Enhancing Noisy Functional Encryption for Privacy-Preserving Machine Learning

Linda Scheu-Hachtel and Jasmin Zalonis

Univerity of Mannheim, Mannheim, Germany
`{linda.scheu-hachtel, zalonis}@uni-mannheim.de`

Abstract. Functional encryption (FE) has recently attracted interest in privacy-preserving machine learning (PPML) for its unique ability to compute specific functions on encrypted data. A related line of work focuses on noisy FE, which ensures differential privacy in the output while keeping the data encrypted. We extend the notion of noisy multi-input functional encryption (NMIFE) to (dynamic) noisy multi-client functional encryption ((Dy)NMCFE), which allows for more flexibility in the number of data holders and analyses, while protecting the privacy of the data holder with fine-grained access through the usage of labels.

Following our new definition of DyNMCFE, we present DyNo, a concrete inner-product DyNMCFE scheme. Our scheme captures all the functionalities previously introduced in noisy FE schemes, while being significantly more efficient in terms of space and runtime and fulfilling a stronger security notion by allowing the corruption of clients.

To further prove the applicability of DyNMCFE, we present a protocol for PPML based on DyNo. According to this protocol, we train a privacy-preserving logistic regression.

Keywords: (Noisy) Multi-Client Functional Encryption · Differential Privacy · Privacy-Preserving Analysis.

1 Introduction

Privacy-preserving machine learning (PPML) is crucial for extracting insights from large datasets in sensitive domains like finance, healthcare, and social research. The necessity to protect the privacy of the data holders sparked an extensive research in PPML.

In this work, we consider a medical-inspired scenario, applicable to other contexts as well. We assume the presence of a trusted authority, e.g., an ethics committee, that provides a platform for PPML, for example as a sub-feature of an eHR app [44]. On this platform, data holders with sensitive information can voluntarily register to participate in analyses. Analysts can also register, initiate new studies, and invite data holders to contribute their data securely.

As an example, assume an analyst wants to train a model to predict risk factors for lung cancer. They can open an analysis with requested attributes

"age", "smoker", "BMI", "chronic disease" and "lung cancer". Then, the data holders can submit their personal data for these requested attributes to the analyst, who stores the data under some pre-determined label, or decline/ignore the request. Once enough data is provided, the analyst and other interested parties in the same attributes can run their analysis. However, we assume all of the data holders will only participate if:

i) They only have to provide their data and not actively engage in the analysis.
ii) Their privacy remains protected.

Additionally, we assume that the authority has limited resources and should not receive the plaintexts or learn the result of the analysis. To address ii), we must not only protect the plain data, ensuring *input* privacy, but also require additional measures to safeguard *output* privacy, as the analysis results may leak information [13].

The only common approach to obtain both directly is local differential privacy (LDP) [38]. LDP perturbs data at its source, either through randomized response or addition of noise, such that the original values cannot be determined. This way, the perturbed data can be collected on a public sever, where any analyst can use it without further user involvement. However, this approach requires an enormous amount of perturbation, which severely impacts the utility of any model trained on such data [30, 40].

Therefore, a lot of research focuses on techniques combining mechanisms providing input privacy with global differential privacy (GDP) [20]. Unlike LDP, GDP perturbs the analysis output rather than the raw data, resulting in lower utility loss. A widely used GDP-based approach for PPML is federated learning (FL) [31], but since the raw data remains untouched, additional protection is required when data is outsourced. Other common approaches for PPML combining input privacy with GDP are based on multi-party computation (MPC) [25, 26, 51] and (fully) homomorphic encryption ((F)HE) [24, 27, 28], each offering different trade-offs discussed in Section 3.

Recently, functional encryption (FE) [41, 49, 14], has been widely discussed to be helpful in PPML. FE overcomes the all-or-nothing decryption limitations of standard encryption schemes, as it provides designated decryption keys. A decryption key dk_f is associated with a specific function f and instead of decrypting directly to the plain input, the decryption key applied to the ciphertext only reveals the decryption of $f(x)$ and nothing more. Further, it allows to outsource some of the computational overhead to the analyst, which is necessary if the trusted party has limited resources and the data holders are not actively involved, i.e. i) is fulfilled.

Zalonis et al. [50] introduced the notion of noisy multi-input functional encryption (NMIFE), which is a method to combine GDP and multi-input functional encryption (MIFE). The idea is to equip the functional decryption key dk_f with a noise value ν sampled according to some distribution to obtain $f(x_1, \ldots, x_n) + \nu$ instead of $f(x_1, \ldots, x_n)$. By choosing an appropriate distribution, this is enough to realize GDP. The security of NMIFE schemes ensures that the decryption key does not leak anything about the noise value.

Unfortunately, while a lot of research in FE aims to realize arbitrary functions, i.e., general circuits, these techniques require heavy tools such as indistinguishability obfuscation or polynomial hardness of assumptions on multilinear maps [12, 23, 48]. As a result, these schemes are far from being practical. Currently, efficient (N)MIFE schemes are only feasible for specific families of functions, namely linear functions [6, 17, 39] and quadratic functions [11, 5]. However, all of the latter require pairings on bilinear groups, making them rather slow. Thus, we focus on linear functions, which can already be used for many applications. Specifically, we present a protocol leveraging ideas from FL, which allows the training of, e.g., linear or logistic regression models.

To meet the requirements of the use case presented above, we extend the definition of NMIFE to noisy multi-client functional encryption (NMCFE), similar as from MIFE to multi-client functional encryption (MCFE) [15], to achieve a stronger security notion and inherit additional benefits relevant to our scenario. Unlike NMIFE, NMCFE equips each ciphertext with an additional label. Only those ciphertexts encrypted under the same label can be combined. This enables a more fine-grained access as it prevents the combination of ciphertexts with different labels and thus mitigates mix-and-match attacks. In our scenario, these labels could refer to the name of the study or dataset, but could also be time stamps for a long-term medical study. Additionally, in contrast to NMIFE, the security of NMCFE allows clients to be corrupted, i.e., the adversary may possess the encryption keys of corrupted clients without compromising the security and privacy of the remaining clients.

We further extend NMCFE to a dynamic setting, where data holders can dynamically register to be part of the dynamic NMCFE (DyNMCFE) for future analysis or drop out without the need to set up a new scheme any time the participants change. As all analyses remain within the same scheme and are overseen by one trusted party, e.g. an ethical committee, this party can easily keep the privacy loss of each participating data holder under control.

With DyNMCFE, we obtain a perfectly tailored primitive to our described scenario. We present a concrete, efficient scheme that supports linear functions. Like all other (MI)FE schemes supporting GDP, it only allows one encryption per client. By using labels, we allow one encryption per client-label pair, effectively enabling multiple encryptions per client across different labels. This may still seem like a limitation, however when training ML models using a predefined dataset is common, especially in a medical context and when combining ML with GDP. Therefore if we assume that each client submits its data with specific attributes only once to a dataset associated with a given label, allowing multiple encryption per slot is unnecessary.

Contributions Our contributions are the following.

- We formalize the notion of DyNMCFE and its security. This is a natural combination from regular MCFE and the definition of NMIFE provided in [50] enhanced to a dynamic setting. In particular, we achieve a stronger security notion as NMIFE, as we allow the dynamic registration of clients and corruptions.

- We provide a protocol capturing the described scenario above using DyN-MCFE for linear functions as our main cryptographic building block. This protocol ensures both input and output privacy and can be used to realize efficient PPML.
- Following our new definition and requirements for the protocol, we present DyNo, an efficient DyNMCFE scheme for inner-product functionality, allowing for a polynomial number of clients. Its security solely relies on a pseudorandom function (PRF). This construction is sufficient to provide GDP and more efficient than any previous noisy FE scheme for inner-product functionality. Tested on datasets of different sizes, our implementation operates in milliseconds, whereas concurrent schemes take seconds, if not minutes or hours.
- In addition, we use our protocol to train a differentially private logistic regression on encrypted data, which satisfies input as well as output privacy. This is the first logistic regression realized by an inner-product FE scheme.

Outline The paper is structured as follows. In Section 2, we present the necessary background on DP and the private gradient descent (GD) algorithm. Section 3 gives an overview and comparison of different approaches to realize PPML and their suitability for the specific scenario described above. The class of DyNM-CFE is introduced in Section 4. We propose the protocol for our application in Section 5. Based on the requirements, we present our concrete linear DyNMCFE construction in Section 6. Section 7 contains the benchmarking of our scheme as well as the results of the trained logistic regression. We conclude this work in Section 8.

1.1   Related Work

Noisy Functional Encryption Bakas et al. [8, 9] were the first to combine FE and DP. They used the construction of [4] for inner-product MIFE to obtain GDP queries to an encrypted database. However, they only considered the aggregated sum and not weighted sums. Moreover, they lack a proper security definition for incorporating noise in the decryption key.

Zalonis et al. [50] filled this gap by introducing noisy FE together with the notion of noise-hiding security and a tailored correctness definition. In particular, noise-hiding allows for the function to be known by the decryptor, but it demands the noise to be hidden. This relaxation from function-hiding, where the decryption key leaks nothing about the function itself, allowed them to transform a function-hiding into a more efficient noise-hiding scheme. This scheme allows only one message to be encrypted per client, which is already sufficient for their discussed use case of PPML. However, their scheme is still based on pairings, which are known to be slow. Compared to our construction, their approach is less efficient in both size and runtime and neither supports the corruption of clients nor the dynamic generation of encryption keys.

In an independent work, Escobar et al. [21] also combined GDP and FE. They consider an encrypted dataset to which an adversary may pose queries

via an FE scheme. Their security definition is coined on computational DP, i.e., an adversary cannot efficiently distinguish between the outputs of the queries on neighboring datasets. Our security definition, extended from [50], is much broader, as it only considers GDP as a use case. Moreover, we consider the multi-client setting, whereas they only have a single data holder encrypting the whole dataset. Note that this is also covered by our definition if we set the number of clients to be one.

A comparison between our scheme and the schemes from Zalonis et al. and Escobar et al. in size of the master secret key msk, the encrypted database X with n plaintext of size m and decryption key dk can be found in Table 1. Here, $\mathbb{G}, \mathbb{G}_1$ and $\mathbb{G}_2$ are cyclic groups of order q and $|\cdot|$ denotes the size of their elements. We ignore public information such as the function embedded inside the decryption key, the associated label and client, as those are negligible and part of any protocol. Also, we stress that all of these schemes only allow the encryption of one ciphertext per data holder.

Table 1. Comparison of sizes of all current FE schemes for data in $\mathbb{Z}_q$ realizing GDP, where n is the number of data holders, m is the vector length of the data and λ the security parameter.

