing, or publishing to advance critical reflection on these key implementation issues.” [29, p3]

Fortunately, this is an issue [30] sought to address by providing a model for first construing the ε parameter and then setting it given accuracy and budget constraints.

A sustained consideration and experimentation with differentially private protocols is not what this project will engage in for some reasons discussed in the next section.

However, it is of some research interest to consider how their model might inform an implementation of differential privacy in our use case. So, I have explained their model and considered how it might bear on the use case in a notebook that can be found in the software archive. Bearing some important caveats in mind, their model yields some interesting results:

- some specific value bounds for how ’large’ a dataset would be needed for a differentially private study in our use case
- sense of how large a budget might be needed for an app which analysts would want to run multiple queries on
- the possiblity that a differentially private version of the mood app could be cheaper than a non-DP version.

The reader is referred to the notebook for the analysis.

3.0.4 A Different Way Forward: SAFE

Even with a suitable DP mechanism to apply for certain use case settings, it would be desirable to find a protocol that bypasses the aforementioned issues and limitations whilst still helping us achieve our aims. Ideally, a protocol should be relatively lightweight (using little resources to run and set up) and also applicable to both smaller and larger settings.

For this, we can consider a relatively new approach in Federated Analytics: Bayesian Federated Analytics. This comes from [31], in which Huth and Chaulwur propose the SAFE protocol. SAFE is a protocol for the secure aggregation of feature vectors for $N > 1$ users in a **semi-honest security model** [31, p3]. A semi-honest security model is one in which adversaries (i.e., actors who wish to compromise the security of a protocol) conform to the protocol execution but try to find out about the private inputs of others [14, p2]. A Jupyter notebook illustrating a run of the SAFE Protocol can be found in the software archive. The main result to highlight is that running the protocol allows an aggregator to compute the aggregate sum (and thus aggregate mean) of the raw feature values of the N feature vectors without having access to the raw feature vectors of the N users. For instance, in the notebook, a SAFE protocol run is conducted with 3 users who each have the following two-dimensional feature vectors:

1. User One's Feature Vector: $\{\{0.493\}, \{0.7682\}\}$
2. User Two's Feature Vector: $\{\{0.0885\}, \{0.1320\}\}$
3. User Three's Feature Vector: $\{\{0.3074\}, \{0.6341\}\}$

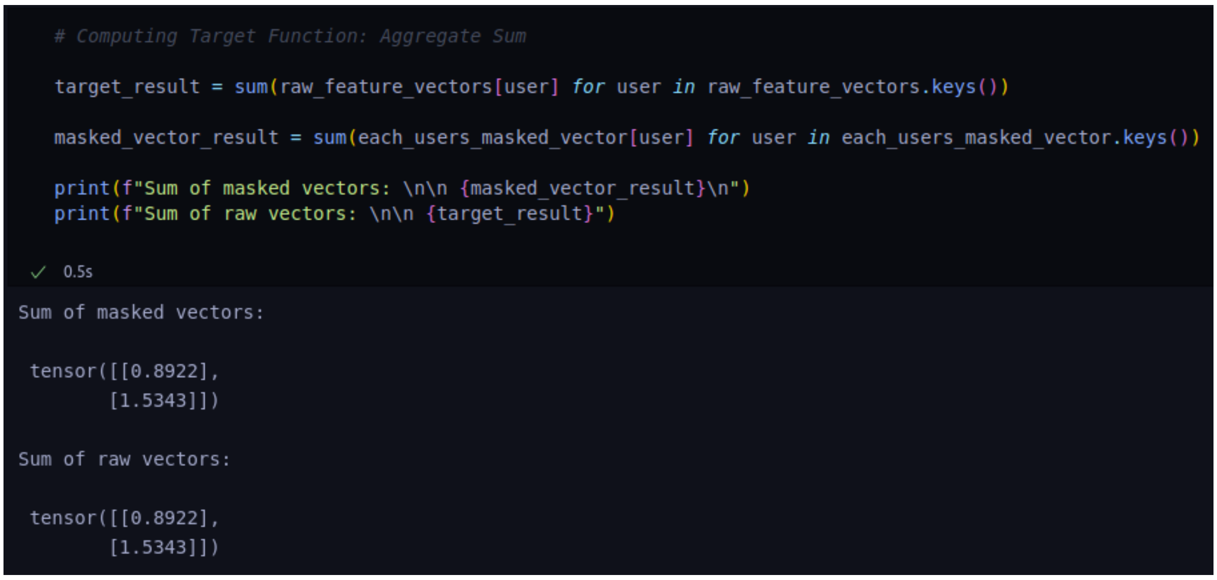
These feature vector values (between 0 and 1 inclusive) represent distributions of raw data.

