Private Lives Matter: A Differential Private Functional Encryption Scheme

Alexandros Bakas
Tampere University
Tampere, Finland
alexandros.bakas@tuni.fi

Antonis Michalas
Tampere University
Tampere, Finland
antonios.michalas@tuni.fi

Tassos Dimitriou
Kuweit University
Kuwait City, Kuwait
tassos.dimitriou@ieee.org

ABSTRACT

The use of data combined with tailored statistical analysis have presented a unique opportunity to organizations in diverse fields to observe users' behaviors and needs, and accordingly adapt and fine tune their services. However, in order to offer utilizable, plausible and personalized alternatives to users, this process usually also entails a breach of their privacy. The use of statistical databases for releasing data analytics is growing exponentially, and while many cryptographic methods are utilized to protect the confidentiality of the data – a task that has been ably carried out by many authors over the years – only a few works focus on the problem of privatizing the actual databases. Believing that securing and privatizing databases are two equilateral problems, in this paper we propose a hybrid approach by combining Functional Encryption with the principles of Differential Privacy. Our main goal is not only to design a scheme for processing statistical data and releasing statistics in a privacy-preserving way but also provide a richer, more balanced and comprehensive approach in which data analytics and cryptography go hand in hand with a shift towards increased privacy.

CCS CONCEPTS

• **Security and privacy** → **Privacy-preserving protocols**; **Management and querying of encrypted data**; **Cryptography**.

KEYWORDS

differential privacy, functional encryption, multi-party computation

ACM Reference Format:
Alexandros Bakas, Antonis Michalas, and Tassos Dimitriou. 2022. Private Lives Matter: A Differential Private Functional Encryption Scheme. In *Proceedings of the Twelveth ACM Conference on Data and Application Security and Privacy (CODASPY '22), April 24–27, 2022, Baltimore, MD, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3508398.3511514

This work was partially funded by the ASCLEPIOS Advanced Secure Cloud Encrypted Platform for Internationally Orchestrated Solutions in Healthcare Project No. 826093 EU research project.
This work was partially funded from the Technology Innovation Institute (TII), Abu Dhabi for the project ARROWSMITH: Living (Securely) on the edge.
The first author thanks the Nokia Foundation for the generous support through a Nokia Scholarship.

ACM ISBN 978-1-4503-9220-4/22/04…$15.00
https://doi.org/10.1145/3508398.3511514

1 INTRODUCTION

The continually increasing sophistication of technology is shaping the unceasing evolution of data and analytics. Industries are undergoing persistent digital transformation resulting in an ever increasing amount of data collected. Today, every business relies on small or big data and valuable insights derived from it. Data analytics is highly resourceful when it comes to understanding the target audience and their preferences. Using this information, organizations can easily anticipate customer needs and potentially gain significant competitive advantage in the market. Telecom and financial services industries are the most active early adopters of big data analytics, with technology and healthcare following in the third and fourth place.

The agressive penetration of data analytics inevitably raises companies' concerns regarding the usage of users' data and possible breaches of privacy. For example, a recent study [23] found that 19 out of a sample of 24 general-purpose mobile health apps shared user data with more than 50 unique companies, most of which were data analytics companies. This, along with other older reported privacy attacks [16, 29] are very alarming developments considering that statistical databases are of significant importance for decision making in numerous fields ranging from sports and entertainment to national security. A response to such attacks was presented in [17] with the formalization of differential privacy.

Differential privacy allows sharing information about a dataset, while simultaneously withholding information about individuals. A curator (data owner) creates the database and then periodically releases statistics upon receiving request from an analyst. In order to ensure the individuals' privacy, the curator filters the statistics through a privacy mechanism and replies to the analyst with a noisy result. The results must be presented in a form allowing the analyst to deduce accurate enough results about the dataset, without breaching individuals' privacy. While the problem of privatizing datasets has been thoroughly studied, further securing the datasets through the use of cryptography has not yet drawn much attention. However, this is an issue of paramount importance when the database is outsourced to a possibly malicious cloud service provider (CSP). To the best of our knowledge, the only work that considered this scenario is the one presented in [5], where authors rely on homomorphic encryption (HE) [30] and structured encryption (SE) [25] to design a scheme for private histogram queries. In this paper, we approach a similar problem by using Functional Encryption (FE) as the starting point.

Functional Encryption (FE) is an emerging cryptographic technique that allows selective computations over encrypted data. FE schemes provide a key generation algorithm that outputs decryption keys with remarkable capabilities. More precisely, each decryption key sk_f is associated with a function f. In contrast to traditional cryptographic techniques, using sk_f on a ciphertext $\mathsf{Enc}(x)$ does *not* recover x but a function $f(x)$ – thus keeping the actual value x private. While the first constructions of FE allowed the computation of a function over a *single* ciphertext, more recent works [21] introduced the more general notion of multi-input FE (MIFE). In a MIFE scheme, given ciphertexts $\mathsf{Enc}(x_1), \ldots, \mathsf{Enc}(x_n)$, a user can use sk_f to recover $f(x_1, \ldots, x_n)$. The function f can allow only highly processed forms of data to be learned by the functional key holder. Unfortunately, while MIFE seems to be a perfect fit for many real-life applications – especially cloud-based ones where multiple users store large volumes of data in remote and possibly corrupted entities – most of the works in the field revolve around constructing *generic* schemes that do not support specific functions. Hence, while the concept of FE has the potential to unleash new, creative, useful and emerging applications, from a practical perspective, it still holds a *largely unfulfilled* promise. Having identified the importance of FE and believing that it is a family of modern encryption schemes that can push us into an uncharted technological terrain, we made a first attempt at smoothing out the identified asymmetries betwixt theory and practice.

Contributions. We make the following contributions:

(1) First, we design a MIFE scheme in the public key setting for the sum of a vectors' components and then we generalize our construction to further support the inner product functionality. We also show how our scheme transformed from the single-client to the multi-client setting. This transformation requires the users to perform a Multi-Party Computation (MPC). More precisely, each user generates their own public/private key pair for the same public-key encryption scheme and then they collaborate to calculate a functional decryption key sk_f which is derived from a combination of all the generated private keys. This result is quite remarkable since users generate their private keys locally and independently. As a result, the keys are never exposed to unauthorized parties and thus, no private information about the content of the underlying ciphertexts is revealed. At the same time, sufficient information to generate the functional decryption key is provided without the use of a fully trusted party.

(2) Our second contribution derives from the identified need to create a dialogue between the theoretical concept of FE and real life applications. We tried to provide a pathway towards new prospects that show the direct and realistic applicability of this promising encryption technique when applied to concrete obstacles. To this end, we showed how our MIFE scheme can be used to provide a solution to the problem of designing encrypted private databases. In particular, we present three different solutions two of which remain private under *continual observations*, while our third solution satisfies the traditional definition of differential privacy but in the multi-client setting.

(3) In comparison with the seminal work [5], our scheme offers more functionalities as it allows an analyst to perform different kind of queries, and not only request the value of a counter. This is because our construction is based on FE which is a better fit for such a scenario, and outperfroms HE in terms of efficiency. Moreover, we consider a stronger threat model by allowing the malicious analyst to collude with the CSP in an attempt to remove the noise from the results. Finally, in contrast with the purely theoretical work in [5], we present extensive experiments to prove that enhancing the security of an encrypted dataset with differential privacy does not add significant computational costs.

2 MOTIVATION AND APPLICATION DOMAIN

The use of analytics and data processing has been used productively in various fields, including the healthcare sector (e.g. medical diagnosis), intelligence analysis, finance, safety, military services and many more. However, the importance of performing privacy-preserving analytics is an issue that has lately gained momentum in public's mind. As a result, a significant number of companies are moving towards implementing services that respect users' privacy.

To facilitate the reader's understanding of the motivation, and the type of problem we are trying to solve, we considered a specific example capable of showing the immediate application of our research. In layman's terms, the goal of this work, is to allow authorized users to perform statistical analyses over arbitrary datasets in a privacy-preserving way. To achieve this, we built a functional encryption scheme that can protect users' data and their privacy against both internal (e.g. malicious servers) and external (e.g malicious analysts) attacks.

Our solution utilizes a binary range tree, similar to the one descibed in [14]. The binary range tree is a complete binary tree in which each node represents a numerical range. Moreover, in each node we store the sum of the values stored in its children nodes. In other words, each node contains a partial sum corresponding to a specific range. To release statistics in a privacy-preserving way, this binary mechanism outputs noisy sums. To make things clearer let us consider the following example:

We consider a scenario in which 40 students have enrolled in a university course. After the final exam, the professor grades students. Grades are assigned as numbers in the range (1-8) where 8 corresponds to the highest possible mark. The professor creates a complete binary tree in which all grades are stored. Finally, the tree is outsourced to the university's cloud server. Without loss of generality, we can assume that the binary tree looks like the one in figure 1, where the content of each node refers to the number of students whose grades were in a specific range and each c_x denotes the ciphertext corresponding to a plaintext x. Furthermore, we assume that there exists a service in the university through which authorized users (e.g. an analyst) can evaluate any course based on students' grades. The analyst should be able to execute any query on the server. A query could be of the form *"How many students got a grade between 1 and 7?"*. To answer this query, the server should release the sum of the nodes that correspond to the specified range. In our example, this would be the nodes representing the ranges (1-4), (5-6) and (7). Our goal is to design an encryption scheme that

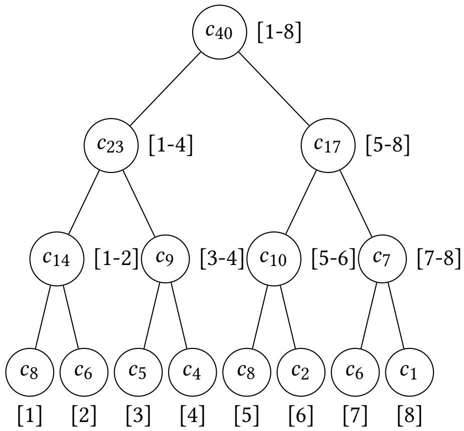

Figure 1: Complete Binary Tree for 40 students graded in the scale [1-8]

will allow an analyst to perform a set of computations on stored data without learning anything about the individual values. In addition to that, our scheme will have to be secure against both internal (compromised university service) and external attacks (corrupted analyst).