	$|\mathsf{msk}|$	$|\mathsf{Enc}(X)|$	$|\mathsf{dk}|$
[50]	$n(m+3)|\mathbb{G}_2|$	$n(m+5)|\mathbb{G}_1|$	$(m+5)|\mathbb{G}_2|$
[21]	3λ	$(nm+2)|\mathbb{G}|$	$4\log q$
Ours	$n\lambda^1$	$nm\log q$	$\log q$

Privacy-Preserving Machine Learning using Functional Encryption There exist some works which combine ML and FE. Although most of them simply focus on prediction [13, 19, 33, 43], some also include training. For instance, Xu et al. [49] employed a 5-layer neural network by observing that the input to the activation function is simply an inner-product of the weights and the data. This inner-product in the first layer is computed using their own presented scheme. All of the other computations are made in the clear. Panzade et al. [41] improved the speed of the scheme using another (function-hiding) IPFE. However, both of them leak the intermediate results and are only selective-IND-CPA secure, whereas our scheme is adaptively secure and provides output privacy.

Recently, Chang et al. [14] combined FL and FE using several MCFE schemes. Their model updates are merged using MCFE, which concludes a lower privacy loss, but still none that provides DP. Moreover, they need to re-encrypt their ciphertexts in each iteration, whereas our data holders only submit their data once and do not have to participate in the analysis.

---

[1] This can be reduced to λ if the clients' keys are generated using, e.g., a PRF. However, this introduces an additional overhead in other algorithms and only makes sense in the non-dynamic version.

2   Preliminaries

2.1   Notation

We always denote the security parameter by $\lambda \in \mathbb{N}$, which parametrizes each scheme and adversary. For integers $m, n \in \mathbb{Z}$, let $[m; n] := \{x \in \mathbb{Z} | m \leq x \leq n\}$ and $[n] := [1; n]$. For $x \in \mathbb{R}$, $\lfloor x \rceil$ is its rounding to the nearest integer. For clarity, we use bold symbols to denote vectors, i.e., $\mathbf{x}$ is a vector, where $\mathbf{x}[j]$ is its jth element. In contrast, objects of some collection that is not regarded as a vector are indexed using subscripts (or superscripts in some cases). For example, $\mathbf{x}_i$ represents a vector, not a component of some vector. If i runs through some index set $[n]$, it means that there are n vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$. If the n objects are scalars (or not explicitly vectors), we will write $x_1, \ldots, x_n$ instead. For a function $f : \mathcal{X}_1 \times \cdots \times \mathcal{X}_n \to \mathcal{Y}$ and some $\nu \in \mathcal{Y}$, we denote by $(f + \nu)(x) := f(x) + \nu$ for all $x \in \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$. For a subset $\Delta \subseteq Y$, we define $f + \Delta = \{f + \nu \mid \nu \in \Delta\}$. For a set $S \subseteq \mathbb{N}$, we sometimes write $f(\{x_i\}_{i \in S}) = f(x_{i_1}, \ldots, x_{i_{|S|}})$ for $i_1 \leq \cdots \leq i_{|S|}$, $i_j \in S, \ j \in [|S|]$.

We use the abbreviation PPT to mean probabilistic polynomial time. A function $\mathtt{negl} : \mathbb{N} \to \mathbb{R}^+$ is said to be negligible if for every $c \in \mathbb{N}$, there exists a $\tau \in \mathbb{N}$ such that for all $x \in \mathbb{N}$ with $x > \tau$, it holds that $|\mathtt{negl}(x)| < 1/x^c$.

For a distribution $\mathbb{D}$ over some set Δ, $\Pr[x \leftarrow \mathbb{D}(\Delta)]$ denotes the probability that x is sampled according to $\mathbb{D}$ over Δ. If $\mathbb{D}(\Delta)$ is clear from the context, we simply write $\mathbb{D}$. For any set Δ, $s \xleftarrow{\$} \Delta$ represents the process of uniformly sampling an element $s \in \Delta$. For two probability distributions $\mathbb{D}$ and $\mathbb{D}'$ over the same domain Δ, we write $\mathbb{D} \equiv \mathbb{D}'$ to indicate that they are equally distributed.

We restrict ourselves to inner-product MIFE schemes. More precisely, we consider functions of the form $f_{\mathbf{y}_1, \ldots, \mathbf{y}_n}(x_1, \ldots, x_n) = \sum_{i \in [n]} \langle \mathbf{x}_i, \mathbf{y}_i \rangle$, i.e., each function can be identified with a vector $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n) \in (\mathbb{Z}^m)^n$. The class of multi-input inner-product functionalities over $\mathbb{Z}_q$ is defined by $\mathcal{F}_{q,n}^m := \{f_{\mathbf{y}_1, \ldots, \mathbf{y}_n} : (\mathbb{Z}_q^m)^n \to \mathbb{Z}_q \mid \mathbf{y}_i \in \mathbb{Z}_q^m\}$. Whenever we write $f_{\mathbf{y}} \in \mathcal{F}_{q,n}^m$, we assume that $\mathbf{y}$ can be divided into n equal vectors of size m.

2.2   Differential Privacy

DP ensures that changing or removing a single individual from a private dataset does not alter the output of an evaluation over this dataset by more than a negligible margin. The overall result should remain approximately the same regardless of whether an individual participates in the evaluation or not. Hence, their private attributes remain protected, but it is still possible to infer some information from the dataset. One important property of DP is its robustness under post-processing [20]. In other words, if the output of a DP mechanism is further processed by another function, the result still fulfills DP.

There are two main forms of DP, LDP and GDP. Since the latter is more common and assumed as the classical DP, *global* is often omitted when it is clear from the context. Formally, we define LDP as follows:

Definition 1 (Local Differential Privacy [38]). *A randomized mechanism* $f : \mathcal{X} \to \mathcal{Y}$ *is ϵ-local differentially private if for all $S \subseteq Range(f)$ and for all* $x, x' \in \mathcal{X}$:

$$\Pr[f(x) \in S] \leq \exp(\epsilon) \Pr[f(x') \in S],$$

where the probability space is over the coin flips of the mechanism f.

In LDP each record is perturbed directly *before* the analysis, whereas in GDP it is perturbed *during* or *after* the analysis. We define GDP as follows. Denote by $X \simeq X'$ that two databases $X, X' \in \mathcal{X}$ are adjacent, i.e., differ in only one entry.

Definition 2 (Global Differential Privacy). *A randomized mechanism f :* $\mathcal{X} \to \mathcal{Y}$ *is (ϵ, δ)-differentially private if for all $S \subseteq Range(f)$ and for all adjacent inputs $X \simeq X'$:*

$$\Pr[f(X) \in S] \leq \exp(\epsilon) \Pr[f(X') \in S] + \delta,$$

where the probability space is over the coin flips of the mechanism f. If $\delta = 0$, we say that f is ϵ-differential private.

Typically, in the case of GDP, there is a curator, which has hold of the data X, and an analyst, who wants to evaluate some function f. The curator returns a perturbed evaluation of the function, i.e., $\widetilde{f}(X) = f(X) + \nu$. This noise ν is sampled from a distribution whose parameters are dependent on the leakage of f and the so called privacy budget of the individuals. As the analyst might query more than one function, it is important that the authority keeps track of the privacy budget and does not answer any more queries once it is consumed. More precisely, if the analyst queries n functions f_i, $i \in [n]$ which are (ϵ_i, δ_i) private, then the consumed privacy budget is $(\sum_{i \in [n]} \epsilon_i, \sum_{i \in [n]} \delta_i)$.

In order to set the parameters of output perturbing distributions, we make use of the l_2-sensitivity. This is a measure of how much one individual, i.e., one record, influences the output of a function.

Definition 3 (l_2-sensitivity). *The l_2-sensitivity of a function $f : \mathcal{X} \to \mathcal{Y}$ is* $\Delta_2(f) = \max_{X, X' \in \mathcal{X}, X \simeq X'} ||f(X) - f(X')||_2$.

Balle et al. [10] introduced the so called analytic Gaussian mechanism, that is mostly used in the ML context.

Definition 4. *Let $f : \mathcal{X} \to \mathbb{R}^d$ be a function and $\Delta_2(f)$ its sensitivity. Let Z be a d-dimensional, centered, independent Gaussian random variable with variance σ^2 and $u = \frac{\Delta_2(f)}{\sigma}$. For any $\epsilon \geq 0$ and $\delta \in [0, 1]$, the analytic Gaussian output perturbation mechanism $\widetilde{f}(x) = f(x) + Z$ is (ϵ, δ)-DP if and only if*

$$\Phi\left(\frac{u}{2} - \frac{\epsilon}{u}\right) - \exp(\epsilon)\Phi\left(-\frac{u}{2} - \frac{\epsilon}{u}\right) \leq \delta, \tag{1}$$

where Φ denotes the Gaussian cumulative distribution function.

Let $\mathsf{GM}(\epsilon, \delta, \Delta_2(f)) \to \mathbb{D}$ be the algorithm that on input of the privacy parameters and the sensitivity outputs a d-dimensional, centered, independent Gaussian distribution whose parameter fulfill (1). GM can be implemented efficiently using binary search.

2.3   Private Gradient Descent

In order to train a ML model, or more precisely its parameters $\boldsymbol{\theta}$, one common approach is to use GD [2], which is an iteration based optimization algorithm. The goal is to minimize the loss function $L(\boldsymbol{\theta}, \cdot)$ of the model. In private GD, additional privacy measures are applied such as the addition of DP noise.

Let $X = \{(\mathbf{x}_i)\}_{i \in [n]}$ be a dataset with m attributes, where $\mathbf{x}[0] := 1$ is a constant, $(\mathbf{x}_i[1], \ldots, \mathbf{x}_i[m])$ the record and $\mathbf{x}[m+1] \in \{0,1\}$ the dependent variable. Moreover, let α be the learning rate and ∇L the gradient of L. Let $F_{\boldsymbol{\theta}}(X) := \frac{\alpha}{n} \sum_{i \in [n]} \nabla L(\boldsymbol{\theta}, (\mathbf{x}_i))$. In traditional GD, the parameters are updated in each iteration t by

$$\boldsymbol{\theta}^{(t+1)} := \boldsymbol{\theta}^{(t)} + F_{\boldsymbol{\theta}}^{(t)}(X). \tag{2}$$

In contrast, in private GD, the parameters are adjusted by applying additional privacy-preserving noise, i.e.,

$$\boldsymbol{\theta}^{(t+1)} := \boldsymbol{\theta}^{(t)} + F_{\boldsymbol{\theta}}^{(t)}(X) + \boldsymbol{\nu}^{(t)}, \tag{3}$$

where $\boldsymbol{\nu}^{(t)}$ is sampled according to a DP distribution.

3   Approaches to Privacy-Preserving Machine Learning

The advantages and disadvantages of using MPC, HE and FE in the context of PPML have been studied in detail in, e.g, [50, 14]. In this section, we expand their comparison to our specific scenario and focus on the applicability of solutions based on LDP, FL, MPC, HE and FE. Recall that our scenario requires that:

 i) The trusted party has limited resources.
 ii) Data holders can join dynamically.
 iii) Data holders do not have to actively participate during the analysis.
 iv) The input and output privacy is ensured.
 v) The analyst decides on when the analysis is performed.