The aggregate sum of these raw feature values is $\{\{0.8922\}, \{1.5343\}\}$. As indicated in the notebook, this is what an aggregator can compute over obfuscated feature vectors that he receives from each user.

3.0.5 Further Experimentation

The SAFE protocol has a few nice advantages for application in our use case. Unlike a local model of differential privacy, it can provide secure computations over user input even with a small dataset. Moreover, unlike MPC or HE schemes there is no complex set up of a public key infrastructure to achieve some level of secure computation (although, as we shall see in 4.0.3, this may be required for certain settings in which SAFE is run).

Thus, it is worth devoting some energy to considering how the protocol might be applied in our use case in greater detail. This is what we will do in the next chapter.



```
# Computing Target Function: Aggregate Sum

target_result = sum(raw_feature_vectors[user] for user in raw_feature_vectors.keys())

masked_vector_result = sum(each_users_masked_vector[user] for user in each_users_masked_vector.keys())

print(f"Sum of masked vectors: \n\n {masked_vector_result}\n")
print(f"Sum of raw vectors: \n\n {target_result}")
```

✓ 0.5s

```
Sum of masked vectors:

 tensor([[0.8922],
        [1.5343]])

Sum of raw vectors:

 tensor([[0.8922],
        [1.5343]])
```

Figure 3.2: Secure Aggregation of Raw Feature Vectors.

3.0.6 Summary

In this chapter, we considered in greater detail what we would like to achieve for our mood app use case. We identified input and output privacy as the main desidereta, whilst acknowledging that these should ideally be achieved along with other features that do not strictly have to do with privacy (e.g. ease of computation, adapatable to various scales). On these counts, we have ended up with an interest in a particular federated analytics protocol amongst the ones we have surveyed here: the SAFE protocol and its use in a Bayesisan federated Analytics. In the next chapter, we will consider how it may be used in our use case in greater detail.

Chapter 4

Experimental Results

4.0.1 Introduction

In this chapter, I document experiments done with the SAFE protocol. First, I apply it to the use case. Then, I consider why and how we might want to make it more secure. Finally, I provide some evaluative comments on the extended protocol and revisit project objectives.

4.0.2 Applying SAFE

Let us begin with describing how SAFE might be applied to the use case. In their paper, Huth and Chaulwar apply the SAFE protocol to perform trend detection analysis [31, p4-6]. It is specifically this metric which we will experiment with.

Definition of a Trend

First, we consider what a 'trend' is. Huth and Chaulwar offer the following definition of a trend for interacted documents in their paper,

> "Keywords that are not frequently present in past interacted documents but that appear frequently in current interacted documents" [31, p4]

They note that while their definition is basic, it suffices to illustrate how their Bayesian approach to federated analytics [31, p4].

Document Interactions

To more clearly specify what is being computed in the approach, we must consider a few key terms and variables:

- D - Document Set
- V - Set of possible trending keywords in D
- t - Trending Keyword
- D_i - subset of D that $user_i$ has interacted with

The aim is to determine a distribution $p(t|D)$ that specifies a probability for a keyword in V being trending.

Using Bayes formula, the task can be phrased as the determination for each term t_i in V:

$$p(t_i = t|D) = \frac{P(D|t_i = t)p(t_i = t)}{p(D)}$$

Here we are interested in calculating the term $p(t_i = t)$, the posterior probablity for a keyword in V for being a trending keyword.

Mood Trends In Use Case

There are two forms of input described in section 3.0.2 that we will want to detect trends on:

1. Pre-defined responses users select when faced with prompt questions
2. Various wellness resources users access whilst using the app

Let us begin with the first.
Responses To Prompt Questions

Let us first clarify what are some possible prompt questions and pre-defined responses users will see.

For a 'mood app', the simplest and most basic question that users could be asked in track their mood is something like,

How are you feeling today?

We could allow for users to respond to this in two ways: either by selecting from a pre-defined list of responses, or by allowing them to explain in their own words how they are feeling.

Certainly, the latter approach would capture a richer and perhaps more accurate picture of how a user was feeling. However, since our purpose is primarily to illustrate the application of SAFE to a use-case, we will opt for the former option, which is simpler. Allowing for users to compose their own responses to this question would require for us to devise a way to handle Out-Of-Vocabulary [13] words in the protocol. This is something which can foreseeably be done (since there are methods for doing so), however, would take us beyond the scope of the current project. See for one possible method for handling OOV words on the fly [13].

Since we opt for a pre-defined list of responses, what could these responses be?

One way to still capture a range of possible emotions from users is to rely on a sufficiently rich emotions taxonomy– each response could take the form of ”I'm feeling X” where X is some emotion as described in the taxonomy.