We consider for example that Alice, another enrolled student, missed the first exam and participated in a new exam after the tree was already published. Now, an analyst by issuing a query for an average upgrade, could easily deduce Alice's grade just by observing how the average was influenced by Alice's grade. To ensure Alice's privacy, we rely on the *differential privacy under continual observations* model that was formalized independently in [18] and [14]. Differential privacy under continual observations ensures that even if the data is constantly modified and updated, the privacy of the individuals will not be compromised. Our solution focuses on combining Differential privacy under continual observations with FE.

Given that the ciphertexts are produced using an FE scheme, the professor can issue functional decryption keys to any party (i.e. an analyst) that wishes to perform statistics based on the grades of the students. Given such a functional key, the server will be able to output results identical to those where the contents of the nodes were in plaintext. To make the data private apart from just encrypting the individual records we embed a randomized error in the plaintext prior to the encryption.

3 RELATED WORK

Functional Encryption. Functional encryption was formalized as a generalization of public-key encryption in [12]. Since then, numerous studies with general definitions and generic constructions of FE have been proposed [21, 22, 33, 35]. Despite the promising works that have been published, there is a clear lack of works proposing FE schemes supporting specific functions – a necessary step that would allow FE to transcend its limitations and provide the foundations for reaching its full potential. To the best of our knowledge, currently the number of supported functionalities is limited to sums [6, 10], inner products [2–4] and quadratic polynomials [34]. In this work, we propose a MIFE scheme for the sum of a vector's components. We first present a generic construction and then show how to instantiate our scheme from well-studied public-key schemes.

Differential Privacy. Differential privacy is a notion first formalized in [17], where authors focused on ensuring the privacy of individuals. More precicely, it was proved that by adding well-calibrated noise to the data, the presence or absence of an individual's information is *irrelevant* to the output of a database query. Since then, differential privacy has drawn the attention of both researchers [11, 28] and key industrial players such as Google [19, 20], Uber [24] and organizations like the US Census Bureau [27]. Another interesting application of differential privacy was deployed by Apple with the recent release of iOS 14 [1]. In iOS 14 Apple offers its users the ability to enable a feature called "*approximated location*". More specifically, for apps that require location access, a user can choose to share an Approximate Location, which is close to the real location but not precisely spot on, making it harder for apps to keep track of where the user is going and better protecting location privacy.

Continual Observations. Modern applications require data to be constantly modified and updated. Having identified this need as well as its possible difficulties and implications, authors in [18] proposed a new model of differential privacy having in mind scenarios such as real-time traffic analysis, social trends observations and disease outbreaks discovery. In [13], authors proved that continual release of statistics, tend to leak more information. This problem was addressed independently in [18] and [14] and since then, the continual observations model is considered to be the new standard in the field of differential privacy [26, 36]

Crypto-assisted Approaches. Over the past few years researchers have started exploring the possibilities of combining differential privacy with cryptographic primitives in an attempt to provide stronger security guarantees [5, 31, 32]. In particular, in [32] authors proposed a framework for combining differential privacy with cryptography in the *centralized differential privacy* (CDP) model. In the CDP model, data are collected and stored in plaintext in a fully trusted entity. In [32], authors relied on traditional cryptographic techniques to obviate the need of a trusted entity. However, they only managed to replace the trusted entity with two semi-honest servers. Another interesting approach is presented in [31], where authors combine differential privacy with searchable encryption to construct a volume-hiding scheme. Such schemes always return the maximum number of data among all possible queries in an attempt to hide the access pattern. Unfortunately, volume-hiding schemes are designed with single-keyword search in mind, and hence, can not be used for range queries.

Most Relevant Related Work. In [5], authors designed the first private encrypted database, and they proved that their construction is ϵ-differential private in the continual observation model. More specifically, their scheme consists of an encrypted counter that is homomorphically encrypted using the Paillier cryptosystem. A data owner periodically updates the value of the counter and can also release its current *noisy* value. Moreover, they combine their encrypted counter with techniques from structured encryption [25], a generalization of symmetric searchable encryption [7, 8], to design a scheme for private histogram queries. Inspired by their work, we sought to explore the new but already emerging field of private encrypted databases in an attempt to build up a feel for what might be an interesting research direction in which to head in the future. With [5] as a starting point of our research, this work is differentiated as follows:

- Instead of using structured and homomorphic encryption, we base our work on FE [1]. We firmly believe that FE is a cryptographic primitive that squarely fits applications where statistics need to be periodically released. As such, to the best of our knowledge, we construct the first scheme for functionally-encrypted private databases.
- Using FE instead of structured encryption as a basis, allow us to release a number of different statistics and not only the current value of a counter. This is a significant result as it is more applicable to a plethora of applications. Our scheme enables the privacy-preserving publication of statistics that can be computed using a sum. Such statistics may involve, but not limited to, averages, range queries, top-k/bot-k queries etc.
- We consider a stronger threat model. More precisely, in [5], the authors suggest that the noise is added to the data by the cloud service provider. Hence, in their model the cloud must be a trusted entity. Otherwise, an attack in which the server colludes with a malicious analyst can be launched and the actual noise used to mask users data can be easily removed. While this is a simple attack, it cancels out the property of differential privacy and, as a result, any malicious analyst can breach the privacy of individuals. To address this problem, in our approach the error is added to the initial data by the actual data owner prior to outsourcing them to the cloud. Hence, the only information that is leaked to the CSP is the final noisy result.

4 PRELIMINARIES

In this section, we present the necessary notation and definitions needed to follow this paper. The section is divided into five parts: We start by describing the basic notations, then we give definitions about Public-Key Encryption, Functional Encryption, Homomorphic Encryption and differential privacy.

Notation. If $\mathcal{Y}$ is a set, we use $y \xleftarrow{\$} \mathcal{Y}$ if y is chosen uniformly at random from $\mathcal{Y}$. The cardinality of a set $\mathcal{Y}$ is denoted by $|\mathcal{Y}|$. For a positive integer m, $[m]$ denotes the set $\{1, \ldots, m\}$. Vectors are denoted in bold as $\mathbf{x} = [x_1, \ldots, x_n]$. A PPT adversary $\mathcal{ADV}$ is a randomized algorithm for which there exists a polynomial $p(z)$ such that for all input z, the running time of $\mathcal{ADV}(z)$ is bounded by $p(|z|)$. A function $negl(\cdot)$ is called negligible if $\forall\, c \in \mathbb{N}, \exists\, \epsilon_0 \in \mathbb{N}$ such that $\forall\, \epsilon \geq \epsilon_0 : negl(\epsilon) < \epsilon^{-c}$.

Definition 4.1 (Inner Product). The *inner product* (or *dot product*) of $\mathbb{Z}^n$, for two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{Z}^n$ is a function $\langle \cdot, \cdot \rangle$ defined by:

$$f(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle = x_1 y_1 + \cdots + x_n y_n$$

[1]We first present $\mathsf{PLM_H}$ – a scheme that uses homomorphic encryption and then, we move on to present PLM – a variation where homomorphic encryption is not taken into account.

Definition 4.2 (ℓ_2 norm). The ℓ_2 norm of $\mathbb{Z}^n$ for a vector $\mathbf{x} \in \mathbb{Z}^n$ is a function $||\cdot||_2$ defined by:

$$f(x) \ = ||\mathbf{x}||_2 = \sqrt{\sum_{i=1}^{i=n} x_i^2}$$

4.1 Public-Key Encryption

Definition 4.3 (Public-Key Encryption scheme). A public-key encryption scheme PKE for a message space $\mathcal{M}$, consists of three algorithms PKE = (Gen, Enc, Dec). A PKE scheme is said to be correct if:

$$Pr[\mathsf{Dec}(\mathsf{sk}, c) \neq m \mid [(\mathsf{pk}, \mathsf{sk}) \longleftarrow \mathsf{Setip}(1^\lambda)] \\ \wedge [m \in \mathcal{M}] \wedge [c \longleftarrow \mathsf{Enc}(\mathsf{pk}, m)]] = negl(\lambda)$$

To formalize the security of a PKE scheme, we follow the IND-CPA paradigm.

Definition 4.4 (Indistinguishability-Based Security). Let PKE = (Gen, Enc, Dec) be a public-key encryption scheme. We define the following experiments:

$Exp^{s-IND-CPA-\beta}(\mathcal{ADV})$

<u>**Initialize**</u>(λ, x_0, x_1)
$(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$
Return pk
<u>**Challenge**</u>()
$c_\beta \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}, m_\beta)$

<u>**Finalize**</u>(β')
$\beta' = \beta$

The advantage ϵ of $\mathcal{ADV}$ is defined as:

$$\epsilon = \Big| Pr[Exp^{s-ind-CPA-0}(\mathcal{ADV}) = 1 \\ - Pr[Exp^{s-ind-CPA-1}(\mathcal{ADV}) = 1] \Big|$$

We say that PKE is s-IND-CPA-β secure if

$$\epsilon = negl(\lambda)$$

Definition 4.5 (Linear Ciphertext Homomorphism (LCH)). We say that a PKE scheme has *linear ciphertext homomorphism* if:

$$\prod_{i\ =1}^{n} \mathsf{Enc}(\mathsf{pk}_i, x_i) = \mathsf{Enc}\left(\prod_{i=1}^{n} \mathsf{pk_i}, \sum_{i=1}^{n} x_i\right)$$

Definition 4.6 (Linear Key Homomorphism (LKH)). Let $(\mathsf{pk}_1, \mathsf{sk}_1)$ and $(\mathsf{pk}_2, \mathsf{sk}_2)$ be two public/private key pairs that have been generated using PKE.Gen. We say that PKE has *linear key homomorphism* if $\mathsf{sk}_1 + \mathsf{sk}_2$ is a private key to a public key computed as $\mathsf{pk}_1 \cdot \mathsf{pk}_2$.