For MPC and HE we can always obtain DP through (one of) the trusted parties, thus we can assume that output privacy is always given in the scenarios.

Local Differential Privacy In LDP, each data holder locally randomizes their data using some mechanism $\mathcal{M}$ such that Definition 1 is satisfied. After that the data is collected and can be stored on a publicly available server, without compromising the privacy. Through the randomization, the data does not leak the plaintexts. A major advantage of LDP is that individuals perturb their data once and do not have to participate any further, independent on the analysis. By the post-processing property of (L)DP, the privacy is protected, no matter how many functions are evaluated on the data. Moreover, there is no need for a trusted party, especially one with demanding resources.

However, while LDP provides stronger individual privacy guarantees and fulfills all of the above requirements, this comes at the cost of severely degraded data utility due to the high level of noise that needs to be added to each individual data point [30, 40].

Federated Learning FL is a well known approach to enable the training of ML models via an iterative GD algorithm over partitioned data, without revealing the plain or pseudonymized data. Usually, FL is used if data holder possess multiple data records. It has been getting an increasing interest not only from academia but also from industry [1, 52, 22].

The core idea is that each data holder trains a local model on their local data and that these models are merged. More precisely, each data holder trains some iterations of the GD on their local data, optimizing their weights. Then, the trained weights are aggregated over all parties and again distributed to resume the local training with aggregated weights.

FL was proposed to preserve privacy during training by sharing no plain data but only the intermediate results after the iterations. As these may still leak too much information, before sharing the local intermediate results, some well chosen noise is added to conceal the concrete result and provide DP.

With FL we can obtain i) and iii)- v). However, as the model is trained locally, this requires the active participation of the data holders during training, which violates iii). Moreover, if we assume a minimal trust model, i.e., as few data holders as possible, each data holder only possesses one record. Hence, the overall amount of noise per local model is significant, almost resembling LDP.

Multi-Party Computation In MPC, a given function is jointly computed by a set of parties, where each parties obtains a so-called share of the input data from the data holders. To obtain, iv), it is important that no party is in possession of all shares. Otherwise, decryption is possible and the plain data is leaked.

While MPC protocols are highly efficient, and the data holders do not need to be involved after providing their data, fulfilling iii), it still has some disadvantages. First of all, it is important that no untrusted party is in possession of all of the shares, as otherwise decryption would be possible. Hence, the security of most MPC protocols either requires the majority of the computing parties to be trusted, or only considers a passive adversary who honestly follows the protocol. If the data is collected over a period of time, as considered in our dynamic use case, each of the computing parties has to potentially store the received shares over a long time. They cannot be collected by the analyst as they would otherwise be able to decrypt. Lastly, there are high communication costs between the parties. Thus, a communication network with strong delivery guarantee is required. In conclusion, it is hard to fulfill conditions i), ii) and v).

Homomorphic Encryption The idea of HE is that one can operate over encrypted data similar than over plain data, i.e. for certain functions f one can also compute f^* over the ciphertexts, where $\mathsf{Dec}(\mathsf{ek}, f^*(\mathsf{ct}_1, \ldots, \mathsf{ct}_n))$ yields $f(\mathbf{x}_1, \ldots, \mathbf{x}_n)$. In particular, HE allows for complex functions without revealing any intermediate result, ensuring iv). However, this comes at the cost of high computational power of the executing party.

As the key used for decryption is independent of the computation applied to the ciphertexts, it should not be in possession of the party that holds the ciphertexts. This leaves two options: First, the analyst is in possession of the

decryption key, which implies that we are in need of a trusted computing party with the necessary resources to evaluate the function on the ciphertexts. The analyst only obtains the final ciphertext, which contains the (possibly perturbed) function evaluation. Second, the authority is in possession of the decryption key and the analyst performs the computation. In this case, additional measures are necessary to verify that the analyst has correctly computed the intended function, e.g., using verifiable HE [47]. However, verifications either require a trusted execution environment or zero-knowledge proofs, which yield an excessive overhead in the complexity of the function. Hence, although ii) and iii) are fulfilled, in both scenarios the trusted party is in need of a lot of resources, contradicting i).

Noisy (Dynamic Multi-Client) Functional Encryption Classical noisy FE [50] allows decryption of a noisy function's evaluation on encrypted data, i.e., any party in possession of a decryption key for a function f can obtain $f(x_1, \ldots, x_n) + \nu$ from the ciphertexts and decryption key and nothing more, ensuring iv). Therefore data holders only need to participate by submitting their encrypted data to the analyst, giving v), while the rest of the communication is between the analyst and authority, providing iii). Moreover, the generation of decryption keys is comparably low effort, fulfilling i).

However, to obtain ii), we have to resort to our new definition of DyNMCFE, which also covers all of the advantages above. Hence, DyNMCFE seems like the perfect fit for our scenario. More precisely, the data holders encrypt their data under the label ℓ using an encryption key from the trusted authority, which they receive upon registration. After they submit their ciphertexts to the analyst, they are not a part of the analysis anymore and only act if they want to join a different analysis. All further communication will then be between the authority and the analyst, who queries functions to receive decryption keys for their analysis.

The only disadvantage is the limitation in the function's complexity. We solve this drawback in Section 5 and provide a protocol how ML models such as linear or logistic regression can be trained with only a linear scheme.

Conclusion The best fits for the described scenario, where conditions i)-v) are fulfilled, are DyNMCFE and LDP. All that is left is the utility of the trained models of the analyst. For this reason, we use LDP as a baseline to compare our utility to in Section 7.

4   Noisy Multi-Client Functional Encryption

In the following we combine the established notions of MCFE [32] and NMIFE [50] and extend it to the new class of (Dy)NMCFE.

As in the case of NMIFE, an NMCFE scheme allows to evaluate *perturbed* n-ary functions f_ℓ on encrypted data, receiving only $f_\ell(\mathbf{x}_{1,\ell}, \cdots, \mathbf{x}_{n,\ell}) + \nu$. Stemming from the notion of MCFE, we introduce labels ℓ to the ciphertext and each decryption key, ensuring that only ciphertexts and decryption keys with the

same label can be combined. This ensures a fine-grained access structure and is especially helpful to achieve DP, where each function evaluation on a different set of data should contain its own perturbation noise. In addition, this allows for different vector lengths of $\mathbf{x}_i$, i.e. different number of attributes, per label, as each function only can be applied to the specific encoded label.

The extension to the dynamic setting is inspired by Chotard et al. [16], where the encryption key generation is separated from the setup phase. This allows us to dynamically register new clients, eliminating the need for a pre-defined number of clients. In an DyNMCFE each function $f_{\ell,S}$ is additionally associated with a set S of clients, i.e., it can only be applied to the ciphertexts of these particular clients. Our definition allows for a polynomial number of clients, which suffices in practice, as the number of data holders is also bounded by, e.g., the people living in a country[1]. A naive way to achieve this with classical NMCFE would be to set the function coefficients for the clients $i \notin S$ to zero, but this would require a pre-defined number of clients, which could be unpractical.

4.1 Definition and Correctness

In the following we give the formal definition of DyNMCFE, where we mark deviations from NMIFE coming from each modification in boxes, i.e., from multi-client and dynamic number of clients respectively. In other words, if we strip away these changes, it gives us the formal definition of NMIFE as in [50].

Definition 5 ((Dynamic) Noisy Multi-Client Functional Encryption).
Let $\mathcal{F}^{m_{\max}}_{\lambda,n_{\max}} = \{\mathcal{F}^{m_\ell}_S\}$ *be a family of* sub-families $\mathcal{F}^{m_\ell}_S$, *parameterized by an index set* $S \subseteq [n_{\max}]$, $n_{\max} = \mathsf{poly}(\lambda)$, *and* $m_{\max} \geq m_\ell \in \mathbb{N}$, *containing functions* $f : (\mathcal{X}^{m_\ell}_\ell)^{|S|} \to \mathcal{Y}_\ell$. *Let* $\mathcal{L} = \{0,1\}^* \cup \{\bot\}$ *be a set of labels. A dynamic noisy multi-client functional encryption scheme (DyNMCFE) of polynomial arity for* $\mathcal{F}^{m_{\max}}_{\lambda,n_{\max}}$ *and* $\mathcal{L}$ *is a tuple of five efficient algorithms* $\mathsf{NMCFE} = (\mathsf{Setup}, \mathsf{EKeyGen}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ *of the following form:*

$\mathsf{Setup}(1^\lambda, m_{\max}, n_{\max})$**:** *Takes as input the security parameter* λ, *the maximum vector length* $m_{\max}$ *and maximum number of clients* $n_{\max}$. *It outputs a set of public parameters* pp *implicitly defining the function family and a master secret key* msk. *All remaining algorithms implicitly take* pp.

$\mathsf{EKeyGen}(\mathsf{msk}, i)$ **:** *Takes as input the master secret key* msk, *an index* $i \in [n_{\max}]$ *and generates a secret encryption key* ek_i. *It sets* $\mathsf{msk} := \mathsf{msk} \cup (i, \mathsf{ek}_i)$ *and returns* ek_i.

$\mathsf{Enc}(\mathsf{ek}_i, x_i, \ell)$**:** *Takes as input the encryption key* ek_i *for a slot* $i \in [n_{\max}]$, *a message* $x_i \in \mathcal{X}^{m_\ell}_\ell$ *and* a label $\ell \in \mathcal{L}$. *It outputs a ciphertext* $\mathsf{ct}_{i,\ell}$.

---

[1] It is important to note that in our concrete scheme presented in Section 6.2, this bound can be chosen arbitrarily large (as long as it remains polynomial in the security parameter), without sacrificing any efficiency.

KeyGen(msk, $S, f, \ell, \mathbb{D}$): *Takes as input the master secret key* msk, *a function* f *of arity* $|S| \subseteq [n_{\max}]$, *a label* $\ell \in \mathcal{L}$ *and a distribution* $\mathbb{D}$ *over some subset* $\Delta \subseteq \mathcal{Y}_\ell$ *such that* $\Pr[f + \Delta \in \mathcal{F}_S^{m_\ell}] = 1$. *Sample* $\nu \leftarrow \mathbb{D}$ *and output a decryption key* $\mathsf{dk}_{f, \ell, S}$.

Dec($\mathsf{dk}_{f, \ell, S}, \{\mathsf{ct}_{i, \ell}\}_{i \in S}$): *Takes as input the decryption key* $\mathsf{dk}_{f, \ell, S}$ *and* $|S|$ *ciphertexts* $\{\mathsf{ct}_{i,\ell}\}_{i \in S}$, *all encrypted under the same label used for the decryption key. It outputs a value* $z \in \mathcal{Y}_\ell$.