The most basic emotion taxonomy that has been used for many datasets [28, p2] is due to Paul Ekman [12], who proposed six categories of emotions:

1. Joy
2. Anger
3. Fear
4. Sadness
5. Disgust
6. Surprise

However, more recent work in psychology has identified a greater variety of possible emotional responses to various stimuli than described by Ekman's scheme [1]. This has generated datasets that feature a greater range of emotional categories. For instance, the recently developed database GoEmotions, which at time of publishing (June 2020) was ”the largest manually annotated dataset of 58k English Reddit comments, labeled for 27 emotion categories or Neutral.” [28, p1].

For merely illustrative purposes however, it will suffice to work with just the six categories proposed by Ekman. One might also imagine that anything beyond six options would be a bit tiresome for users to scroll through whilst using the app. So, let us suppose that the pre-defined list of responses consists in responses of the form ”I'm feeling X” where X is some emotion as described in the Ekman's taxonomy. For a little more flexibility, we can add the option ”I feel neutral” to the list.

Trend For Interaction Over A Static List Of Resources

A similar approach to detecting which resources (from a static list of available resources) are trending can be adopted using the SAFE protocol. Just as we track user interaction (i.e., taps) over some pre-defined list of responses to a prompt question, we can also track user interaction over some-predefined list of resources that are on offer.

Experiments and illustrations of how this can be done are shown in a notebook in the software archive.

4.0.3 Information Flow in SAFE

As demonstrated in the notebooks, SAFE allows for an aggregator to securely aggregate feature vectors sent by users, without the loss of any accuracy. Moreover, this can be done for even a small number of users.

However, let us more carefully scrutinize the degree of security that the protocol affords us. We can do this with respect to our two main product desidereta:

- **Input Privacy:**
  - User Perspective: able to allow processing on their information without exposing that information
  - Aggregator Perspective: able to run computations on information hidden to them
- **Output Privacy:**
  - User Perspective: able to contribute to an information flow without the fear that someone can reverse engineer from the results your original inputs
  - Aggregator Perspective: able to read results of computation without being able to reverse engineer from the results to the original inputs.

It seems that we have strong input privacy. In fact we may have something even stronger than input privacy because what users allow for processing on (and aggregators end up computing) is not exactly raw data as much as it is distributions that approximate the raw data [31, p3].

However, on reflection we may not have achieved output privacy yet.

First, let us consider the perspective of a user or a group of users. Since a semi-honest security model is assumed, we assume that users will conform to the protocol but may try to discern the private inputs of other users. However, it is not clear how this can be done because all that users end up being able to observe from other users are the distributed random shares from the $[D, -D]$ interval– which is what each user draws from themselves anyway. Thus, there do not seem to be any sensitive information flows from the user's perspective.

The other party to consider is the aggregator. In this case, there could be some privacy-threatening information flows the protocol allows for. In particular, the aggregator may be able to infer narrow bounds on user's privates values given what they know and receive in the protocol's information flows.

As a first step toward considering what information flows that are possible, let us consider a simplified case where one user has a feature vector of a single-dimension, and they simply draw a share from the $[D, -D]$ to obfuscate their feature vector by adding it to that value. Say that the interval is $[-10, 10]$. These are all known by the aggregator. The aggregator also knows that feature vector values are in the range $[0, 1]$.

Now, suppose that some user has a private feature vector value 0.3. Going through the SAFE protocol, they end up sending an obfuscated feature vector to the aggregator -9.6. What can the aggregator infer from this feature vector based off what he knows?

The aggregator knows that this feature vector's value can be expressed in terms of the following equation:

$$Z + Y = -9.6$$

where Y is a random variable that represents the given user's private value, and Z is a random variable that represents the random share that was drawn from the known interval.

It seems that the aggregator could infer the following: since $Y \in [0, 1]$, and $Z \in [-10, 10]$, $Y \leq 0.4$ (since if it is > 0.4, that would result in a Z value that exceeds -10.) This bound is not far from the actual private value, 0.3.

Now, this toy illustration does not represent what actually occurs in the SAFE protocol. Z is not a single share drawn from the $[-D, D]$ interval but instead a sum of $N - 1$ shares, and the Y variable is not the raw value but yet another share derived from operations done on the private value using more random shares.

As the authors note,

> ”Only [the obfuscated feature vector of user i] may leak some information about v_i and only the aggregator will receive this private input. But this leakage is considerably mitigated against, since $N - 1$ random shares from $N - 1$ users are combined with the Nth share of user i to compute this private input.” [31, p4]

Thus the randomness introduced by the share creation and distribution makes the potential for information leakage much less than described in the toy case.