A direct result of definitions 4.5 and 4.6 is that if a PKE scheme is linear ciphertext and key homomorphic, then the public keys of PKE live in multiplicative group $\mathbb{G}_{pub} = (\mathbb{G}, \cdot, 1_{\mathbb{G}_{pub}})$ and the private keys in an additive group $\mathbb{H}_{priv} = (\mathbb{H}, +, 0_{\mathbb{H}_{priv}})$.

4.2 Multi-Input Functional Encryption

Definition 4.7 (Multi-Input Functional Encryption). A Multi-Input Functional Encryption scheme MIFE for a message space $\mathcal{M}$ is a tuple MIFE = (Setup, Enc, KeyGen, Dec) such that:

- $\mathsf{Setup}(1^\lambda)$: The Setup algorithm is a probabilistic algorithm that on input the security parameter λ, outputs a master public/private key pair (mpk, msk).
- $\mathsf{Enc}(\mathsf{mpk}, \mathsf{x})$: The encryption algorithm Enc is a probabilistic algorithm that on input the master public key mpk and a message $\mathbf{x} = \{x_1, \ldots, x_n\} \in \mathcal{M}$, outputs a ciphertext $\mathbf{c} = \{c_1, \ldots, c_n\}$.
- $\mathsf{KeyGen}(\mathsf{msk}, \mathsf{f})$: The key generation algorithm KeyGen is a deterministic algorithm that on input the master secret key msk and a function f, outputs a functional key sk_f.
- $\mathsf{Dec}(\mathsf{sk}_f, c)$: The decryption algorithm Dec is a deterministic algorithm that on input a functional key sk_f and a ciphertext $\mathbf{c}$, outputs $f(x_1, \ldots, x_n)$.

A MIFE scheme is said to be correct if:

$$Pr[\mathsf{Dec}(\mathsf{sk}_f, \mathbf{c}) \neq f(\mathbf{x}) \mid [(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)] \wedge [\mathbf{c} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathbf{x})] \wedge [\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)]] = negl(\lambda)$$

Just like in the case of PKE we base our security definition on the selective-IND-CPA formalization:

Definition 4.8 (MIFE Indistinguishanility-Based Security). For a MIFE scheme MIFE = (Setup, Enc, KeyGen, Dec) we define the following experiments:

$Exp^{s-IND-FE-CPA-\beta}(\mathcal{ADV})$

Initialize(λ, x_0, x_1)	**Challenge**()
$\mathsf{mpk}, \mathsf{msk} \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$	$\mathbf{c}_\beta \xleftarrow{\$} \mathsf{Enc}(\mathsf{mpk}, \mathbf{x}_\beta)$
$L \leftarrow \emptyset$	**Finalize**(β')
Output mpk	If $\exists f \in L$:
Key Generation(f)	$f(\mathbf{x}_0) \neq f(\mathbf{x}_1)$
$L \leftarrow L \cup \{f\}$	Output $\perp$
$\mathsf{sk}_f \xleftarrow{\$} \mathsf{KeyGen}(\mathsf{msk}, \mathsf{f})$	Else
Output sk_f	$\beta' = \beta$

The advantage ϵ of $\mathcal{ADV}$ is defined as:

$$\epsilon = \Big| Pr[Exp^{s-ind-FE-CPA-0}(\mathcal{ADV}) = 1 - Pr[Exp^{s-ind-FE-CPA-1}(\mathcal{ADV}) = 1] \Big|$$

We say that PKE is s-IND-FE-CPA-β secure if

$$\epsilon = negl(\lambda)$$

4.3 Homomorphic Encryption

A homomorphism is a structure-preserving map between two algebraic structures. Homomorphic Encryption is simply an encryption scheme that retains the homomorphic property. Let us consider the function $\mathsf{Enc} : (\mathbb{G}, \oplus) \longrightarrow (\mathbb{H}, \otimes)$, for some groups $\mathbb{G}$, $\mathbb{H}$ and some operators $\oplus, \otimes$. Then, the function Enc is a homomorphism if and only if for any $x, y \in \mathbb{G}$:

$$\mathsf{Enc}(x \oplus y) = \mathsf{Enc}(x) \otimes \mathsf{Enc}(y) \tag{1}$$

4.4 Differential Privacy

We proceed by providing the main definitions of ϵ-differential privacy and the main properties of the Laplace mechanism. For the rest of the paper, two databases DB and DB' are called neighbouring if they differ at most in one entry.

Definition 4.9 (ϵ-Differential Privacy). A privacy mechanism $\mathcal{M} : \mathbb{N}^{|DB|} \longrightarrow Im(\mathcal{M})$ is ϵ-Differentially private if $\forall\, \mathcal{S} \subset Im(\mathcal{M})$ and $\forall$ neighboring databases $DB, DB' \in \mathbb{N}^{|\mathcal{D}|}$:

$$Pr[\mathcal{M}(DB) \in \mathcal{S}] \;\leq e^{\epsilon} Pr[\mathcal{M}(DB') \in \mathcal{S}]$$

It needs to be noted, that the above definition assumes a *static* database and a curator who must reply to queries non-interactively. In our approach, the database is *dynamic* and a mechanism must update the published statistics as new data arrived. To this end, we rely on the continual observations model of differential privacy. However, to work on the continual observations model, we first need to formalize the curator operations. To do so, we use a similar formalization to the one presented in [14]. More precisely, we assume that curator's operations are given by an input stream $\boldsymbol{\sigma} \in \{0, 1\}^{\mathbb{N}}$. The bit $\boldsymbol{\sigma}(t)$, denotes the occurrence of an event at time t. We consider two cases of update: we assume that the only update possible, is to increase or decrease the value of a database entry.

Definition 4.10 (ϵ-Differential privacy under Continual Observations). A privacy mechanism $\mathcal{M} : \mathbb{N}^{|DB|} \longrightarrow Im(\mathcal{M})$ is ϵ-Differentially private under continual observations if $\forall\, \mathcal{S} \subset Im(\mathcal{M})$, $\forall$ neighboring databases DB, DB' and for all neighbouring sequences of curator operations $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_n)$ and $\boldsymbol{\sigma'} = (\sigma'_1, \ldots, \sigma'_n)$:

$$Pr[\mathcal{M}(DB_1), \ldots \mathcal{M}(DB_n) \in \mathcal{S}] \\ \leq e^{\epsilon} Pr[\mathcal{M}(DB'_1), \ldots \mathcal{M}(DB'_n) \in \mathcal{S}]$$

In this work we only consider two cases of update. In particular, we assume that the only update possible, is to increase or decrease the value of a database entry.

Apart from being private, we would also like the private mechanism to be useful. In other words, we would like $\mathcal{M}$ to return well approximated results after any update.

Definition 4.11. A mechanism $\mathcal{M}$ is said to be (a, δ)-useful at time t, if for any string $\boldsymbol{\sigma}$ with probability at leat $1 - \delta$, we have $|\sum_1^t \sigma(t) - \mathcal{M}(\sigma(t))| \leq a$.

One of the most used privacy mechanisms in literature is the Laplace mechanism, in which the noise is drawn form the Laplace distribution. We use $Lap(b)$ to denote the Laplace distribution with mean 0 and variance $2b^2$. Its probability density function is given by $x \leftarrow \frac{1}{2b} exp\left(-\frac{|x|}{b}\right)$.

We are now ready to proceed with the definition of the Laplace Mechanism [17].

Definition 4.12 (Laplace Mechanism). Given a query $q : \mathbb{N}^{|DB|} \longrightarrow \mathbb{R}$, the Laplace Mechanism is:

$$M_L(DB, q, \epsilon) = q(DB) + Y_i,$$

where $Y_i \sim Lap(b)$

A proof showing that the Laplace Mechanism is ϵ-differentially private can be found in [17]. In particular in [17], the authors proved the following:

Lemma 4.13 (The Laplace mechanism maintains ϵ-differential privacy). *Let $\alpha, \beta \in \mathbb{R}$ such that $|\alpha - \beta| \leq 1$. Moreover, let $e \sim Lap\left(\frac{1}{\epsilon}\right)$. Then $\forall$ measurable subsets $S \subseteq \mathbb{R}$:*

$$Pr[\alpha + e \in S] \;\leq exp(\epsilon) \cdot Pr[\beta + e \in S]$$

A private mechanism $\mathcal{M}$ is said to be B-bounded if it only accepts strings $\boldsymbol{\sigma}$ of length B.

We will now present two important results from [14] that are crucial for our work:

Lemma 4.14 (Sum of Independent Laplace Distributions). *Suppose e_i's are independent random variables, where each e_i has Laplace distribution $Lap(b_i)$. Suppose $Y = \sum_i e_i$, and $b_m = max(b_i)$. Let $v \geq \sqrt{\sum_i b_i^2}$ and $0 < \lambda < \frac{2\sqrt{2}v^2}{b_m}$. Then $Pr[Y > \lambda] \leq exp\left(-\frac{\lambda^2}{8v^2}\right)$*

Corollary 4.15 (Measure Concentration). *Let Y, v, b_i, b_m be defined as in Lemma 4.14. Then if we set $v = \sqrt{\sum_i b_i^2} \cdot \sqrt{ln\frac{2}{\delta}}$ we get that Y is at most $O\left(\sqrt{\sum_i b_i^2}\log\left(\frac{1}{\delta}\right)\right)$*

The proofs for both Lemma 4.14 and Corollary 4.15 can be found in [14].

5 MULTI-INPUT FUNCTIONAL ENCRYPTION FOR SUMS

In this Section, we present MIFE_{sum} – a functional encryption scheme for the sum of a vector's components $\mathbf{x} = \{x_1, \ldots, x_n\}$.