The definition is equivalent to its non-dynamic version if all parameters are fixed and encryption keys are generated during setup. In detail, if $S = [n_{\max}]$ for all keys, $m_\ell = m_{\max}$ for all $\ell \in \mathcal{L}$ being part of the public parameters, and each ek_i is generated during setup, this yields NMCFE.

Remark 1. The knowledgeable reader may notice that in contrast to the definition of classical MCFE, the functions are also tied to the labels. This is necessary to ensure that only decryption keys under the same label as the ciphertexts are able to decrypt these, which is needed for DP to ensure independent noise in each function evaluation. Generally, the label can always be provided to KeyGen, but is ignored inside the algorithm to guarantee that each key can be applied to the ciphertexts independent of their labels.

Correctness The correctness definition for an NMIFE scheme says that if all algorithms have been applied correctly, then the distribution of $\mathsf{Dec}(\mathsf{dk}_f, \mathsf{Enc}(x)) - f(x)$ should be indistinguishable from $\mathbb{D}_f$. This results in obtaining $f(x) + \nu$ with almost the same probability as the probability that ν is drawn. For DyNMCFE, the definition is modified such that decryption keys can only decrypt those ciphertexts which share the same label and are in the corresponding index set.

Formally, we obtain the following definition, which is a straightforward adaptation from [50].

Definition 6 (Correctness of DyNMCFE). *A dynamic NMCFE scheme* $\mathsf{NMCFE} = (\mathsf{Setup}, \mathsf{EKeyGen}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ *is correct if for any security parameter* λ, *maximum number of clients* $n_{\max} = \mathsf{poly}(\lambda)$ *for all* $S \subseteq [n_{\max}]$, $f \in \mathcal{F}_S^{m_\ell}$, $m_{\max} \in \mathbb{N}$, $\ell \in \mathcal{L}, x_i \in \mathcal{X}_\ell^{m_\ell}$ *for all* $i \in |S|$, $m_\ell \leq m_{\max} \in \mathbb{N}$ *and all distributions* $\mathbb{D}_f$ *over some set* $\Delta \subseteq \mathcal{Y}_\ell$ *with* $f + \Delta \in \mathcal{F}_S^{m_\ell}$, *when* $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, m_{\max})$, $\{\mathsf{ek}_i \leftarrow \mathsf{EKeyGen}(\mathsf{msk}, i)\}_{i \in S}$ *and* $\mathsf{ct}_{i,\ell} \leftarrow \mathsf{Enc}(\mathsf{ek}_i, x_i, \ell)$, *it holds:*

$$\Pr\left[\mathsf{Dec}(\mathsf{KeyGen}(\mathsf{msk}, S, f, \ell, \mathbb{D}_f), \{\mathsf{ct}_{i,\ell}\}_{i \in S}) - f(\{x_i\}_{i \in S})\right] \equiv \mathbb{D}_f,$$

where the probability is taken over the random coins of the algorithms of DyNMCFE.

4.2   Security of NMCFE

Informally speaking, an NMCFE scheme is considered secure, if the ciphertexts and decryption keys do not reveal any information about the plaintext data

and the noise for any label, except for the desired information $f_\ell(\{x_{i,\ell}\}_{i \in S}) + \nu$. Similar as in the definition of DyNMCFE (Definition 5), Definition 7 combines the security definitions of NMIFE, MCFE and takes inspiration from [16].

The biggest difference from (N)MIFE to (N)MCFE is the allowance of corruptions, which means that the adversary can corrupt clients and obtain their encryption keys. Moreover, the admissibility conditions must hold with respect to each label ℓ and each queried subset of clients S.

It is necessary that before any decryption key query including clients in S is posed, the clients in S are already registered and have obtained an encryption key. For this reason, we introduce a dummy oracle QEKeyGen, which generates encryption keys. We stress that the adversary does not receive an output from that oracle and hence, does not obtain any other encryption keys other than those they corrupted.

For now, we only consider security for one-time schemes, i.e., schemes, which only permit one ciphertext per label and client. These schemes suffice for our use case as each function evaluation needs its own independent noise for different datasets. We stress that all concurrent noisy FE schemes also only provide one-time security.

In the following, $\mathbb{D}_\nu$ defines the distribution that outputs ν with probability one. Deviations from one-time NMIFE coming from MCFE and the dynamic setting are again marked in the respective boxes. If a whole algorithm or condition exists due to that change, only its name will be marked.

Definition 7 (One-time Security of DyNMCFE). *Consider the DyNMCFE scheme* DyNMCFE= (Setup, EKeyGen, Enc, KeyGen, Dec) *for a maximum plaintext length of* $m_{\max}$, *maximum number of clients* $n_{\max}$ *and label set* $\mathcal{L}$. *For any security parameter* λ, *consider the following* $\mathsf{IND}_\beta^{\mathsf{DyNMCFE}}$ *game between an adversary* $\mathcal{A}$ *and a challenger* $\mathcal{C}$. *The game involves a set* $\mathcal{HS}$ *of honest clients, initialized to* $\mathcal{HS} := [n_{\max}]$, *a set of corrupted clients* $\mathcal{CS}$, *initialized to* $\mathcal{CS} := \emptyset$ *and a set of queried index sets per label* $\mathcal{S}$, *initialized to* $\mathcal{S} := \emptyset$.

Initialization: *At the beginning,* $\mathcal{C}$ *runs* $(\mathsf{pp},) \leftarrow \mathsf{Setup}(1^\lambda, m_{\max}, n_{\max})$. *Then, they choose a random bit* $\beta \leftarrow \{0,1\}$ *and hand* pp *to* $\mathcal{A}$.

Corruption queries: $\mathcal{A}$ *can pose queries to the corruption oracle* $\mathsf{QCor}(i)$ *before any other queries to obtain* $\mathsf{ek}_i \leftarrow \mathsf{EKeyGen}(\mathsf{msk}, i)$. $\mathcal{C}$ *set* $\mathcal{CS} := \mathcal{CS} \cup \{i\}$ *and* $\mathcal{HS} := \mathcal{HS} \setminus \{i\}$. *Any further query to* $\mathsf{QCor}(i)$ *is answered with the same* ek_i.

Encryption key queries: *The adversary* $\mathcal{A}$ *can adaptively pose encryption key queries* $\mathsf{QEKeyGen}(i)$, *for which* $\mathcal{C}$ *generates* $\mathsf{ek}_i \leftarrow \mathsf{EKeyGen}(\mathsf{msk}, i)$ *and updates* $\mathsf{msk} := \mathsf{msk} \cup \{\mathsf{ek}_i\}$. *For any given* $i \in [n_{\max}]$, *only one query is allowed and any subsequent query for* i *is ignored. Nothing is returned to* $\mathcal{A}$.

Encryption queries: *The adversary* $\mathcal{A}$ *can adaptively transmit encryption queries* $\mathsf{QEnc}(i, x_i^0, x_i^1, \boxed{\ell})$. *If either* $i \in \mathcal{CS}$ *or* $\mathsf{QEKeyGen}(i)$ *has been previously* called, *they are answered by* $\mathsf{ct}_{i,\ell} \leftarrow \mathsf{Enc}(\mathsf{ek}_i, x_i^\beta, \ell)$ *and ignored otherwise. For any given pair* (i, ℓ), *only one query with return value is allowed and any subsequent queries involving the same* (i, ℓ) *are ignored.*

Decryption key queries: $\mathcal{A}$ *can adaptively pose queries* $\mathsf{QKeyGen}(\boxed{S}, f, \boxed{\ell,}$
$\nu^0, \nu^1)$ *to receive functional decryption keys. If for all* $i \in S$, $\boxed{i \in \mathcal{CS} \text{ or}}$
$\boxed{\mathsf{QEKeyGen}(i)}$ *has been called,* $\mathcal{C}$ *returns* $\mathsf{dk}_{f,\ell,S} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, S, f, \ell, \mathbb{D}_{\nu^\beta})$
and $\boxed{\text{updates } \mathcal{S} := \mathcal{S} \cup (\ell, S).}$ *Otherwise, the request is ignored.*

Finalize: $\mathcal{A}$ *outputs a bit* $\beta' \in \{0, 1\}$. $\mathcal{C}$ *checks, if* $\mathcal{A}$ *acted admissibly. If not,* $\mathcal{C}$
sets $\beta' = 0$. $\mathcal{A}$ *wins, if* $\beta' = \beta$.

Depending on the specification of the game defined in advance, we call $\mathcal{A}$
admissible, if all of the following conditions hold.

$\boxed{i)}$ *If* $i \in \mathcal{CS}$, *then for any query* $\mathsf{QEnc}(i, x_i^0, x_i^1, \ell)$, $x_i^0 = x_i^1$, *i.e.,* $\mathcal{A}$ *cannot trivially distinguish both cases by creating their own ciphertexts.*

ii) *For all* $(\boxed{\ell,} \boxed{S}) \in \mathcal{S}$, $\mathsf{QEnc}(i, \cdot, \cdot, \ell)$ *has been queried for all* $i \in \mathcal{HS} \cap S$, *i.e., for each queried function, decryption should be possible.*

iii) *For any label* $\ell \in \mathcal{L}$, *any tuple* $(\ell, S) \in \mathcal{S}$, *any family containing all honest encryption queries with respect to* S, *i.e.,* $\{\mathsf{QEnc}(i, x_i^0, x_i^1, \boxed{\ell})\}_{i \in \boxed{\mathcal{HS} \cap S}}$, *for any family of inputs* $\{x_i \in \mathcal{X}_\ell^{m_\ell}\}_{i \in \mathcal{CS}}$, *any query* $\mathsf{QKeyGen}(\boxed{S}, f, \boxed{\ell,} \nu^0, \nu^1)$, *we require that:*

$$f(\{x_i^0\}_{\boxed{i \in S}}) + \nu^0 = f(\{x_i^1\}_{\boxed{i \in S}}) + \nu^1, \tag{4}$$

i.e., no function evaluation can yield distinction.

The advantage of $\mathcal{A}$ *in this game is defined as*

$$\begin{aligned}
\mathsf{Adv}_{\mathsf{DyNMCFE}, \mathcal{A}}^{\mathsf{IND}}(\lambda) = \Big| &\Pr(\mathsf{IND}_0^{\mathsf{DyNMCFE}}(\lambda, \mathcal{A}) = 1) \\
&- \Pr(\mathsf{IND}_1^{\mathsf{DyNMCFE}}(\lambda, \mathcal{A}) = 1) \Big|.
\end{aligned}$$

A DyNMCFE scheme $\mathsf{DyNMCFE}$ *provides* IND*-security, if* $\mathsf{Adv}_{\mathsf{DyNMCFE}, \mathcal{A}}^{\mathsf{IND}}(\lambda) \leq \mathsf{negl}(\lambda)$.