However, it points to a source of information leakage in the protocol: the $[-D, D]$ interval. Let us now consider a more realistic example.

If the $[-D, D]$ interval has a fixed value for each round, and aggregation rounds are carried out consistently enough, the aggregator could have enough information to carry out statistical attacks to infer bounds on private values.

Suppose there are 3 users in a secure aggregation round. According to the protocol, a given user (let us designate them as u_0, will generate random vectors (shares) for the other two users. Let us designate the other users as u_1 and u_2, and the shares that u_0 generates for them respectively as $\tilde{v_0^1}$ as $\tilde{v_0^2}$.

u_0 will also receive shares from u_1 and u_2 ($\tilde{v_1^0}$ and $\tilde{v_2^0}$ respectively), and generates their obfuscated feature vector v_0' as,

$$v_0' = \tilde{v_0^0} + \tilde{v_1^0} + \tilde{v_2^0}$$

where $\tilde{v_0^0}$ is their N^{th} share they derived from their own raw feature vector and the shares they generated for the users,

$$\tilde{v_0^0} = v_0 - \tilde{v_0^1} - \tilde{v_0^2}$$

Substituting the equation for $\tilde{v_0^0}$ into the equation for obfuscation feature vector calculation, we get,

$$v_0' = v_0 - \tilde{v_0^1} - \tilde{v_0^2} + \tilde{v_1^0} + \tilde{v_2^0}$$

which can be re-arranged as,

$$v_0' = v_0 + (\tilde{v_1^0} - \tilde{v_0^1}) + (\tilde{v_2^0} - \tilde{v_0^2})$$

Now, notice that the terms $(\tilde{v_1^0} - \tilde{v_0^1})$ and $(\tilde{v_2^0} - \tilde{v_0^2})$ are sums that have mean 0 if the shares are drawn from the $[-D, D]$ interval in a (nearly) uniform way (which, we can assume would be the case to ensure greater randomness). The aggregator can treat each such term as a random variable $x_i^j - x_j^i$ where x_i^j is the share user i receives from user j and x_j^i is the share they send to user j. They then have a way to start working out statistical properties of the distribution $[-2D, 2D]$ from which this random variable is drawn (e.g. by drawing random vectors from $[-2D, 2D]$ themselves to simulate the drawing of shares), and consequently, a way to derive statistical information about the values that make up Y in the following equation:

$$v_0' = v_0 + Y$$

where Y is a variable for the sum of the $x_i^j - x_j^i$ terms that are used to generate a given user's feature vector.

In this light, it is easy to see how a similar inference to the one described in the toy case is possible. In particular, their knowledge of $[-D, D]$, and consequently $[-2D, 2D]$ allows them to simulate drawing shares and determine empirical distributions of statements that can statistically rule out some regions of the feature values.

It is desirable then to consider if we can mitigate against this information flow to prevent any sort of inferences the aggregator might be able to make.

4.0.4 Decreasing Information Flow in SAFE

In this section, I propose a way to mitigate the kind of privacy-threatening information flow in the SAFE protocol. As we shall see, this procedure has limitations and costs. One will need to weigh the extent to which the information flows described in the previous section are worth the costs involved.

The intent is not to be thorough or complete; each step in the suggested procedure will throw up a host of issues that, to fully resolve, would take us beyond the scope of the project. The aim is to sketch a *proof of concept* and nudge in the direction of how to develop this approach more fully.

Description of Randomization Procedure

The basic idea is to randomize the $[-D, D]$ interval from which users draw shares in each round of the protocol by devising a way for users (and not the aggregator) to draw the same random D value for each round.

One way to do this would be to:

1. Assume all users begin with some shared secret seed
2. Using this shared seed, they generate a float value using some shared Pseudorandom Generator

Pseudorandom Generators (PRNG) are algorithms that generate a sequence of numbers that approximate a sequence of truly random numbers [11]. They are 'pseudorandom' because the sequence they generate is entirely determined by the initial seed value that is passed in. This seed value initialises the internal state of the PRNG. PRNGs use their internal state in their algorithms to generate the pseudorandom output. After each generation of an output, their internal state gets updated.
The important point is that if users use the same seed for the same PRNG, each time they invoke the PRNG, they will end up with the same output. That means if users can all seed the same PRNG with the same seed, they can derive the same random output. This means they will effectively have the same random D value to constitute $[-D, D]$ interval from which they draw their random shares.

Consequently, so long as the shared secret seed remains a secret to the aggregator, each protocol run will be done with a D value that the aggregator does not know, thus preventing inferences on the basis of $[-D, D]$.