Construction. Let PKE = (Gen, Enc, Dec) be an IND-CPA secure cryptosystem, that also fulfils the LCH and LKE properties. Then we define our MIFE_{sum} as MIFE = (Setup, Enc, KeyGen, Dec) where:

(1) $\mathsf{Setup}(1^\lambda, n)$: The setup algorithm invokes the PKE's key generation algorithm Gen and generates n public/private key pairs as $(\mathsf{pk_1}, \mathsf{sk_1}), (\mathsf{pk_2}, \mathsf{sk_2}) \ldots, (\mathsf{pk_n}, \mathsf{sk_n})$. The public keys are then used to create and output a master public/private key pair (mpk, msk), where mpk = $(\mathsf{params}, \mathsf{pk_1}, \ldots, \mathsf{pk_n})$ and msk = $(\mathsf{sk_1}, \ldots, \mathsf{sk_n})$[2].
(2) $\mathsf{Enc}(\mathsf{mpk}, \mathbf{x})$: The encryption algorithm Enc, takes as input the master public key mpk and a vector $\mathbf{x}$ and outputs $\mathbf{c} = \{c_1, \ldots, c_n\}$, where $c_i = \mathsf{Enc}(\mathsf{pk_i}, x_i)$.
(3) KeyGen(msk): The key generation algorithm, takes as input the master secret key msk and outputs a functional key sk_{sum} as $\mathsf{sk}_{sum} = \sum_1^n \mathsf{sk_i}$[3].
(4) $\mathsf{Dec}(\mathsf{sk}_{sum}, \mathbf{c})$: The decryption algorithm takes as input the functional key sk_{sum} and an encrypted vector $\mathbf{c}$ and outputs $\mathsf{PKE.Dec}(\mathsf{sk}_{sum}, \prod_{i=1}^{n} \mathbf{c})$.

Correctness. The correctness of our construction follows directly since:

$$\mathsf{MIFE_{sum}.Dec}\ (\mathsf{sk}_{sum}, \mathbf{c}) = \mathsf{PKE.Dec}\left(\mathsf{sk}_{sum}, \prod_{i=1}^{n} \mathsf{PKE.Enc}(\mathsf{pk_i}, x_i)\right)$$
$$= \mathsf{PKE.Dec}\left(\mathsf{sk}_{sum}, \mathsf{PKE.Enc}(\prod_{i=1}^{n} \mathsf{pk_i}, \sum_{i=1}^{n} x_i)\right) = \sum_{i=1}^{n} x_i$$

where we used the LCH property. Since the LKE property holds, we know that sk_{sum} is a valid secret key that decrypts $\prod_{i=1}^{n} \mathbf{c}$.

Theorem 5.1 (Selective Indistinguishability). *Let* PKE *be an IND-CPA secure public key cryptosystem that is linear-key and linear-ciphertext homomorphic. Moreover, let* MIFE_{sum} *be our Multi-Input Functional Encryption scheme for the sum of a vector's components which is obtained through* PKE. *Then* MIFE_{sum} *is s-IND-FE-CPA secure.*

Proof Overview

(1) $\mathcal{ADV}_{PKE}$ sends $(0, \mu)$ to the challenger C.
(2) C flips a random coin, and sends (c_b, pk_C), back to $\mathcal{ADV}_{PKE}$, where $b \in \{0, \mu\}$.
(3) $\mathcal{ADV}_{PKE}$ invokes $\mathcal{ADV}_{MIFE}$ on mpk and receives two messages x_0, x_1.
(4) $\mathcal{ADV}_{MIFE}$ $\mathcal{ADV}_{PKE}$ for functional keys for vectors $\mathbf{x_1}, \ldots, \mathbf{x_n}$, such that $\|x_i\|_1 = \|x_j\|_1, \forall\, i, j \in [1, n]$.
(5) $\mathcal{ADV}_{PKE}$ flips a random coin, and sends c_β back to $\mathcal{ADV}_{MIFE}$.
(6) $\mathcal{ADV}_{MIFE}$ outputs a bit a_1.
(7) $\mathcal{ADV}_{PKE}$ outputs a bit a_2.

Figure 2: Sketch of our Security Proof for MIFE

Proof. To prove the security of our construction, we will show that the s-IND-FE-CPA security game is indistinguishable from a game in which a challenger C encrypts a random linear combination of the challenge messages whose coefficients sum up to one. Let $\mathcal{ADV}_{MIFE}$ be an adversary that breaks the IND-FE-CPA security of MIFE. Then, we will show that there exists an adversary $\mathcal{ADV}_{PKE}$ that breaks the IND-CPA security of PKE. We assume that two different games run independently but simultaneously. The first game is the one described in definition 4.4, in which $\mathcal{ADV}_{PKE}$ plays against a challenger C. The second game is the s-IND-FE-CPA game (definition 4.8), in which $\mathcal{ADV}_{PKE}$ acts as the challenger against $\mathcal{ADV}_{MIFE}$. We show that $\mathcal{ADV}_{PKE}$ can perfectly simulate the environment for $\mathcal{ADV}_{MIFE}$, and at the same time infer enough information to break the IND-CPA security of PKE. In particular, if ϵ_{MIFE} is the advantage of $\mathcal{ADV}_{MIFE}$ and ϵ_{PKE} the advantage of $\mathcal{ADV}_{PKE}$, we will prove that $\epsilon_{MIFE} \leq \epsilon_{PKE}$.

$\mathcal{ADV}_{PKE}$ initiates the game by sending $(0, \mu)$ to the challenger C where μ is a random element in the message space of PKE. Upon reception, C generates a $(\mathsf{pk}_C, \mathsf{sk}_C)$ key pair, encrypts one of them at random using pk_C and replies to $\mathcal{ADV}_{PKE}$ with (c_b, pk_C). Upon reception, $\mathcal{ADV}_{PKE}$ invokes $\mathcal{ADV}_{MIFE}$ and receives two messages $\mathbf{x_0}$ and $\mathbf{x_1}$. Recall that $\mathcal{ADV}_{MIFE}$ can only ask for functional decryption keys for vectors $\mathbf{x_0}$ and $\mathbf{x_1}$ such that $\|\mathbf{x_0}\|_1 = \|\mathbf{x_1}\|_1$. Hence, $\mathcal{ADV}_{MIFE}$ is allowed to issue queries to a vector space $V \subset \mathcal{M}$ such that $\forall\, \mathbf{x_i} \in V : \|\mathbf{x_i}\|_1 = 0$ and is not able to decrypt vectors in other vector spaces. An overview of our proof if given in figure 2.

Public Key Generation. To generate mpk, $\mathcal{ADV}_{PKE}$ first selects $n-1$ random vectors $\mathbf{z_1}, \ldots, \mathbf{z_{n-1}}$ such that $\|\mathbf{z_i}\|_1 = 0, \forall\, i \in [1, n-1]$, and then produces a basis of V as $(\mathbf{x_1} - \mathbf{x_0}, \mathbf{z_1}, \ldots \mathbf{z_{n-1}})$. Finally, $\mathcal{ADV}_{PKE}$ writes the canonical vectors of the basis as:

[2] The public parameters params depend on the choice of the PKE scheme.
[3] We omit the description of the function since in this case we are only focusing on the sum

$$\mathbf{e} = \alpha_i(\mathbf{x_1} - \mathbf{x_0}) + \sum_{1}^{n-1} \mathbf{z_j} \tag{2}$$

where $\alpha_i = \frac{\mathbf{x_{1,i}} - \mathbf{x_{0,i}}}{\|\mathbf{x_{1,i}} - \mathbf{x_{0,i}}\|_2^2}$.

As a next step, $\mathcal{ADV}_{PKE}$ runs $(\mathsf{pk}_{\mathsf{z_j}}, \mathsf{sk}_{\mathsf{z_j}}) \leftarrow \mathsf{PKE.Gen}, \forall\ j \in [n-1]$ and finally sets:

$$\mathsf{pk_i} = \mathsf{pk}_C{}^{\alpha_i} \prod_{j=1}^{n-1} \mathsf{pk}_{z_j}, \tag{3}$$

where pk_C is the public key received from C. The master public key is then:

$$\mathsf{mpk} = (\mathsf{pk_i})_{i\in[n]} \tag{4}$$

Moreover, to ensure that $\|\mathbf{x_1} - \mathbf{x_0}\|_2^2 \neq 0 \mod q$ in the message space $\mathcal{M} = \{0, \dots N-1\} \subseteq \mathbb{Z}_q$, we need to set q to be a prime larger than N^2. Finally, due to the LKH property of the public-key encryption scheme, $\mathcal{ADV}_{PKE}$ is unknowingly setting

$$\mathsf{sk}_i = \alpha_i \mathsf{sk}_C + \sum_{1}^{n-1} \mathsf{sk}_{\mathsf{z_j}} \tag{5}$$

where sk_C is not known to $\mathcal{ADV}_{PKE}$.

Challenge ciphertext. Upon receiving $\mathbf{x_0}$ and $\mathbf{x_1}$, from $\mathcal{ADV}_{MIFE}$, $\mathcal{ADV}_{PKE}$ is expected to pick a $\beta \in \{0, 1\}$ and reply with $\mathbf{c}_\beta$. However, instead of encrypting x_β using the corresponding public key, $\mathcal{ADV}_{PKE}$ sets the challenge ciphertext $\mathbf{c}$ to be:

$$c = c_b^{\alpha} \cdot \mathsf{PKE.Enc}\left(\prod_{i=1}^{n-1} \mathsf{pk}_{z_j}, 0\right) \cdot \mathsf{PKE.Enc}(1_{G_{pub}}, \mathbf{x}_\beta) \tag{6}$$

where $1_{G_{pub}}$ is the identity element of the group G_{pub}. Finally $\mathcal{ADV}_{PKE}$ replies to $\mathcal{ADV}_{MIFE}$ with c.