In particular, conditions i) - iii) are commonly used to prevent trivial attacks through either the possession of encryption keys or through varying function evaluations. Note that ii) is required such that Equation (4) always has to hold, as soon as there is a function query including at least one honest client. These restrictions are common in FE and are simply modified to fit our use of labels and subsets of clients.

In general, the security definition can be further generalized to allow different modifications. such as adaptive corruptions, i.e., enabling corruptions at any point in time, allowing more than one ciphertext per slot per label or requiring that $\nu^0 = \nu^1$, which would yield regular MCFE. Such a definition is included in Appendix A for a non-dynamic NMCFE, as it facilitates our proofs. Note an DyNMCFE scheme which satisfying Definition 7 suffices for our use case and any modification, except for the adaptive corruptions, does not give us any benefit.

5   Privacy-Preserving Protocol

In the scenario described in Section 1, we assume that the data is distributed across multiple data sources. Via a central trusted entity, analysts and data holders can be connected and the analysis can be performed. In the following, we propose a protocol for PPML based on DyNMCFE. To prove the efficiency of the proposed protocol, we train a logistic regression model on medical data in Section 7 using DyNo (Section 6). Our generic approach is not limited to logistic regression but allows for linear or linearly approximated ML models, which are trained using an iteration-based algorithm, e.g., private GD. To describe our protocol, we first examine the parties involved.

Parties We consider three different parties.

Authority: The authority is a trusted party with limited resources who handles the setup of the DyNMCFE scheme. They provide encryption keys to the data holders and decryption keys to the analyst. This could be for example a hospital or an ethical committee that oversees such studies. On request of a function f, they answer truthfully with a perturbed decryption key dk_f, where the noise ν is sampled according to a DP providing distribution. They keep track of the privacy budget, ensuring that DP is always achieved.

Data Holder: We assume the data is distributed between several parties, e.g., the patients themselves, where each party is in possession an entire record. They are willing to submit their individual data to research honestly (possibly at different points in time). On the one hand, they still want to maintain their privacy and on the other hand, do not wish to actively participate in the process themselves.

Analysts: The analysts want to learn information from the research data, e.g., by training some ML model. They can behave maliciously in the sense that they can corrupt data holders, i.e., work with them to obtain their data in plain or even their secret keys, and request any functions supported by the functionality of DyNMCFE. As the sampled noise directly depends on the queried function and not on the training of the ML model, the privacy of the data holders remains protected.

Design Considerations Although there is extensive research in the area of FE, efficient schemes are still restricted to support only linear or quadratic functions. For efficiency, we focus on linear schemes. In fact, a linear scheme suffices for our use case for the following reason. Remember that in private GD, we want to compute $F_{\boldsymbol{\theta}}(X)$ to update our gradients, which is the sum over all gradients $\nabla L(\boldsymbol{\theta}, \mathbf{x}_i)$, i.e., $F_{\boldsymbol{\theta}}(X) := \frac{\alpha}{n} \sum_{i \in [n]} \nabla L(\boldsymbol{\theta}, (\mathbf{x}_i))$. Particularly, each ∇L only depends on the record of one individual. Hence, if ∇L can be approximated as a polynomial, we can precompute its monomials to obtain an extended plaintext $\widetilde{\mathbf{x}}_i$. As each monomial now has its own slot, ∇L can be computed as a scalar product between $\widetilde{\mathbf{x}}_i$ and its respective coefficients, i.e., it can be seen as a linear function. To ensure that only the necessary monomials are determined and the

coefficients are rightfully multiplied with the monomials, we can identify this transformation with a mapping $M : \mathcal{X} \to \widetilde{\mathcal{X}}$. This way, $F_{\boldsymbol{\theta}}(X)$ can be correctly evaluated by a linear DyNMCFE scheme.

M influence the size of the plaintext and hence of the ciphertext. Thus, they have to be known before encryption to verify that the scheme supports the individual plaintexts of that particular size. In our scheme, this should generally not be a problem as the maximum bit-length is rather large. Moreover, it is desirable that the overall plaintext size does not grow too much as this directly influences the overall runtime. This will be further discussed in Section 7.

Protocol Given the three parties and the linearized private GD algorithm, our PPML protocol is as follows.

The authority sets up the environment for the DyNMCFE scheme with a maximum plaintext bit-length $l_{\max}$ and a label space $\mathcal{L}^2$.

Whenever a data holder joins the system, they exchange their privacy budget (ϵ_i, δ_i) for the individual encryption key.

Whenever an analyst wants to start a new analysis on data that has not been provided yet, they send a request to the authority about what data they need and the form of the data, e.g., some normalization factors, realized by M. If the required data is within the limits of the scheme, the authority approves the study and assigns the analysis an unused label $\ell \in \mathcal{L}$. The request is forwarded together with all necessary information to the data holders.

Each data holder can decide, whether they want to participate and tell the authority their decision. If they do, they encrypt their data according to the requirements and submit it to a server which can be accessed by the analyst. Although they may submit different data under several labels, possibly over time, they are not involved further in the evaluation process. It is also possible that a data holder registers at this point in time and participates in the particular study. This exchange is displayed in Figure 1.

To train an ML model on the gathered ciphertexts using GD, teh analyst first initialize the weights of the model, $\theta^{(0)}$. In each iteration t, they pose queries including the requested functions according to M and the amount of privacy budget in said iteration, i.e., $(F_{\boldsymbol{\theta}^{(t)}} = \{F_{\boldsymbol{\theta}^{(t)}}[j]\}_{j \in [0;m]}, (\epsilon^{(t)}, \delta^{(t)}))$. The authority checks if this would exceed any participant's remaining budget. If so, they contact the analyst to remove the particular individual's in the next decryption key queries. Otherwise, they update each participant's privacy budget according to the consumption of privacy and compute $\mathbb{D} \leftarrow \mathsf{GM}(\epsilon^{(t)}, \delta^{(t)}, \Delta_2(F_{\boldsymbol{\theta}}^{(t)}))$, which provides DP according to Definition 4. They generate the functional decryption keys with respect to $\mathbb{D}$ and return them to the analyst.

After each iteration, the analyst updates their model parameters $\boldsymbol{\theta}$ and starts the next iteration by again requesting functional decryption keys. The process ends after a fixed number of iterations T or if the privacy budget of all data

---

[2] Technically, we also require a maximum number of clients $n_{\max}$. However, as $n_{\max} = \mathsf{poly}(\lambda)$, we can set it to 2^{34}, which is more than twice of current world population. Hence, this restriction is ignored in the protocol.
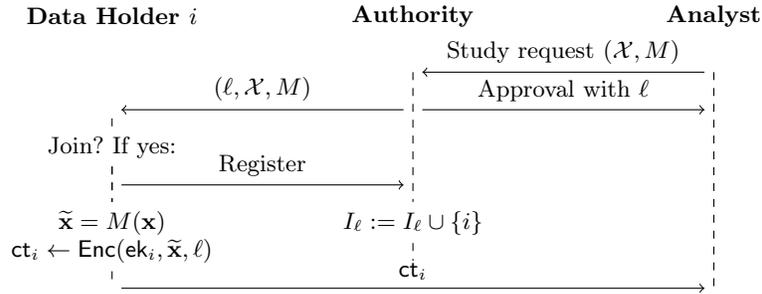
Fig. 1. Analysis request and data gathering.

holders is exceeded. Figure 2 shows the training phase if none of the data holders' privacy budgets are depleted before iteration T.

Note that all of the decryption keys can be made public for other analysts. Further, another analyst may conduct a training as in Figure 2 for a different model, depending on the remaining privacy budget.

Threat Model and Privacy Analysis We require that the authority honestly generates the decryption keys and takes care of the privacy budget. As the authority tracks the privacy budget of the data holders over all labels and function queries, DP and hence output privacy is always guaranteed. Note that at least one trusted party is always required, especially in protocols using HE or MPC. LDP would be the only approach without a trusted party, but distorts the utility of the trained model, as we show in Section 7.

The analyst on the other hand may collude with data holders. More precisely, they may know their data and even their particular encryption keys. The privacy of the non-colluding data holders is still guaranteed due to DP and the allowance of corruptions of the DyNMCFE scheme.

Moreover, the analyst is not limited to functions which help them train a ML model. However, due to the assurance of DP, the analyst would only jeopardize the usefulness of their model by asking for other decryption keys.

Overall, by employing this approach, both input and output privacy of all individuals are ensured.

6   DyNo: A concrete DyNMCFE instantiation

In this section, we present our one-time DyNMCFE scheme, DyNo. The main idea of DyNo is similar to the one-time MIFE scheme of Abdalla et al. [3], which makes use of the linearity of the one-time pad. We extend their scheme to support noise-hiding, i.e., to be a secure NMIFE scheme. Let us first recall their scheme and our extension to noise-hiding, before we present DyNo.
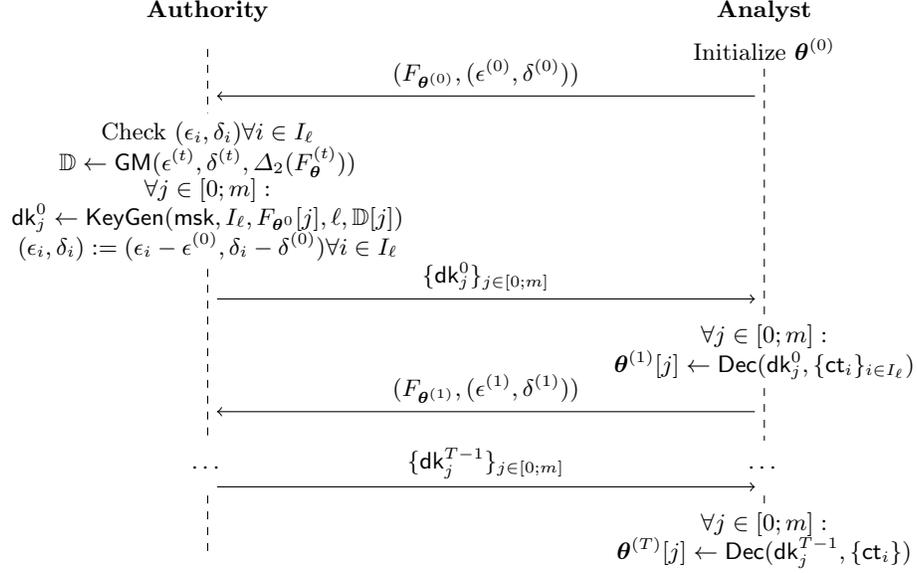
Fig. 2. Training phase.

6.1 Warm up: NMIFE$^{\text{ot}}$

In the following we refer to the scheme of Abdalla et al. [3] as MIFE$^{\text{ot}}$. Extensions implemented to lift MIFE$^{\text{ot}}$ to a noisy variant NMIFE$^{\text{ot}}$ are marked in boxes. If $\mathbb{D}$ is chosen to be the all-zero distribution, both schemes are equivalent.