However, we should consider what this would really take. Let us consider each step of this procedure now.

Step 1: Shared Secret Seed

Unfortunately, things are not very simple. Establishing a shared secret seed is not a trivial task.

In this section, we consider ways to do so. Ultimately, no protocol comes without limitations and additional complexities. So, these are important considerations when deciding whether or not the greater security afforded by this proposed algorithm is worth the costs.

Diffe-Helman Key Exchange

A well-established protocol for establishing some secure secret in an insecure channel (i.e., a communication channel where adversaries are able to see what all that is publically communicated on the channel) is the Diffe-Helman Key Exchange. This section's basic description of the protocol relies on [10].

For illustrative purposes, consider we have just two users in a SAFE protocol run: Alice and Bob.

1. Both users publically agree on a p and g value, where p is some prime and g is the primitive root modulo p or *group generator* for the multiplicative group of integers modulo p.
2. They each choose a secret integer: let a denote Alice's secret and b denote Bob's secret.
3. Alice computes $A = g^a \; mod \; p$ and sends A to Bob
4. Bob computes $B = g^b \; mod \; p$ and sends B to Alice
5. Alice computes $B^a \; mod \; p$
6. Bob computes $A^b \; mod \; p$.

Because

$$S = (g^a \; mod \; p)^b \; mod \; p = (g^b \; mod \; p)^a \; mod \; p$$

both Alice and Bob would have arrived at the same value, S.

Even if the aggregator knows g, p, A and B, it is very hard for them to derive the S that Alice and Bob now share, so long as a and b remain hidden.

How hard is 'very hard'? The hardness of deriving S consists in solving the *discrete logarithmic problem*, a problem deemed very hard for even modern supercomputers [10].

N Users

Of course, our use case is not likely to involve just two users, but many, many more.

Assuming we try to simply scale the bare version of the protocol, we end up with a fairly untenable growth in complexity. More specifically, with N users, each user will do N exponentiations, meaning that there'll be a total of N^2 exponentiations for N users.

An exponential growth in operations is probably undesirable even if the use-case is constrained to a relatively small scale. So, let us consider alternative protocols.

Key Distribution

An alternative to having each user communicate with every other user is to designate some trusted server or chairman who distributes the secret seed to each user. However, this, among other things, merely introduces another third party and for whom undesirable information flows (assuming a semi-honest model) could result from. So, this option is a non-starter.

A Scalable Key Agreement Protocol

One alternative protocol comes from [26]. Tseng presents a multi-party key agreement protocol which allows N users to collaboratively establish a common key without a designated chairman.

The general idea behind the protocol is this: each user begins with two broadcast round. In each broadcast round, they calculate some random variable labelled with their user-id and send it to all other participants. The results of the first broadcast round are used to calculate the results distributed during the second broadcast round.

After receiving the second broadcast round variates from all other users, each user can compute two checks to see if there is any malicious participant in the protocol run. This check is user-specific, so pinpointing who the malicious participant is can be done. However, to save computational costs, users could instead bypass these initial checks and compute the common key K according to some formula. This would be what happens if the two checks raise no issues anyway. (Since SAFE operates with a semi-honest security model, this bypass can be done.)

Then, they check if they can decrypt encrypted conference messages. If not, then there is a malicious particpant who has disrupted the conference and they must run the fault deduction procedure, and oust the malicious participant. If users can decrypt conference messages using the constructed key K, that means there are no malicious participants in this conference round.

Tseng proves that the total computational cost of this protocol for each participant is $O(N)$ for N participants, resistant to malicious participants. Moreover, number of messaging rounds and message size stay constant as N grows [26, p477].

Applying to SAFE

This protocol can be nicely applied on top of SAFE because users need to communicate shares to one another anyway in the bare protocol. So, we are simply laying two additional broadcast rounds on top of existing communication infrastructure.

Moreover, there aren't any conference messages for users to decrypt in the SAFE protocol. All that is needed is for users to generate some common seed.

Assuming a semi-honest model (which is what SAFE assumes), this is tenable because all users will conform to the protocol and compute the secret key accordingly. So, we could use the operations proposed in the paper to generate the requisite seed.

A fuller and more rigorous analysis of the various issues would be in order if one really were pressed to ensure both the scalability and security of this procedure run over the SAFE protocol. However, for our purposes we have a protocol that is at least *prima facie* workable, so I will simply assume that the N users have established the secret seed by the means of the key agreement protocol, and consider the next step.

Step 2: Shared PRNG

Above, I mentioned that with the same secret seed and the same PRNG (which can be public), users can generate random D values that an aggregator will not be able to know.

This is technically incorrect. The random D value will only really be secure if a *cryptographically secure* PRNG (CPRNG) is used.