Functional Keys. To generate a functional key for a vector $\mathbf{x} \in V$, $\mathcal{ADV}_{PKE}$ simply sets:

$$\mathsf{sk}_{sum} = \sum_{1}^{n-1} \mathsf{sk}_{\mathsf{z_i}} \tag{7}$$

The game concludes as follows: $\mathcal{ADV}_{MIFE}$ correctly guesses β which implies that $\mathcal{ADV}_{PKE}$ guesses that C encrypted 0 or $\mathcal{ADV}_{MIFE}$ fails to guess β and $\mathcal{ADV}_{PKE}$ guesses that C encrypted μ. What remains to be done is show that $\mathcal{ADV}_{PKE}$ simulated correctly the environment for $\mathcal{ADV}_{MIFE}$. We distinguish two cases based on C's choice:

(1) C encrypted 0. In this case, the challenge ciphertext from equation 6 becomes:

$$\begin{aligned} c &= \mathsf{PKE.Enc}(\mathsf{pk}_C, 0)^{\alpha} \cdot \mathsf{PKE.Enc}\left(\prod_{i=1}^{n-1} \mathsf{pk}_{z_i}, 0\right) \\ &\quad \cdot \mathsf{PKE.Enc}(1_{G_{pub}}, \mathbf{x}_\beta) \\ &= \mathsf{PKE.Enc}\left(\mathsf{pk}_C{}^{\alpha} \cdot \prod_{i=1}^{n-1} \mathsf{pk}_z \cdot 1_{G_{pub}}, 0 + 0 + \mathbf{x}_\beta\right) \\ &= \mathsf{PKE.Enc}(\mathsf{pk_i}, \mathbf{x}_\beta) \end{aligned} \tag{8}$$

Hence, it can be seen that in this case $\mathcal{ADV}_{PKE}$ perfectly simulates the environment for $\mathcal{ADV}_{MIFE}$. As a result, if $\mathcal{ADV}_{MIFE}$ can correctly guess β with advantage ϵ, then $\mathcal{ADV}_{PKE}$ will guess that C encrypted 0, with exactly the same ϵ.

(2) C encrypted μ. In this case, the challenge ciphertext from equation 6 becomes:

$$\begin{aligned} c &= \mathsf{PKE.Enc}(\mathsf{pk}_C, \mu)^{\alpha} \cdot \mathsf{PKE.Enc}\left(\prod_{i=1}^{n-1} \mathsf{pk}_{z_i}, 0\right) \\ &\quad \cdot \mathsf{PKE.Enc}(1_{G_{pub}}, \mathbf{x}_\beta) \\ &= \mathsf{PKE.Enc}\left(\mathsf{pk}_C{}^{\alpha} \cdot \prod_{i=1}^{n-1} \mathsf{pk}_z \cdot 1_{G_{pub}}, \alpha\mu + 0 + \mathbf{x}_\beta\right) \\ &= \mathsf{PKE.Enc}(\mathsf{pk_i}, \alpha\mu + \mathbf{x}_\beta) = \mathsf{PKE.Enc}(\mathsf{pk_i}, x') \end{aligned} \tag{9}$$

where x' is a vector defined as:

$$\begin{aligned} x' &= x_\beta + \alpha\mu \\ &= \frac{\mu}{\|x_1 - x_0\|_2^2}(\mathbf{x_1} - \mathbf{x_0}) + \mathbf{x}_\beta \\ &= \frac{\mu}{\|x_1 - x_0\|_2^2}(\mathbf{x_1} - \mathbf{x_0}) + \mathbf{x_0} + \beta(\mathbf{x_1} - \mathbf{x_0}) \end{aligned} \tag{10}$$

Setting $u = \frac{\mu}{\|x_1 - x_0\|_2^2} + \beta$, yields $\mathbf{x'} = u\mathbf{x_1} + (1-u)\mathbf{x_0}$, which is the message that corresponds to the challenge ciphertext. Note that $\mathbf{x'} \in V$, since $\mu \in V$, and hence $\mathbf{x'}$ is a linear combination of elements that live in V. Hence, we conclude that the challenge ciphertext is a valid ciphertext for $\mathbf{x'} = u\mathbf{x_1} + (1-u)\mathbf{x_0}$, which is a random linear combination of $\mathbf{x_0}$ and $\mathbf{x_1}$ whose coefficients sum up to one. Finally, β is information theoretically hidden as the distribution of u is independent of β. As a result, the advantage of $\mathcal{ADV}_{PKE}$ is 0 when a non-zero vector is encrypted by C.

To calculate the overall advantage of $\mathcal{ADV}_{PKE}$, we simply need to sum its advantage for each case. Hence, we have that $\mathcal{ADV}_{PKE}$'s advantage is $\epsilon + 0 = \epsilon$. However, recall that ϵ is defined to be the advantage of $\mathcal{ADV}_{MIFE}$ against MIFE. Thus, the best advantage one can get against the CPA security of MIFE_{sum} is bounded by the best advantage one can get against IND-CPA PKE. □

Functional Keys for Vectors in Different Vector Spaces: As already mentioned, $\mathcal{ADV}_{MIFE}$ is only allowed to request functional keys for vectors living in a vector space $V \subset M$, where $\forall x_i \in V : \|x_i\|_1 = 0$. Notice that by allowing $\mathcal{ADV}_{MIFE}$ to obtain functional decryption keys for vectors $x \notin V$, our scheme can be trivially broken. However, this would imply that $\mathcal{ADV}_{PKE}$ can generate such functional decryption keys, which is *impossible* since $\mathcal{ADV}_{PKE}$ does not know sk_C. Hence, the generated functional keys can only decrypt ciphertexts whose plaintexts are elements of V. This is a valid assumption since otherwise, we would demand security in a scenario where the master secret key is known to the adversary.

5.1 From Sums to Inner Products

We will now show how our construction for the sums can be generalized to further support the inner-product functionality. More precisely, given two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{Z}^n$, we allow the computation of their inner product $\langle \mathbf{x}, \mathbf{y} \rangle = x_1 y_1 + \cdots + x_n y_n$.

Construction for Inner Products. Let $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an IND-CPA secure cryptosystem, that also fulfils the LCH and LKE properties. Then we define our MIFE scheme for inner products, $\mathsf{MIFE_{IP}}$, as $\mathsf{MIFE_{IP}} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ where:

(1) $\mathsf{Setup}(1^\lambda, n)$: The setup algorithm invokes the PKE's Gen algorithm and generates n public and private key pairs as $(\mathsf{pk_1}, \mathsf{sk_1}), (\mathsf{pk_2}, \mathsf{sk_2}), \ldots, (\mathsf{pk_n}, \mathsf{sk_n})$. The generated public keys are then used to create and output a master public/private key pair $(\mathsf{mpk}, \mathsf{msk})$, where $\mathsf{mpk} = (\mathsf{params}, \mathsf{pk_1}, \ldots, \mathsf{pk_n})$ and $\mathsf{msk} = (\mathsf{sk_1}, \ldots, \mathsf{sk_n})$[4].
(2) $\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}, \mathbf{y})$: The encryption algorithm Enc, takes as input the master public key mpk and two vectors $\mathbf{x}, \mathbf{y}$ and outputs $\mathbf{c} = \{c_1, \ldots, c_n\}$, where $c_i = \mathsf{Enc}(\mathsf{pk_i}, x_i)^{y_i}$.
(3) $\mathsf{KeyGen}(\mathsf{msk}, \mathbf{y})$: The key generation algorithm, takes as input the master secret key msk and the vector $\mathbf{y}$ and outputs a functional key $\mathsf{sk_y}$ as $\mathsf{sk_y} = \langle y_i, \mathsf{sk_i} \rangle$[5].
(4) $\mathsf{Dec}(\mathsf{sk}_{sum}, \mathbf{c})$: The decryption algorithm takes as input the functional key $\mathsf{sk_y}$ and an encrypted vector $\mathbf{c}$ and outputs $\mathsf{PKE.Dec}\left(\mathsf{sk_y}, \prod_{i=1}^{n} \mathbf{c}\right)$.

The correctness and the IND-CPA security of $\mathsf{MIFE_{IP}}$, are derived directly from the corresponding properties of MIFE_{sum}. However, in the security proof, we now require that the adversary asks for functional decryption keys, for vectors $\mathbf{y}$ such that $\langle \mathbf{x_0}, \mathbf{y} \rangle = \langle \mathbf{x_1}, \mathbf{y} \rangle$. This implies that the adversary can ask decryption keys for vectors $\mathbf{y}$ that live in the vector space spanned by $(\mathbf{x_1} - \mathbf{x_0})^\perp$ (i.e. they are orhtogonal to $(\mathbf{x_1} - \mathbf{x_0})$). Hence, the adversary will not be able to decrypt any inner product for a vector $\mathbf{y}$ such that $\mathbf{y} \notin (x_1 - x_0)^\perp$.

6 FUNCTIONALLY-ENCRYPTED AND DIFFERENTIALLY PRIVATE DYNAMIC DATABASES

We are now ready to present $\mathsf{PLM_H}$ and PLM; two schemes for functionally-encrypted private databases. To do so, we will use our MIFE_{sum} construction from Section 5 and the binary mechanism presented in [14]. In the first part of the section, we discuss $\mathsf{PLM_H}$, our first approach to the problem that relies on the homomorphic property of the public-key encryption scheme PKE. Then, we present PLM – a modified version of $\mathsf{PLM_H}$ that does *not* require homomorphic encryption. Both versions share the same architecture, presented below:

Architecture. We assume the existence of the following entities:

- **Curator (C)**: **C** is responsible for creating an encrypted and private database. **C** outsources the database to a CSP where it will be stored. Moreover, **C** can issue update queries to the CSP update specific entries of the database. To do so, **C** keeps locally the latest version of the database.
- **Analyst (A)**: **A** is an analyst that can perform statistics on the data stored in the CSP.
- **CSP**: A cloud service provider that stores an encrypted database. The CSP releases statistics upon request of the analyst.

Both of our constructions are proven to be differentially private in the continual observations model. In other words, by assuming two neighbouring sequence operations $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_n)$ and $\boldsymbol{\sigma'} = (\sigma'_1, \ldots, \sigma'_n)$, applied on two neighbouring databases DB and DB', we ensure that after n updates, the presence or absence of an individual does *not* affect the result of a query.