Construction 1. (NMIFE$^{\text{ot}}$**)** *Let $\mathcal{F}_{q,n}^m$ be the class of multi-input inner products over $\mathbb{Z}_q$.* NMIFE$^{\text{ot}}$ *scheme for $\mathcal{F}_{q,n}^m$ consists of the following algorithms:*

Setup$^{\text{ot}}(1^\lambda, m, n)$**:** *On input of the security parameter λ, the vector length m and number of clients n, set $\mathsf{msk} := \emptyset$. For all $i \in [n]$, run $\mathsf{ek}_i \leftarrow$ EKeyGen$^{\text{ot}}(\mathsf{msk}, i)$ and return msk.*

EKeyGen$^{\text{ot}}(\mathsf{msk}, i)$**:** *On input of the master secret key msk, sample $\mathsf{ek}_i \xleftarrow{\$} \mathbb{Z}_q^m$. Set $\mathsf{msk} := \mathsf{msk} \cup \{(i, \mathsf{ek}_i)\}$ and return ek_i.*

Enc$^{\text{ot}}(\mathsf{ek}_i, \mathbf{x}_i)$**:** *On input of the encryption key ek_i and message $\mathbf{x}_i \in \mathbb{Z}_q^m$ for slot $i \in [n]$, return $\mathsf{ct}_i := (i, \mathbf{c}_i := \mathbf{x}_i + \mathsf{ek}_i \mod q)$.*

KeyGen$^{\text{ot}}(\mathsf{msk}, f_{\mathbf{y}}, \boxed{\mathbb{D}})$**:** *On input of the master secret key $\mathsf{msk} = \{(i, \mathsf{ek}_i)\}_{i \in [n]}$, a function $f_{\mathbf{y}}$ defined through $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n)$ and a $\boxed{\text{distribution } \mathbb{D} \text{ over}}$ $\boxed{\mathbb{Z}_q, \text{ sample } \nu \leftarrow \mathbb{D}.}$ Return $\mathsf{dk}_{\mathbf{y}} = (\mathbf{y}, z := \sum_{i \in [n]} \langle \mathsf{ek}_i, \mathbf{y}_i \rangle \boxed{-\nu} \mod q)$.*

Dec$^{\text{ot}}(\mathsf{dk}_{\mathbf{y}}, \mathsf{ct}_1, \ldots \mathsf{ct}_n)$**:** *On input of the decryption key $\mathsf{dk}_{\mathbf{y}} = ((\mathbf{y}_1, \ldots, \mathbf{y}_n), z)$ and ciphertexts $\mathsf{ct}_i = (i, \mathbf{c}_i)$, $i \in [n]$, return $\sum_{i \in [n]} \langle \mathbf{c}_i, \mathbf{y}_i \rangle - z \mod q$.*

Correctness The correctness of the scheme in $\mathbb{Z}_q$ follows directly by the way z is chosen and that $\sum_{i \in [n]} \langle \mathbf{c}_i, \mathbf{y}_i \rangle = \sum_{i \in [n]} \langle \mathbf{x}_i + \mathsf{ek}_i, \mathbf{y}_i \rangle$. Thus, the result after decryption is $f(\mathbf{x}_1, \ldots, \mathbf{x}_n) + \nu$, whose distribution is solely determined by $\mathbb{D}_f$.

Remark 2. For correctness in $\mathbb{Z}$, the modulus q has to be chosen large enough. In particular, assume $\max_{j \in [m]} \mathbf{x}_i[j] < X$ for all $i \in [n]$, $\max_{j \in [mn]} \mathbf{y}[j] < Y$ and $\Pr(\nu > d) \leq \mathtt{negl}(\lambda)$, where $\nu \leftarrow \mathbb{D}_f$, for all queried f. Then $\mathsf{NMIFE}^{\mathrm{ot}}$ is correct in $\mathbb{Z}$, if

$$nmXY + d < q. \tag{5}$$

Security For our security proof, we use the information theoretical security of $\mathsf{MIFE}^{\mathrm{ot}}$ as described above. In a sequence of games we transform $\mathsf{NMIFE}^{\mathrm{ot}}$ to $\mathsf{MIFE}^{\mathrm{ot}}$, which concludes the following theorem. Its proof can be found in Appendix B.1.

Theorem 1. *The* $\mathsf{NMIFE}^{\mathrm{ot}}$ *scheme presented in Construction 1 is* IND-*secure with* $\mathsf{Adv}^{\mathsf{IND}}_{\mathsf{NMIFE}^{\mathrm{ot}}, \mathcal{A}}(\lambda) = 0$ *for any* PPT *adversary* $\mathcal{A}$.

6.2   DyNo: An efficient DyNMCFE scheme

Recall that the difference between NMIFE and DyNMCFE is the use of labels and the ad-hoc registration of clients. Thus, we need the encryption keys ek_i to be dependent on the label. As we additionally require that the keys are labeled as well, we can simply replace the one-time pad with a PRF. This PRF takes as input the encryption key ek_i of client i and the label ℓ. If ek_i is sampled uniformly at random, this function is computationally indistinguishable from a truly random function. A formal definition is given in Definition 11.

For any PRF PRF of output length m, denote by $\mathsf{PRF}_{m'}$ the PRF which only outputs the first $m' \leq m$ values. This transformation is straightforward by cutting for example the last $m - m'$ values.

Construction 2. (DyNo)   *Let* $\{\mathcal{F}^{m_{\max}}_{q,S}\}_{S \subseteq [n_{\max}]}$ *be the family of sub-families of multi-input inner-products over* $\mathbb{Z}_q$ *with maximum attributes* $m_{\max}$ *and maximum number of clients* $n_{\max}$. *Let* $\mathcal{L}$ *be the supported label space and* $\mathsf{PRF} : \{0,1\}^\lambda \times \mathcal{L} \to \mathbb{Z}_q^{m_{\max}}$. *The* DyNo *scheme for* $\{\mathcal{F}^{m_{\max}}_{q,S}\}_{S \subseteq [n_{\max}]}$ *consists of the following algorithms:*

$\mathsf{Setup}(1^\lambda, m_{\max}, n_{\max})$**:** *On input of the security parameter* λ, *maximum vector length* $m_{\max}$ *and maximum number of clients* $n_{\max}$, *set* $\mathsf{msk} := \emptyset$. *Output* msk *and the public parameters* $\mathsf{pp} = (m_{\max}, n_{\max}, \mathcal{L}, \mathsf{PRF})$.

$\mathsf{EKeyGen}(\mathsf{msk}, i)$**:** *On input of the master secret key and slot* $i \in [n_{\max}]$, *sample* $\mathsf{ek}_i \xleftarrow{\$} \{0,1\}^\lambda$. *Update* $\mathsf{msk} := \mathsf{msk} \cup \{(i, \mathsf{ek}_i)\}$ *and return* ek_i.

$\mathsf{Enc}(\mathsf{ek}_i, \mathbf{x}_i, \ell)$**:** *On input of the secret key* ek_i, *a plaintext* $\mathbf{x}_i \in \mathbb{Z}_q^m$ *with* $m \leq m_{\max}$, *for slot* $i \in [n_{\max}]$ *and a label* $\ell \in \mathcal{L}$, *calculate* $\zeta = \mathsf{PRF}_m(\mathsf{ek}_i, \ell)$ *and return* $\mathsf{ct}_{i,\ell} = (i, \ell, \mathbf{c}_i := \mathbf{x}_i + \zeta \mod q)$.

$\mathsf{KeyGen}(\mathsf{msk}, S, f_{\mathbf{y}}, \ell, \mathbb{D})$**:** *On input of the master secret key* msk*, a set of indices* $S \subseteq [n_{\max}]$*, a function* $f_{\mathbf{y}}$*, a label* $\ell \in \mathcal{L}$ *and a noise distribution* $\mathbb{D}$*, if there exists* $(i, \cdot) \notin \mathsf{msk}$ *for some* $i \in S$*, return* $\perp$*. Otherwise, sample* $\nu \leftarrow \mathbb{D}$ *and calculate* $\zeta_i = \mathsf{PRF}(\mathsf{ek}_i, \ell)$ *for all* $i \in S$*. Return* $\mathsf{dk}_{\mathbf{y},\ell,S} = (\ell, \mathbf{y}, z := \sum_{i \in S} \langle \zeta_i, \mathbf{y}_i \rangle - \nu \mod q)$*.*

$\mathsf{Dec}(\mathsf{dk}_{\mathbf{y},\ell,S}, \{\mathsf{ct}_{i,\ell}\}_{i \in S})$**:** *On input of the decryption key* $\mathsf{dk}_{\mathbf{y},\ell,S} = (\ell, \{\mathbf{y}_i\}_{i \in S}, z)$ *and ciphertexts* $\{\mathsf{ct}_{i,\ell} = (i, \ell, \mathbf{c}_i)\}_{i \in S}$*, return* $\sum_{i \in [S]} \langle \mathbf{c}_i, \mathbf{y}_i \rangle - z \mod q$*.*

Note that the construction can serve as an alternative to $\mathsf{NMIFE}^{\mathsf{ot}}$ for a single label if we aim to minimize the size of the encryption keys and set $n_{\max} = n, m_{\max} = m$. This is especially of interest if the data to be encrypted is large. The size of a ciphertext is $m \log q$ bits, both in $\mathsf{NMIFE}^{\mathsf{ot}}$ and DyNo, which matches the size of a encryption key in $\mathsf{NMIFE}^{\mathsf{ot}}$, whereas the encryption key in DyNo is of size λ. Thus, if $m \log q > \lambda$, there is a space savings, though this comes with increased computational costs from the PRF. In practical applications, we find that q tends to grow rapidly, which enhances the effectiveness of DyNo. There are two primary reasons for this. Firstly, we require correctness in $\mathbb{Z}$ rather than $\mathbb{Z}_q$, necessitating that q be sufficiently large, specifically $q > nmXY + d$, where X, Y, and d provide bounds on the ciphertexts, functions, and noise, respectively. Secondly, to convert real-world data in $\mathbb{R}$ to data in $\mathbb{Z}$, a common technique is to employ fixed-point arithmetic using a scaling, as elaborated in Section 5. As X and Y may need to be quite large to achieve the desired precision this directly effects q and therefore the key sizes for $\mathsf{NMIFE}^{\mathsf{ot}}$.

Remark 3. In a similar manner as Abdalla et al. [4], DyNo (respectively, $\mathsf{NMIFE}^{\mathsf{ot}}$) can be utilized to transform any single input IPFE scheme, fulfilling certain properties, to a secure label-keyed NMCFE (respectively, NMIFE) scheme, which allows to encrypt multiple messages per client per label. The only difference we have to make is to exchange $\mathsf{MIFE}^{\mathsf{ot}}$ with DyNo (respectively, $\mathsf{NMIFE}^{\mathsf{ot}}$).