A CPRNG is a PRNG that satisfies two requirements [7]:

1. **pass the next-bit test**: given the first k bits of output from the generator, there should not be a computationally tractable algorithm that can predict the next k+1 bit with probability non-trivially greater than 50%
2. **resilient against state-compromise extensions**: if a part or all of the internal state of the generator becomes known (or correctly inferred), it should not be possible to recover past outputs prior to this leak.

There are CPRNGs available that allow for common seeding [9, p4] [8]. However, in my implementation of this procedure, since the aim is merely proof-of-concept, I will defer this complication and use a regular PRNG.

The PRNG we implement is adapted from Python's random module. It is a PRNG that generates floats given a fixed mantissa bit size, and a varying exponent size. This varying exponent size is chosen by an internal PRNG that is seeded with the shared secret seed as well. The varying exponent size helps to increase the range of possible D values. Implementation details can be found in the Python scripts in the software archive.

4.0.5 Implementation of new SAFE Protocol

The notebook in the software archive showcases a test run whilst randomizing the D value for each round. We note that there are no differences in accuracy– the aggregator is still able to compute the target function despite the randomized D value.

4.0.6 Evaluation of New Protocol

The randomization of D values for each round comes with costs: namely, the communication costs incurred by users have to establish a shared secret seed. The growth in costs being $O(N)$ for N participants is less than ideal, but this is not much different from what is required in the SAFE protocol itself– each user needs to broadcast shares to every other user. So, we are in effect just laying onto existing infrastructures some extra messages and computation. Therefore while costs increase the overall configuration can remain largely the same.

It is still possible for the aggregator to, with enough determination, try to determine statistical information such as the bounds on D values that are generated by the shared PRNG. However, combined with the randomisation inherent in the SAFE protocol itself, it is looking to be unlikely that much information will leak to them as a result of the obfuscated feature vectors. Moreover, the use of a CPRNG will make this task highly intractable.

Overall, one will need to consider whether the increase in communication costs and setup complexity is worth the extra security. By itself, the SAFE protocol makes it sufficiently hard for the aggregator to reverse engineer from the obfuscated inputs to the raw ones. (Recall also that the raw feature vectors are only approximations of raw data, being distributions.) Thus output privacy might be satisfied in a weaker sense with the bare protocol too.

4.0.7 Review of Objectives

We have arrived at a way to compute aggregation-related metrics in a way that satisfy input and output privacy for a mood app use case. This protocol is secure in a semi-honest setting, and can be tailored to the specific scale settings of the app (both in terms of available computation resources and user population).

4.0.8 Summary

In this chapter, we have considered how the SAFE protocol could apply to the use case at length. We have also considered how to mitigate some possibly privacy-threatening information flows.

Chapter 5

Future Work

In this chapter, I gesture at further work that could be done to take the project further.

A point of interest would be to consider **integrations of the different secure computation** methods surveyed in this report. Indeed, often many of the cutting-edge protocols being released of late are of this sort– they are not strictly homomorphic encryption schemes or strictly differentially-private algorithms, but rather creative and effective amalgamations of the various techniques.

Take for instance [23], who propose a system called **Honeycrisp** that facilitates a way to perform differentially private aggregation (of the kind that we mentioned Google does) in a central model, however *without* requiring users to trust the central server that facilitates the computations by providing requisite bandwidth and resources [23, p197]. It does this through ”a cocktail of cryptographic techniques” including ”a form of homomorphic encryption” [23, p197]. As this example shows, combining techniques can help address the limitations of others. We had mentioned that a drawback of the central model of differential privacy was that it required a trusted third party. However, in this instance, because of the HE techniques used, this concern is avoided. Work that uses a mix of methods like these would be of great interest to probe for application to our use case.

Doing this might be the key to another point of interest for further work: enabling **more complex and descriptive metrics** for the use case. As mentioned in 3.0.2, the goal is to maximize the analytical freedom of the analyst whilst protecting user privacy.

The last point of interest has to do with improving the application of SAFE and Bayesian trend detection in our use case. In particular, we would want to preprocess textual data. Further work could be done to implement **some robust natural language procsesing** (NLP) methods for the application. It would be interesting to employ methods to preprocess free-form journal entries that users log in the app, instead of just working with integer encodings of input as our current application does.

To illustrate, consider a user who has the following two separate journal entries:

> I felt angry today. I could not understand why they did that to me. I wished that they'd stop.

and

> I am so joyful. I won't forget the great experience I had today. How could I be angry at them ever?