6.1 PLM using Homomorphic Encryption

Overview. At a high-level, our construction works as follows: A curator **C** generates a binary tree similar to the one described in Section 2 in which each node contains a noisy value where the noise is sampled from the Laplace distribution. Then, **C** encrypts each noisy value using MIFE_{sum} with an additive homomorphic public-key encryption scheme PKE. The result, is an encrypted binary tree which is then outsourced and stored in the CSP. To update the values stored in the tree, **C** uses the homomorphic property of PKE. At any given time, and after the tree has been stored in the CSP, an analyst **C** can use the values stored in the tree to generate statistics in a privacy-preserving way. To do so, **A** first contacts the curator and requests a functional decryption key. Upon reception of the key, the analyst forwards it to the CSP who will reply with a sum corresponding to the analyst's query. In our construction, errors are sampled as $e \sim Lap\left(\frac{1}{\epsilon'}\right)$, where $\epsilon' = \frac{\epsilon}{\log N}$ and N is the total number of nodes in the tree. The reason for this, is that these parameters help us achieve ϵ-differential privacy as we will see in the proof of theorem 6.1.

Formal Construction. $\mathsf{PLM_H}$ makes use of the MIFE_{sum} and a public-key encryption scheme $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ that satisfies the LCH and LKH properties. Moreover, the encryption function of PKE must be additively homomorphic. $\mathsf{PLM_H}$ is then defined as $\mathsf{PLM_H} = (\mathsf{Setup}, \mathsf{Update}, \mathsf{Read})$. Our construction is illustrated in figure 3 and works as follows:

Setup : Setup is a two party protocol between C and the CSP. C outputs a complete binary tree T with n nodes and adds Laplacian noise to the content of each node. As a next step, C runs $\mathsf{MIFE}_{sum}.\mathsf{Setup}$ and generates n public/private key pairs $(\mathsf{pk_i}, \mathsf{sk_i})$. Finally, C encrypts each node i using a public key pk_i and T is outsourced to the CSP.

Update : Update is a two party protocol between **C** and the CSP. To update the content of a node, **C** makes use of the homomorphic property of PKE.Enc. More precisely, assuming that **C** wishes to add a value κ to the content of a leaf node i, she first finds the path from the root of the tree to the leaf i. For every node j in the path, C samples a distinct $e_j \sim Lap\left(\frac{1}{\epsilon'}\right)$ and computes $\kappa'_j = \kappa_j + e_j$. As a next step, C encrypts each κ'_j using pk_j . Apart from that, **C** samples a fresh noise e_m for every other node m of the tree and encrypts it using $\mathsf{pk_m}$. Finally, for each node of the tree, **C** sends a pair (n, c_n) to the CSP. Upon reception, the CSP updates each node i using c_i by computing $c'_n = c_{n_{old}} \cdot c_n$, where $c_{n_{old}}$ the current content of the node n.

Read : Read is a three party protocol between **C**, **A** and the CSP. This protocol is initiated by **A** who wishes to perform statistics on

[4]The public parameters params depend on the choice of the PKE scheme.
[5]In this case, the description of the function f, is the vector $\mathbf{y}$.

$\mathsf{PLM_H}$

Let MIFE_{sum} be our construction from Section 5 instantiated with a public key encryption scheme PKE that satisfies the LCH, LKH properties and is additively homomorphic. Moreover, assume that the total number of nodes in the tree is N and let $\epsilon' \leftarrow \epsilon/logN$.

$\underline{\mathsf{PLM_H}.\mathsf{Setup}}$
C generates a binary tree T
For each node $i \in T$:
  C runs MIFE_{sum}.Setup
  C samples $e_i \leftarrow Lap\left(\frac{1}{\epsilon'}\right)$
  C calculates $a_i' = a_i + \epsilon$, where a_i is the content of the node i
  C computes $c_{a_i'} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_i, a_i')$
  C replaces the content of node i with $c_{a_i'}$
CSP receives T

$\underline{\mathsf{PLM_H}.\mathsf{Update}}$
$L = \{\}$
C wishes to update the content of a leaf k by either adding or subtracting to it a $\kappa \in \mathbb{R}$
For every node j in the path from the root of T to the leaf i:
  C samples $e_j \leftarrow Lap(\frac{1}{\epsilon'})$ and computes $\kappa' = \kappa + e_j$
  C computes $c_{\kappa'} = \mathsf{PKE.Enc}(\mathsf{pk}_j, \kappa')$
  $L = L \cup \{j, c_{\kappa'}\}$
For every other node $m \in T$:
  C samples an error $e_m \leftarrow Lap\left(\frac{1}{\epsilon'}\right)$
  C computes $c_{e_m} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_m, e_m)$
  $L = L \cup \{m, c_m\}$
C sends L to the CSP
For each ciphertext $c_n \in L$
  CSP computes $c_n' = c_{n_{old}} \cdot c_n$
  CSP replaces the content of node n with c_n'

$\underline{\mathsf{PLM_H}.\mathsf{Read}}$
A request a functional key sk_f from C for a function f
C constructs $\mathsf{sk}_f = \sum \mathsf{sk}_i$ where each i is picked based on the description of f
C sends sk_f to A
A sends a query to CSP including a range $[a, b]$ and sk_f
CSP finds the appropriate nodes $n_1, \ldots, n_j$ and runs $\mathsf{MIFE}_{sum}.\mathsf{Dec}(\mathsf{sk_f}, n_1, \ldots, n_j)$
A receives a noisy result

Figure 3: PLM based on Homomorphic Encryption

the data stored in the CSP. As a first step, **A** contacts **C** and requests a functional decryption key for a function f. This function can be the sum of all nodes, a top-k/bot-k query, or any function that can be computed using a sum. Upon receiving the query, **C** generates the functional decryption key sk_f by summing up the appropriate secret keys that were generated during MIFE_{sum}.Setup. **C** forwards sk_f to the CSP and receives back a noisy result.

6.2 PLM without Homomorphic Encryption

We will now present PLM; a modified version of $\mathsf{PLM_H}$ in which we show that homomorphic encryption can be dropped entirely. This is a significant improvement in terms of complexity and efficiency as homomorphic operations are particularly computationally expensive. Just like in PLM_H, errors are sampled as $e \sim Lap\left(\frac{1}{\epsilon'}\right)$. PLM is illustrated in figure 4 and works as follows:
Setup : PLM.Setup is identical to $\mathsf{PLM_H}$.Setup
Update : When **C** wishes to update the content of a node in the tree T, she proceeds as in the case of $\mathsf{PLM_H}$.Update. However, instead of sending the list L to the CSP, **C** now sends directly the updated tree T' to the CSP. Upon reception, the CSP deletes T and stores T'. This is possible, because, as already discussed, **C** always keeps a version of the current tree locally.
Read : PLM.Read is identical to $\mathsf{PLM_H}$.Read

PLM

Let MIFE_{sum} be instantiated with a public key encryption scheme PKE that satisfies the LCH and LKH properties. Moreover, assume that the total number of nodes in the tree is N and let $\epsilon' \leftarrow \epsilon/logN$.

$\underline{\mathsf{PLM.Setup}}$
Identical to $\mathsf{PLM_H}$.Setup

$\underline{\mathsf{PLM_H}.\mathsf{Update}}$
$L = \{\}$
C wishes to update the content of a leaf k by either adding or subtracting to it a $\kappa \in \mathbb{R}$
C runs PLM.Setup where the content of the leaf k and every node in the path from the root to the leaf k is updated by either adding or subtracting κ. C outputs a tree T'
CSP receives T', deletes T and stores T'

$\underline{\mathsf{PLM.Read}}$
Identical to $\mathsf{PLM_H}$.Read

Figure 4: PLM without Homomorphic Encryption

6.3 Privacy and Utility

We will now prove that both $\mathsf{PLM_H}$ and PLM satisfy ϵ-differential privacy. Moreover, we prove the usefulness of our two schemes.

Theorem 6.1. *The* Read *algorithm in both* $\mathsf{PLM_H}$ *and* PLM *is ϵ-differentially private as per definition 4.10.*

Proof. Suppose a privacy mechanism $\mathcal{M}$ adds $Lap\,(1/\epsilon)$ noise to every sum before releasing it. Since in each update operation, we add freshly sampled noise to every node of the tree, and since each node contains a sum, we conclude that N sums are affected by a factor of $1/\epsilon$ during every update. Hence, if the tree has a total of N nodes, then $\mathcal{M}$ achieves $N \cdot \epsilon$-differential privacy. To achieve ϵ-differential privacy, we can scale appropriately to $\epsilon' = \frac{\epsilon}{N}$. Observe, that each sum maintains $\frac{\epsilon}{\log T}$ since Laplace mechanism maintains differential privacy. Now, if the mechanism $\mathcal{M}$ adds $Lap\left(\frac{1}{\epsilon'}\right)$ noise to each released sum, we get: $Lap\left(\frac{1}{\epsilon'}\right) = Lap\left(\frac{1}{\frac{\epsilon}{N}}\right)$ □

Since, as we said before, adding $Lap\,(1/\epsilon)$ results to $N{\cdot}\epsilon$-differential privacy, by adding $Lap\,(1/\epsilon')$ results to:

$$N \cdot \frac{\epsilon}{N} = \epsilon\text{-differential privacy}$$

Theorem 6.2. *For each update $\sigma(t)$ at time t, both* $\mathsf{PLM_H}$ *and* PLM *are* $\left(O\left(\frac{1}{\epsilon}\right) \cdot \sqrt{N} \cdot \sqrt{\log N} \cdot \log \frac{1}{\delta}, \delta\right)$*-useful at time t.*

Proof. Let ϵ_i be independent random variables, where each e_i has Laplace distribution $Lap\left(\frac{\log N}{\epsilon}\right)$. Note that $|\sum_1^t \sigma(t) - \mathcal{M}(\sigma(t))| = \sum_1^t e_i$. Hence, using Corollary 4.15, with $b_i = \frac{\log N}{\epsilon}$ we get that:

$$\sum_i e_i \leq O\left(\sqrt{\sum_i \left(\frac{\log N}{\epsilon}\right)^2 \log\left(\frac{1}{\delta}\right)}\right)$$

From which we conclude that both $\mathsf{PLM_H}$ and PLM are

$$\left(O\left(\frac{1}{\epsilon}\right) \cdot N \cdot \sqrt{\log N} \cdot \log \frac{1}{\delta}, \delta\right) - useful. \tag{11}$$

□

6.4 Comparison between $\mathsf{PLM_H}$ and PLM

Since, $\mathsf{PLM_H}$ requires the CSP to perform a homomorphic encryption on every node of the tree, we conclude that, $\mathsf{PLM_H}$ requires to perform $O\left(2^{\log_2 N}\right) = O(N)$ homomorphic encryptions. As a result, we see that PLM outperforms $\mathsf{PLM_H}$ by a factor of N. It is important to note that despite its inefficiencies, $\mathsf{PLM_H}$ is a very good candidate for a multi-client model, in which each node of the tree is encrypted by a different user. However, dealing with the updates in a multi-client scenario is not trivial as it would require the cooperation of every user to embed a freshly sampled noise to each node of the tree. As such, we leave it for future work. However, in the next section, we present a scheme that offers ϵ-differential privacy, in the multi-client model when the database is static.