Most inner-product scheme based on lattices and bilinear groups fulfill the required properties, e.g., [6, 39]. This gives us an even broader family of noise-hiding functions which do not rely on function-hiding and can be build from lattices. As we are only interested in one-time schemes for DP, we will not go further into this topic, but leave it as an interesting observation.

Correctness Similar as in the case of $\mathsf{NMIFE}^{\mathsf{ot}}$, one can directly see that the scheme is correct in $\mathbb{Z}_q$, if the PRF in use is deterministic. Correctness in $\mathbb{Z}$ follows if Equation (5) is fulfilled.

Security The security of DyNo relies on the security of the underlying PRF as well as $\mathsf{NMIFE}^{\mathsf{ot}}$. We focus on deterministic PRFs, as these are more efficient and more common in practice.

Theorem 2. *If* $\mathsf{NMIFE}^{\mathsf{ot}}$ *from Construction 1 is* IND*-secure and* PRF *is secure, then* DyNo *from Construction 2 is* IND*-secure. In particular, for any* PPT *adversary* $\mathcal{A}$*, there exist* PPT *adversaries* $\mathcal{B}_1$ *and* $\mathcal{B}_2$ *such that*

$$\mathsf{Adv}^{\mathsf{IND}}_{\mathsf{DyNo},\mathcal{A}}(\lambda) \leq 2n_{\max}q_\ell \cdot \mathsf{Adv}_{\mathsf{PRF},\mathcal{B}_1}(\lambda) + q_\ell \cdot \mathsf{Adv}^{\mathsf{IND}}_{\mathsf{NMIFE}^{\mathsf{ot}},\mathcal{B}_2}(\lambda),$$

where q_ℓ denotes the number of distinct labels queried to QEnc *and* QKeyGen.

The theorem can be proven by straightforward reduction to the non-dynamic variant, where all secret keys are sampled during the setup phase. On the other hand, for the non-dynamic variant, we show that it suffices to show security for one label. From there on, we only have to rely on the security of the PRF, i.e., PRF can only be distinguish from a truly random function with negligible advantage. At this point, an adversary against NMIFE$^{\text{ot}}$ can perfectly simulate the view of an attacker against the modified scheme. Since NMIFE$^{\text{ot}}$ is information theoretically secure, the IND-security of DyNo follows. The whole proof can be found in Appendix B.2.

7   Implementation

We provide an implementation of our DyNMCFE scheme DyNo employed in Section 6.2. All experiments were performed on a system running Ubuntu 22.04.2 LTS, 256GB RAM and 18 vCPUs (AMD Epyc 7272).

As real world data usually takes values in $\mathbb{R}$ instead of $\mathbb{Z}_q$, we make use of fix-point arithmetic. More precisely, a value $x \in \mathbb{R}$ is converted to $\lfloor x \cdot s \rceil$ for an appropriate scaling factor s and an accordingly chosen q. This scaling factor determines for example the level of precision. Choosing it large enough is especially important in the presence of DP as the noise also has to be in $\mathbb{Z}_q$. Normally, this value is rather small. However, depending on the rounding of the noise, the added noise may be larger than necessary for DP, which reduces the utility of the model. Note that scaling or any normalization process can be considered as part of M and do not influence the overall protocol presented in Section 5.

In order to achieve DP, we use the analytic Gaussian mechanism, as displayed in Definition 4, which is included in the Google Go library [46]. Their implementation of the analytic gaussian mechanism employs binomial random variables and appropriate rounding to argue for DP on integers.

As our PRF, on which the security of DyNo is based on, we choose AES-256 as implemented in the Go crypto library [7]. This is a standard protocol recommended by NIST [45] and plausibly post-quantum secure [42].

Moreover, we consider the non-dynamic variant for comparison, i.e., the encryption keys are generated as part of the setup and all clients part of the analysis. The overall runtime and communication costs are almost the same for the dynamic version, except that there may be data holders which never participate in any analysis, but query an encryption key. In addition, we did not include the request for the analysis and the participation check, as both are implicitly present in any protocol, including the non-dynamic NMCFE scheme. However, these should be negligible.

7.1   Benchmarks

We compare the runtime of DyNo with the schemes of Zalonis et al. [50], DiffPIPE, and Escobar et al. [21], RIPFE. These are currently the only (noisy) MIFE schemes providing inner-product functionality and supporting DP.

   Let $N = nm$ be the size of the dataset. Table 2 shows the runtime of all three schemes on the same datasets for different dataset sizes N, where $\mathbf{x}_i \in [0, 2^{16}]^m$ for all $i \in [n]$ and $\mathbf{y} \in [0, 2^7]^N$ were sampled randomly. RIPFE only supports one client, i.e., $n = 1$, so we set $m = N$. Since DiffPIPE is pairing based, it becomes very slow for big m. DiffPIPE is originally a multi-input scheme, therefore to better compare runtimes, we set $n = m = \sqrt{N}$. We evaluate our scheme for both partitions of the dataset and achieve better runtimes in all algorithms. Even for larger datasets of size 10^6, we are still in the range of seconds and milliseconds, respectively. While running DiffPIPE with $N = 10^6$, we needed to cancel the setup phase on our server after several days as the RAM was exhausted.

Table 2. Runtime comparison of all FE schemes supporting DP. Since DiffPIPE did not terminate after several days for $N = 10^6$, we are not able to present runtimes.

N	n	m	Scheme	Setup	Enc	KeyGen	Dec
	1	10^2	RIPFE [21]	3.50 s	0.25 s	1.89 ms	0.14 s
10^2	1	10^2	DyNo	0.01 ms	0.12 ms	0.14 ms	0.01 ms
	10	10		0.01 ms	0.09 ms	0.11 ms	0.02 ms
	10	10	DiffPIPE [50]	37.25 ms	5.55 ms	77.12 ms	1.06 s
	1	10^4	RIPFE [21]	3.51 s	24.48 s	17.83 ms	1.51 s
10^4	1	10^4	DyNo	0.01 ms	4.23 ms	3.13 ms	1.31 ms
	10^2	10^2		0.06 ms	2.27 ms	0.77 ms	1.28 ms
	10^2	10^2	DiffPIPE [50]	57.32 s	0.23 s	5.64 s	25.20 s
	1	10^6	RIPFE [21]	3.59 s	41.03 min	2.23 s	33.09 s
10^6	1	10^6	DyNo	0.019 ms	0.31 s	0.24 s	75.41 ms
	10^3	10^3		0.47 ms	50.93 ms	30.48 ms	85.45 ms
	10^3	10^3	DiffPIPE [50]	-	-	-	-

7.2   Logistic Regression

Besides benchmarking, we also train a logistic regression using DyNo. We only consider the data exchange (including providing the encryption keys) and the training process, as any other costs can be seen as negligible and appear in any other protocol as well. For the sake of simplicity, we assume that all data holders share the same privacy budget.

   As a baseline, we compare the utility to LDP, as this is the only primitive that neither requires the active participation of the data holders nor one or multiple trusted parties with a lot of computational power.

Model Let $X = \{\mathbf{x}_i\}_{i \in [n]}$ be a dataset where each $\mathbf{x}_i \in \mathbb{R}^{m+2}$ includes m real features, a binary label $\mathbf{x}_i[m+1] \in \{0, 1\}$, and a constant $\mathbf{x}_i[0] = 1$. For the sake of readability, we define in the following $z_i := \sum_{j \in [0;m]} \boldsymbol{\theta}[j] \mathbf{x}_i[j]$. The goal of a logistic regression training is to find optimal $\boldsymbol{\theta}^* \in \mathbb{R}^{m+1}$, e.g., using private GD, such that given a fresh $\mathbf{x}$, the classification y can be predicted with high probability. More precisely, it should hold that $\lfloor \sigma(z_i) \rceil = \mathbf{x}_i[m+1]$ with high probability, where $\sigma(x) = \frac{1}{1 + \exp(-x)}$ is the sigmoid function.

Let T be a predefined number of iterations, α the learning rate and $\boldsymbol{\theta}^0 \in \mathbb{R}^{m+1}$ randomly chosen. In a logistic regression, each update of the parameters, i.e., (2), is given as

$$\boldsymbol{\theta}^{(t+1)}[j] := \boldsymbol{\theta}^{(t)}[j] + \frac{\alpha}{n} \sum_{i \in [n]} \left(\mathbf{x}_i[m+1] - \sigma(z_i^{(t)}) \right) \mathbf{x}_i[j]. \tag{6}$$

Inner-product Transformation Since DyNo only supports inner-product functionality we need to adapt the training algorithm. A naive approach would be to view Equation (6) as two nested inner-products. First, $z_i^{(t)}$ is calculated and plugged into σ. Then, $\sigma(z_i^{(t)})$ is regarded as the function vector for $\mathbf{x}_i$ to calculate the outer sum.

However, this approach requires to add noise twice in each iteration to preserve privacy, which is too much noise to ensure the convergence towards optimal model parameter. To reduce the amount of noise, (6) can be calculated directly by the underlying DyNMCFE scheme, if we linearize it.

Conveniently, the sigmoid function can be approximated by the least square polynomial $g(x)$ of degree 3 over the domain $[-8, 8]$ [29]. More precisely, for

$$g(x) = -a_1 x^3 + a_2 x + 0.5$$

with $a_1 = 0.81562/8^3$ and $a_2 = 1.20096/8$, it holds that $g(x) \approx \sigma(x)$ for $x \in [-8, 8]$.

Replacing $\sigma(x)$ with $g(x)$ in Equation (6) yields a polynomial $f(x)$ of degree 4. The monomials of f consist only of variables from the same $\mathbf{x}_i$, which means that all of them can be precomputed by client i. Thus, as established in Section 5, instead of encrypting just $\mathbf{x}_i$, client i precomputes all unique monomials up to degree 4 and encrypts them as well, yielding $\widetilde{\mathbf{x}}_i$. In total, as we only have to consider unique monomials, this extends the number of values in each ciphertext to $\widetilde{m} = \frac{1}{24}m^4 + \frac{5}{12}m^3 + \frac{35}{24}m^2 + \frac{37}{12}m + 2$. This allows us to capture the update function as a inner-product with the expanded encrypted values. The coefficients for the function can be found in Appendix C.

Remark 4. As $\widetilde{m}$ grows polynomial in m, this method is only suitable for datasets with relatively small number of attributes. For a large number of m, we may have to resort to a linear approximation of the sigmoid function ($g'(x) = 0.5 + 0.25x$). Then, the number of values to be encrypted shrinks down to $\widetilde{m} = 0.5 \cdot (3m + m^2)$. Although this number is still quadratic, the only existing MIFE scheme which supports quadratic functions also requires to encrypt $(\mathbf{x} \otimes \mathbf{x}, \mathbf{x}, 1)$ to obtain

truly quadratic functions [5], and they lack to protect the intermediate results by incorporating noise.