A simple matcher that matches for occurrences of one of the categories in our defined emotions taxonomy would not be nearly robust enough to yield actual analytical insight. In these two examples, the word 'angry' occurs in two very different semantic contexts– one in a declarative statement; the other in a rhetorical question. It would be desirable to have robust NLP methods that could capture keyword frequences for the SAFE protocol in a way that is *semantically sensitive*.

This would hopefully enable us to follow up on the further work mentioned in 4.0.2. As mentioned in that section, it would be of interest to have methods that could work with a richer emotional taxonomy like those identifed in [1] to identify these sentiments in user entries. We mentioned in that section that recent psychological work has broadened emotional taxonomies beyond the traditional six basic categories we've worked with for application. If the NLP methods that could capture keyword frequencies in a semantically sensitive way could do so for these richer taxonomies, that would be a boon to our project.

Chapter 6

Conclusion

To conclude, we have considered the application of protocols developed in Federated Analytics to a hypothetical use case: a 'mood app' that offers users a way to track their mental and emotional state as well as access to wellness resources. We have considered various techniques in secure computation like differential privacy and homomorphic encryption, as well as a more recently introduced method: Bayesian Federated Analytics. We have seen one way to specify the product desidereta for an app that seeks both user privacy and analytical utility. Moreover we've explored a way to make a particular protocol, SAFE, more secure.

Taking this project further could mean the dawn of a world where the benefits of data collection are present in a way that is consistent with our privacy wants and needs.

Chapter 7

Bibliography

(1) Cowen A, Sauter D, Tracy JL, Keltner D. Mapping the Passions: Toward a High-Dimensional Taxonomy of Emotional Experience and Expression. Psychological science in the public interest : a journal of the American Psychological Society. 2019; 20 (1): 69-90. 10.1177/1529100619850176.

(2) Ramage D, Mazzocchi S. Federated Analytics: Collaborative Data Science without Data Collection. http://ai.googleblog.com/2020/05/federated-analytics-collaborative-data.html [Accessed Aug 31, 2022].

(3) Cheon JH, Kim A, Kim M, Song Y. Homomorphic Encryption for Arithmetic of Approximate Numbers. Advances in Cryptology – ASIACRYPT 2017. Cham: Springer International Publishing; 2017. pp. 409-437.

(4) Paillier P. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. Advances in Cryptology — EUROCRYPT '99. Berlin, Heidelberg: Springer Berlin Heidelberg; 1999. pp. 223-238.

(5) Narayanan A, Shmatikov V. Robust De-anonymization of Large Sparse Datasets. 2008 IEEE Symposium on Security and Privacy (sp 2008) : IEEE; May 2008. pp. 111-125. 10.1109/SP.2008.33.

(6) Rebooting the Web of Trust. Decentralized Public Key Infrastructure. Rebooting Web of Trust; 2015.

(7) Wikipedia. Cryptographically secure pseudorandom number generator. https://en.wikipedia.org/w/index.php?title=Cryptographically_secure_pseudorandom_number_generator&oldid=1095062908 [Accessed Aug 31, 2022].

(8) BLUM M, MICALI S. How to generate cryptographically strong sequences of pseudo-random bits. SIAM journal on computing. 1984; 13 (4): 850-864.

(9) Bonawitz K, Ivanov V, Kreuter B, Marcedone A, McMahan H, Patel S, et al. Practical Secure Aggregation for Privacy-Preserving Machine Learning. Proceedings of the

2017 ACM SIGSAC Conference on computer and communications security : ACM; Oct 30, 2017. pp. 1175-1191. 10.1145/3133956.3133982.

(10) Wikipedia. Diffie–Hellman key exchange. https://en.wikipedia.org/w/index.php?title=Diffie%E2%80%93Hellman_key_exchange&oldid=1105073123 [Accessed Aug 31, 2022].

(11) Wikipedia. Pseudorandom number generator. https://en.wikipedia.org/w/index.php?title=Pseudorandom_number_generator&oldid=1104997760 [Accessed Aug 31, 2022].

(12) Ekman P. "Are there basic emotions?" Psychological review 99 3 (1992): 550-3

(13) Lee W, Song G, Shim Kyuseok, Inverse Document Frequency-Based Word Embedding of Unseen Words for Question Answering Systems. Journal of KIISE (2016). 43. 902-909. 10.5626/JOK.2016.43.8.902.

(14) Paverd A, Martin A, Brown I. Modelling and Automatically Analysing Privacy Properties for Honest-but-Curious Adversaries. Smart Grid Security. Cham: Springer International Publishing; 2014. pp. 1-15.

(15) Wikipedia. X.509. https://en.wikipedia.org/w/index.php?title=X.509&oldid=1104964862 [Accessed Aug 31, 2022].