7 A STATIC PRIVATE DATABASE IN THE MULTI-CLIENT MODEL

In this section, we are addressing the multi-client model and design a scheme for a functionally encrypted private database. Both in PLM and $\mathsf{PLM_H}$, the Setup function is executed by a single curator who has total control over all ciphertexts. However, if we consider that each ciphertext in the database is generated by a different user, then generating a functional decryption key is *not* a trivial problem. To address this problem, we design $\mathsf{PLM_M}$ in which we show how several users can cooperate to generate such a key. Our solution is based on an MPC similar to the one presented in [15].

Probelm Statement (MIFE$_{sum}$ with Multi-Client Support). *Let $\mathcal{U} = \{u_1, \ldots, u_n\}$ be a set of users. Each user $u_j \in \mathcal{U}$ generates a public/private key pair* $(\mathsf{pk}_j, \mathsf{sk}_j)$ *for a public-key encryption scheme satisfying the properties defined in definitions 4.5 and 4.6, and uses* pk_j *to encrypt a message x_j. Additionally, assume that all generated ciphertexts are outsourced and stored in a remote location operated by an untrusted (i.e. possible malicious) CSP. Furthermore, we assume that an analyst (e.g. a user from $\mathcal{U}$) wishes to perform statistics on the data stored on the CSP. Our multi-client construction shows how a legitimate analyst can do this without learning any valuable information about the individual values x_j.*

MPC. Upon request of $\mathbf{A}$, each user $u_i \in \mathcal{U}$ generates a random number r_i and breaks it into n shares as $r_i = r_{i,1} + \cdots + r_{i,n}$. Each share will be sent to a different user from the set $\mathcal{U} = \{u_1, \ldots, u_n\}$. Upon receiving $n-1$ different shares, each user u_i mask her private key sk_i as $b_i = \mathsf{sk}_i + r_i - \sum_{j=1}^n r_{j,i}$, and sends the masked key to $\mathbf{A}$. When $\mathbf{A}$ has gathered all the masked keys, she computes sk_{sum} as $\mathsf{sk}_{sum} = \sum_1^n \mathrm{b_i}$. The MPC is illustrated in algorithm 1.

Algorithm 1 MPC

A generates r_i
A writes r_i as $r_i = r_{i,1} + \cdots + r_{i,n}$
for $j \in [n-1]$ **do**
  A sends $r_{i,j}$ to u_j
for all $u_j \in \mathcal{U}/\{\mathrm{A}\}$ **do**
  u_j generates r_j
  u_j writes r_j as $r_j = r_{j,1} + \cdots + r_{j,n}$
  u_j computes the masked values of the key as $b_j = \mathsf{sk}_j + r_j - \sum_{k=1}^n r_{k,i}$
  u_j sends b_j to A
  u_j sends $r_{j,\ell}$ to $u_\ell, \forall \ell \neq j$.
A computes $\sum_1^n s_j = \sum_1^n \mathsf{sk}_j = \mathsf{sk}_{sum}$

It is important to highlight that splitting and distributing the random numbers to the different users, allows the users to work in parallel for the MPC and hence, we overcome the limitations that would emerge by using a ring topology.

We are now ready to describe $\mathsf{PLM_M}$. Our construction consists of two algorithms such that $\mathsf{PLM_M} = (\mathsf{Setup}, \mathsf{Read})$. PLM_M is illustrated in figure 5 and works as follows:

Setup : During the Setup, each u_i generates a public/private key pair $(\mathrm{pk_i}, \mathrm{sk_i})$ for a linear ciphertext and key homomorphic public key encryption scheme PKE. Apart from that, u_i picks a x_i that wishes to encrypt. Before the encryption, u_i samples $e_i \sim Lap\,(1/\epsilon)$ and calculates $x_i' = x_i + e_i$. Finally, u_i runs $c_i \leftarrow \mathsf{PKE.Enc}(\mathrm{pk_i}, x_i')$ and sends c_i to the CSP. When all users are done, the CSP has received n distinct ciphertexts.

Read : The analyst $\mathbf{A}$ first needs to generate the functional key sk_{sum}. To do so, $\mathbf{A}$ initiates the MPC described in algorithm 1. As soon as $\mathbf{A}$ retrieves the functional key sk_{sum}, she simply forwards it to the CSP. Upon reception, the CSP runs $\mathsf{Dec}(\mathsf{sk}_{sum}, c_1, \ldots, c_n) = \sum_1^n x_i'$ and sends the result to $\mathbf{A}$.

Showing that the $\mathsf{PLM_M}$.Read maintains e-differential privacy is trivial as it is a direct result of the fact that the Laplace mechanism maintains differential privacy. In other words, to prove that $\mathsf{PLM_M}$ is ϵ-differentially private one needs to prove that the Laplace mechanism is ϵ-differential private.

Theorem 7.1. *The* Read *protocol defined in* $\mathsf{PLM_M}$.Read *is ϵ-differential private.*

Theorem 7.2. *Let $\mathcal{ADV}$ be an adversary that corrupts at most $n-2$ users out of those in $\mathcal{U}$. Then, $\mathcal{ADV}$ cannot infer any information about the secret keys of the legitimate users.*

Due to space limitations, the proof can be found in the full version of the paper [9].

$\mathsf{PLM_M}$

Let MIFE_{sum} be our construction from Section 5 instantiated with a public key encryption scheme PKE that satisfies the LCH and LKH properties. Moreover, let $e_i \sim Lap\,(1/\epsilon)$.

$\mathsf{PLM_M}$.Setup

For $i \in [n]$

u_i runs $(\mathsf{pk_i}, \mathsf{sk_i}) \leftarrow (\mathsf{PKE.Gen})$
u_i samples $e_i \sim Lap\left(\frac{1}{\epsilon}\right)$
u_i computes $x'_i = x_i + e_i$
u_i runs $c_i \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_i, x'_i)$
u_i sends c_i to the CSP

$\mathsf{PLM_M}$.Read

A initiates the MPC protocol form algorithm 1 and receives sk_{sum}
A forwards sk to the CSP
CSP runs $\mathsf{MIFE}_{sum}.\mathsf{Dec}(\mathsf{sk}_{sum}, c_1, \ldots, c_n) \rightarrow \sum_1^n x'_i$
CSP sends $\sum_1^n x'_i$ to **A**

Figure 5: Multi-Client PLM

8 EXPERIMENTAL EVALUATION

Below, we present the measured processing time of the experiments in our construction. For the implementation of our MIFE scheme, we used ElGamal as the public-key encryption scheme. All experiments were executed on a Lenovo T470p with 2.81 GHz Intel Core i7 and 32GB RAM running Windows 10, 64-bit. The construction was implemented in Python 3.9.4 using the `PyCryptoDome` and `numpy` libraries. For the experiments we mainly focused on *(1) The Setup time* and *(2) The generation of functional decryption keys*. The results presented are the average processing time computed after 50 runs of each experiment. Our results support our claim that using differential privacy on top of encryption, does not add a noticeable increase to the total processing time.

Setup phase This phase consists of *(1)* Generating and populating a binary tree with plaintext values and *(2)* Embed noise and encrypt each node of the tree. We used randomly generated datasets of different size consisting of real numbers (100, 500, 1000 and 10000).

- Tree generation: The tree was implemented as a list, where each element on the list corresponded to a leaf on the tree. To make our construction compatible with continuous variables, each leaf represented a subinterval in the interval defined by subtracting the min value of the dataset from the max. Hence, the value of each leaf represents the number of values in a specific interval. We measured the time to generate the tree for different datasets and number of nodes. The total number of nodes can be calculated by the number of leaves, since a complete binary tree with 2^n leaves, consists of a total $2^{n+1} - 1$ nodes. Our experiments were conducted for $n = 5, 6, 7, 8, 9, 10$, resulting in binary trees of sizes $63, 127, 255, 511, 1023$ and 2047 respectively. This procedure did *not* add any noticeable burden to the overall processing time, as in the worst case scenario, the tree generation took less than a second.
- Encryption and Noise: After the tree generation, we had to *(1)* Add noise to the value of each node, *(2)* Generate an ElGamal key pair for each node and *(3)* Encrypt all noised contents. This part of the experiments depended only on the number of nodes and **not** on the size of the dataset. Embedding Laplacian noise to the tree's nodes was much faster than key generation and encryption. The time for adding noise varied from 0.16ms for 63 nodes tree to 5.7ms for a 2,047 nodes tree. In contrast, generating 63 and 2,047 ElGamal key pairs of size 1,024bits took 0.28s and 9.14s respectively. Similarly, encrypting 63 and 2,047 nodes was measured at 0.377s and 9.4574s respectively. It is important to note that the key generation times were significantly accelerated since all keys were sampled from the same group $\mathbb{G}$, using the same generator g. Despite this acceleration, as shown in table 1, the key generation and tree node encryption time comprised more than 99% of the total processing time. This is an important result, as it proves that further securing an encrypted dataset with differential privacy does *not* add significant computational burden. In Figure 6, we see that the overall setup time is $O(n)$.

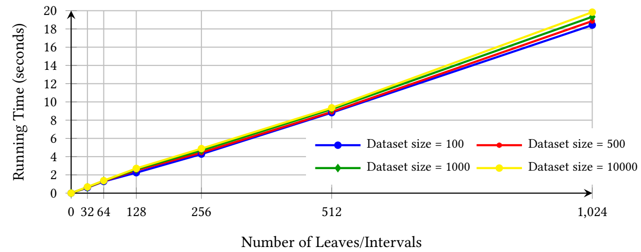

Figure 6: Total Setup Time

Functional Decryption Key Generation. In this phase of our experiments, we assumed that the analyst performs queries of the form *"How many values lie in the interval $I = [a, b]$"*. To reply to such a query, we must: *(1)* Find all the subintervals I_i such that $\sum_i I_i = I$ and retrieve the ciphertext that lies in each interval and *(2)* Retrieve the private key that corresponds to each interval and compute the functional decryption key. To prove the efficiency of our construction, we assumed the analyst makes a complex query of the form *"How many values lie in the first interval and how many values lie in the second interval and . . . and how many values lie in the last interval"*. To answer such a query, all we have to do is retrieve the value from the root of the tree and decrypt it. To capture a, fully unrealistic/worst-case scenario, we measured the time required to answer such a query sequentially, that is we only retrieved values from tree's sibling leaves, and for each pair of siblings, generated a functional decryption key. The time required to retrieve all the leaf values from a 1,024 leaves tree, was 0.9777s. When we exploited the tree structure to reply to such a query, the required time was imperceptible. Similarly, the required time for generating functional keys was also negligible. For reference, the average time to create a functional decryption key as the sum of 1,000 private keys, was 0.119ms.

9 CONCLUSION

Achieving competitive advantage in today's market is largely a function of deploying better and more advanced analytics. Analytics' expansion is driven by systematic, fully automated data collection and capture of behavioural data from multiple touch points. Companies use this data not only to see the current consumer choices and behaviours but to shape the future ones. However the systems

Time Required for each Function					
Total Number of Tree Nodes	Laplacian Noise	Key Generation	Encryption	Tree Generation (Dataset size = 10000)	Total Setup Time
63	0.16ms	0.28s	0.37s	31ms	0.6811s
127	0.35ms	0.56s	0.75s	60.6ms	1.3709s
255	0.67ms	1.41s	1.43s	137.8ms	2.7087s
511	1.4ms	2.27s	2.36s	247.8ms	4.8792s
1021	2.8ms	4.57s	4.73	483.9ms	9.3511s
2047	5.7ms	9.14s	9.45s	942.9ms	19.5386s

Table 1: Processing time for all Setup functions for the most demanding dataset.

using statistical models to analyze users' behaviours are incorporating proxies which are often *inexact* and *unfair*. As big data is here to stay, and statistical models increasingly will be the tools to rely on, bringing transparency into the game is crucial. Creating schemes capable of performing high accuracy predictions, whilst being unable to learn anything about processed data, would inevitably ensure improved fairness.

REFERENCES

[1] [n.d.]. Apple. https://support.apple.com/en-us/HT211808.

[2] Michel Abdalla, Dario , Dario Fiore, Romain Gay, and Bogdan Ursu. 2018. Multi-Input Functional Encryption for Inner Products: Function-Hiding Realizations and Constructions Without Pairings. In *Advances in Cryptology – CRYPTO 2018*.

[3] Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. 2015. Simple functional encryption schemes for inner products. In *IACR International Workshop on Public Key Cryptography*. Springer, 733–751.

[4] Michel Abdalla, Romain Gay, Mariana Raykova, and Hoeteck Wee. [n.d.]. Multi-input inner-product functional encryption from pairings. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer.

[5] Archita Agarwal, Maurice Herlihy, Seny Kamara, and Tarik Moataz. 2019. Encrypted Databases for Differential Privacy. *Proceedings on Privacy Enhancing Technologies* 2019, 3 (2019), 170–190.

[6] Alexandros Bakas and Antonis Michalas. 2020. Multi-input functional encryption: efficient applications from symmetric primitives. In *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, 1105–1112.

[7] Alexandros Bakas and Antonis Michalas. 2020. Power range: Forward private multi-client symmetric searchable encryption with range queries support. In *2020 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 1–7.

[8] Alexandros Bakas and Antonis Michalas. 2021. Nowhere to Leak: A Multi-client Forward and Backward Private Symmetric Searchable Encryption Scheme. In *IFIP Annual Conference on Data and Applications Security and Privacy*. Springer, 84–95.

[9] Alexandtros Bakas, Antonis Michalas, and Tassos Dimitriou. 2021. Private Lives Matter: A Differential Private Functional Encryption Scheme (extended version). Cryptology ePrint Archive, Report 2021/1692. https://ia.cr/2021/1692.

[10] Alexandros Bakas, Antonis Michalas, and Amjad Ullah. 2020. (f) unctional sifting: A privacy-preserving reputation system through multi-input functional encryption. In *Nordic Conference on Secure IT Systems*. Springer, 111–126.

[11] Avrim Blum, Katrina Ligett, and Aaron Roth. 2013. A learning theory approach to noninteractive database privacy. *Journal of the ACM (JACM)* 60, 2 (2013), 1–25.

[12] Dan Boneh, Amit Sahai, and Brent Waters. 2011. Functional encryption: Definitions and challenges. In *Theory of Cryptography Conference*. Springer, 253–273.

[13] Joseph A Calandrino, Ann Kilzer, Arvind Narayanan, Edward W Felten, and Vitaly Shmatikov. [n.d.]. " You might also like:" Privacy risks of collaborative filtering. In *2011 IEEE symposium on security and privacy*. 231–246.

[14] T.-H. Hubert Chan, Elaine Shi, and Dawn Song. 2011. Private and Continual Release of Statistics. *ACM Trans. Inf. Syst. Secur.* 14, 3, Article 26 (Nov. 2011).

[15] Tassos Dimitriou and Antonis Michalas. 2014. Multi-party trust computation in decentralized environments in the presence of malicious adversaries. *Ad Hoc Networks* 15 (2014), 53 – 66.

[16] Irit Dinur and Kobbi Nissim. 2003. Revealing information while preserving privacy. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 202–210.

[17] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*. Springer, 265–284.

[18] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N Rothblum. 2010. Differential privacy under continual observation. In *Proceedings of the forty-second ACM symposium on Theory of computing*. 715–724.

[19] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*. 1054–1067.

[20] Giulia Fanti, Vasyl Pihur, and Úlfar Erlingsson. 2016. Building a rappor with the unknown: Privacy-preserving learning of associations and data dictionaries. *Proceedings on Privacy Enhancing Technologies* 2016, 3 (2016), 41–61.

[21] Shafi Goldwasser, S Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. 2014. Multi-input functional encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 578–602.

[22] Shafi Goldwasser, Yael Tauman Kalai, Raluca Ada Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. 2013. How to run turing machines on encrypted data. In *Annual Cryptology Conference*. Springer, 536–553.

[23] Quinn Grundy, Kellia Chiu, Fabian Held, Andrea Continella, Lisa Bero, and Ralph Holz. 2019. Data sharing practices of medicines related apps and the mobile ecosystem: traffic, content, and network analysis. *BMJ* 364 (2019). https://doi.org/10.1136/bmj.l920 arXiv:https://www.bmj.com/content/364/bmj.l920.full.pdf

[24] Noah Johnson, Joseph P Near, and Dawn Song. 2018. Towards practical differential privacy for SQL queries. *Proceedings of the VLDB Endowment* 11, 5 (2018).

[25] Seny Kamara, Tarik Moataz, and Olya Ohrimenko. 2018. Structured encryption and leakage suppression. In *Annual International Cryptology Conference*. Springer.

[26] Georgios Kellaris, Stavros Papadopoulos, Xiaokui Xiao, and Dimitris Papadias. 2014. Differentially private event sequences over infinite streams. (2014).

[27] Ashwin Machanavajjhala, Daniel Kifer, John Abowd, Johannes Gehrke, and Lars Vilhuber. 2008. Privacy: Theory meets practice on the map. In *2008 IEEE 24th international conference on data engineering*. IEEE, 277–286.

[28] Frank McSherry and Kunal Talwar. [n.d.]. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science*.

[29] Arvind Narayanan and Vitaly Shmatikov. [n.d.]. Robust de-anonymization of large sparse datasets. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*.

[30] Pascal Paillier. 1999. Public-key cryptosystems based on composite degree residuosity classes. In *International conference on the theory and applications of cryptographic techniques*. Springer, 223–238.

[31] Sarvar Patel, Giuseppe Persiano, Kevin Yeo, and Moti Yung. 2019. Mitigating Leakage in Secure Cloud-Hosted Data Structures: Volume-Hiding for Multi-Maps via Hashing. Cryptology ePrint Archive, Report 2019/1292.

[32] Amrita Roy Chowdhury, Chenghong Wang, Xi He, Ashwin Machanavajjhala, and Somesh Jha. 2020. Crypt: Crypto-assisted differential privacy on untrusted servers. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 603–619.

[33] Amit Sahai and Hakan Seyalioglu. 2010. Worry-free encryption: functional encryption with public keys. In *Proceedings of the 17th ACM conference on Computer and communications security*. 463–472.

[34] Edouard Dufour Sans, Romain Gay, and David Pointcheval. 2018. Reading in the Dark: Classifying Encrypted Digits with Functional Encryption. *IACR Cryptology ePrint Archive* 2018 (2018), 206.

[35] Brent Waters. 2015. A punctured programming approach to adaptively secure functional encryption. In *Annual Cryptology Conference*. Springer, 678–697.

[36] Yonghui Xiao, James Gardner, and Li Xiong. 2012. Dpcube: Releasing differentially private data cubes for health information. In *2012 IEEE 28th International Conference on Data Engineering*. IEEE, 1305–1308.