(16) Wikipedia. Differential privacy. https://en.wikipedia.org/w/index.php?title=Differential_privacy&oldid=1105347059 [Accessed Aug 30, 2022].

(17) Desfontaines D. Differential privacy in (a bit) more detail. https://desfontain.es/privacy/differential-privacy-in-more-detail.html#definition [Accessed August 31, 2022].

(18) Benaissa A, Clark W, OpenMined. Homomorphic Encryption - Homomorphic Encryption Characteristics. https://courses.openmined.org/courses/foundations-of-private-computation/eef7a738-d233-4f42-898b-95689c4ab352/541b0a7c-d542-47eb-87dc-99dd22fdfe13 [Accessed Aug 30, 2022].

(19) Lopardo A, Farrand T, Hall A J, Benaissa A. What is Homomorphic Encryption? https://blog.openmined.org/what-is-homomorphic-encryption/ [Accessed Aug 30, 2022].

(20) OpenMined, Will Clark, Ayoub Benaissa. Homomorphic Encryption - Introduction to Homomorphic Encryption. https://courses.openmined.org/courses/foundations-of-private-computation/eef7a738-d233-4f42-898b-95689c4ab352/7836d4c9-8b52-434b-a706-5984e4da4bc6 [Accessed Aug 30, 2022].

(21) Bonawitz K, Salehi F, Konecny J, McMahan B, Gruteser M. Federated Learning with Autotuned Communication-Efficient Secure Aggregation. 2019 53rd Asilomar Conference on Signals, Systems, and Computers : IEEE; Nov 2019. pp. 1222-1226. 10.1109/IEEECONF44664.2019.9049066.

(22) Art. 1 GDPR - Subject-matter and objectives. https://gdpr.eu/article-1-subject-matter-and-objectives-overview/ [Accessed Aug 30, 2022].

(23) Roth E, Noble D, Falk B, Haeberlen A. Honeycrisp. Proceedings of the 27th ACM Symposium on operating systems principles : ACM; Oct 27, 2019. pp. 196-210. 10.1145/3341301.3359660.

(24) Trask A, Bluemke E, Garfinkel B, Cuervas-Mons C G, Dafoe A. Beyond Privacy Trade-Offs with Structured Transparency. CoRR 2020, abs/2012.08347

(25) Ding B, Kulkarni J, Yekhanin S. Collecting Telemetry Data Privately. Ithaca: Cornell University Library, arXiv.org. 2017. https://search.proquest.com/docview/2076820990

(26) Tseng Y-. A Robust Multi-Party Key Agreement Protocol Resistant to Malicious Participants. Computer journal. 2005; 48 (4): 480-487. 10.1093/comjnl/bxh111.

(27) Reyzin L, Smith A, Yakoubov S. Turning HATE into LOVE: Compact Homomorphic Ad Hoc Threshold Encryption for Scalable MPC. Cyber Security Cryptography and Machine Learning. Cham: Springer International Publishing; 2021. pp. 361-378.

(28) Demszky D, Movshovitz-Attias D, Ko J, Cowen A, Nemade G, Ravi S. GoEmotions: A Dataset of Fine-Grained Emotions. Association for Computational Linguistics; 2020;

(29) Dwork C, Kohli N, Mulligan D. Differential Privacy in Practice: Expose your Epsilons. Journal of Privacy and Confidentiality. 2019; 9 (2): 10.29012/jpc.689.

(30) Hsu J, Gaboardi M, Haeberlen A, Khanna S, Narayan A, Pierce BC, et al. Differential Privacy: An Economic Method for Choosing Epsilon. 2014 IEEE 27th Computer Security Foundations Symposium Ithaca: IEEE; Jul 2014. pp. 398-410. 10.1109/CSF.2014.35.

(31) Chaulwar A, Huth M. Secure Bayesian Federated Analytics for Privacy-Preserving Trend Detection. 2021; https://arxiv.org/abs/2107.13640.

(32) Kairouz P, McMahan HB, Avent B, Bellet A, Bennis M, Nitin Bhagoji A, et al. Advances and Open Problems in Federated Learning. Foundations and trends in machine learning. 2021; 14 (1̆20132): 1-210. 10.1561/2200000083.

(33) Kenthapadi K, Tran T. PriPeARL. Proceedings of the 27th ACM International Conference on information and knowledge management : ACM; Oct 17, 2018. pp. 2183-2191. 10.1145/3269206.3272031.

(34) Nissenbaum H. Privacy as contextual integrity. Washington law review. 2004; 79 (1): 119.

(35) Garrido G M. Global sensitivity for differential privacy from scratch. https://blog.openmined.org/global-sensitivity/ [Accessed Aug 31, 2022].