To ensure that (ϵ, δ)-DP is provided after the last iteration, we set this as our privacy budget $(\epsilon_{\max}, \delta_{\max})$. In each iteration, a fraction of this privacy budget is consumed without exceeding $(\epsilon_{\max}, \delta_{\max})$ in total. The privacy budget could be divided equally among all iterations, i.e., each iteration fulfills $(\epsilon_{\max}/T, \delta_{\max}/T)$-DP. However, to achieve a better convergence, we dynamically adapt the the privacy budget consumed in each iteration by spending less at the beginning and more towards the end, as it is also done in [18]. In other words, in the first iterations, we statistically have larger noise values, which can still be smoothed out in the later iterations, where statistically smaller noise values are added.

To determine the amount of noise, we must look at the sensitivity of the update function. Although each of the coefficients of $\boldsymbol{\theta}^{(t)}$ is calculated separately in the GD, the whole iteration can be seen as evaluating a function with an $(m+1)$-dimensional output. Without loss of generality, we may assume $\mathbf{x}_i \in [0,1]^{m+2}$. Let $\Theta = \sum_{j \in [0;m]} |\boldsymbol{\theta}[j]|$. Then, the l_2-sensitivity of each iteration is bounded by

$$\Delta(F_{\boldsymbol{\theta}}) \leq \sqrt{m+1}\frac{\alpha}{n}(1 + |a_1\Theta^3 - a_2\Theta|). \tag{7}$$

Results We train a logistic regression on four standard benchmarking datasets from the medical context: the Low Birth Weight Study (LBW) [34], Prostata Cancer Study (PCW) [36], Umaru Impact Study (UIS) [37] and Nhanes III [35]. Table 3 shows the dataset sizes and the total runtime for 50 iterations. We did not consider any optimizations of the GD such as for example batching, which could further improve the overall runtime.

The communication costs for each dataset are displayed in Table 4. The scaling factor is set to be $s = 10^6$. For LBW, PCS and UIS, we set the modulus $q = 2^{64}$ and for NHANES III, $q = 2^{72}$. We noticed that $y \in [-5,5]$, i.e., each queried function slot consists of $\lceil \log_2(10 \cdot s) \rceil$ bits. The rest of the table can be computed according to Table 1.

Table 3. Runtime for training of the logistic regression with 50 iterations on different datasets in minutes.

Dataset	n	m	Total runtime
LBW	189	10	0.19 min
PCS	380	8	0.24 min
UIS	575	8	0.33 min
NHANES	16 427	11	42.29 min

Figure 3 displays the utility compared to LDP in form of accuracy of the first three datasets for a fixed number of iterations $T = 50$ in dependency of $\epsilon_{\max}$. The parameter $\delta_{\max} = 1/n$ is fixed.

Table 4. Package sizes for the encryption process per client, per iteration in the training process and in total after 50 iterations, including the encryption key and ciphertext exchange.

Dataset	Encryption		GD Iteration		Total
	ek_i	ct	f	dk_f	
LBW	256 b	7.9 KB	32.61 KB	88 B	3.06 MB
PCS	256 b	3.93 KB	13.29 KB	72 B	2.13 MB
UIS	256 b	3.93 KB	13.29 KB	72 B	2.88 MB
NHANES	256 b	12.10 KB	48.41 KB	108 B	197.02 MB

The utility highly depends on the dataset, the privacy budget and the learning rate, which can be also seen through the LDP values. For small epsilon, the noise may be too large such that convergence is not possible, which we marked by setting the accuracy to 0. We stress that by simply guessing the dependent variable, an accuracy of 0.5 is always possible, which can be also achieved by a random model. Therefore, LDP is not better than guessing for small epsilon.
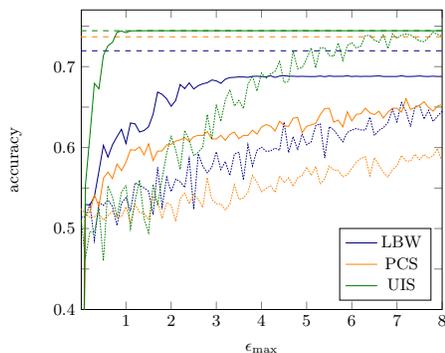
Fig. 3. Model utility for 50 rounds, in dependency of ϵ_{max}. The dashed line shows peak accuracy on plaintext after 500 rounds. Dotted lines represent the maximal accuracy achieved with LDP with 500 rounds.
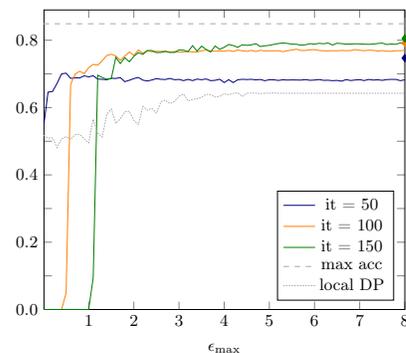
Fig. 4. Model utility on Nhanes III versus ϵ_{max} and training rounds. The dashed line shows peak accuracy on plaintext after 500 rounds; the diamond marks max convergence without noise. Dotted lines indicate LDP's best accuracy at 500 rounds. Accuracy $= 0$ means no convergence.

Another trade-off comes from the number of iterations and the privacy budget. The more rounds we train with a fixed privacy budget, the smaller the privacy budget per iteration and thus, the larger the noise. On the other hand, less iterations may not be enough to reach a good convergence. Figure 4 visualize this consideration based on NHANES III. Even without noise, a relatively small

number of iterations may not be enough to reach the best possible parameter, but for higher number of iterations, it takes a larger ϵ to obtain a good utility.

Hence, although the training of our logistic regression is relatively fast (see Table 3), there is still the drawback regarding the number of iterations and the privacy. However, this problem is a common problem using private GD and only indirectly dependent on our scheme. In fact, since we only add the noise at the end of the iteration, i.e., after all inputs of the clients have been combined, and not before, the amount of noise needed is reduced, as can be seen compared to LDP in Figure 3 and Figure 4. This is because one individual does not influence the function outcome as heavily.

8   Conclusion

In this work, we introduced the class of DyNMCFE, which supports DP and can be used in PPML settings where the data holders want to be excluded from the analysis and the authority has limited resources. We gave a concrete instantiation of a DyNMCFE scheme, namely DyNo. This scheme is faster than any previous instantiation of noisy FE, such that even the training of a logistic regression model can be realized in reasonable time. We believe that it can also be embedded into other ML protocols to obtain output privacy. To further improve the utility and efficiency, many optimizations known from privacy-preserving ML are possible, for example optimal partition of the privacy budget or batching.

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015), https://www.tensorflow.org/, software available from tensorflow.org
2. Abadi, M., Chu, A., Goodfellow, I., McMahan, H.B., Mironov, I., Talwar, K., Zhang, L.: Deep learning with differential privacy. In: Proceedings of the 2016 ACM SIGSAC conference on computer and communications security. pp. 308–318 (2016)
3. Abdalla, M., Benhamouda, F., Kohlweiss, M., Waldner, H.: Decentralizing inner-product functional encryption. pp. 128–157 (2019). https://doi.org/10.1007/978-3-030-17259-6_5
4. Abdalla, M., Catalano, D., Fiore, D., Gay, R., Ursu, B.: Multi-input functional encryption for inner products: Function-hiding realizations and constructions without pairings. pp. 597–627 (2018). https://doi.org/10.1007/978-3-319-96884-1_20
5. Agrawal, S., Goyal, R., Tomida, J.: Multi-input quadratic functional encryption: Stronger security, broader functionality. In: Theory of Cryptography Conference. pp. 711–740. Springer (2022)

6. Agrawal, S., Libert, B., Stehlé, D.: Fully secure functional encryption for inner products, from standard assumptions. pp. 333–362 (2016). https://doi.org/10.1007/978-3-662-53015-3_12

7. Authors, T.G.: crypto/aes package. https://pkg.go.dev/crypto/aes@go1.23.2 (2024), https://pkg.go.dev/crypto/aes@go1.23.2, version go1.23.2

8. Bakas, A., Michalas, A.: Heal the privacy: Functional encryption and privacy-preserving analytics. arXiv preprint arXiv:2205.03083 (2022)

9. Bakas, A., Michalas, A., Dimitriou, T.: Private lives matter: A differential private functional encryption scheme. In: Proceedings of the Twelveth ACM Conference on Data and Application Security and Privacy. pp. 300–311 (2022)

10. Balle, B., Wang, Y.X.: Improving the Gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 394–403. PMLR (10–15 Jul 2018), https://proceedings.mlr.press/v80/balle18a.html

11. Baltico, C.E.Z., Catalano, D., Fiore, D., Gay, R.: Practical functional encryption for quadratic functions with applications to predicate encryption. In: Annual International Cryptology Conference. pp. 67–98. Springer (2017)

12. Boyle, E., Chung, K.M., Pass, R.: On extractability obfuscation. pp. 52–73 (2014). https://doi.org/10.1007/978-3-642-54242-8_3

13. Carpov, S., Fontaine, C., Ligier, D., Sirdey, R.: Illuminating the dark or how to recover what should not be seen in fe-based classifiers. Proceedings on Privacy Enhancing Technologies **2020**(2), 5–23 (2020)

14. Chang, Y., Zhang, K., Gong, J., Qian, H.: Privacy-preserving federated learning via functional encryption, revisited. IEEE Transactions on Information Forensics and Security **18**, 1855–1869 (2023)

15. Chotard, J., Dufour Sans, E., Gay, R., Phan, D.H., Pointcheval, D.: Decentralized multi-client functional encryption for inner product. pp. 703–732 (2018). https://doi.org/10.1007/978-3-030-03329-3_24

16. Chotard, J., Dufour-Sans, E., Gay, R., Phan, D.H., Pointcheval, D.: Dynamic decentralized functional encryption. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology – CRYPTO 2020. pp. 747–775. Springer International Publishing, Cham (2020)

17. Datta, P., Okamoto, T., Tomida, J.: Full-hiding (unbounded) multi-input inner product functional encryption from the k-Linear assumption. pp. 245–277 (2018). https://doi.org/10.1007/978-3-319-76581-5_9

18. Du, J., Li, S., Chen, X., Chen, S., Hong, M.: Dynamic differential-privacy preserving sgd. arXiv preprint arXiv:2111.00173 (2021)

19. Dufour-Sans, E., Gay, R., Pointcheval, D.: Reading in the dark: Classifying encrypted digits with functional encryption. Cryptology ePrint Archive (2018)

20. Dwork, C., Roth, A.: The algorithmic foundations of differential privacy. Foundations and Trends® in Theoretical Computer Science **9**(3–4), 211–407 (2014)

21. Escobar, F.A., Canard, S., Laguillaumie, F., Phan, D.H.: Computational differential privacy for encrypted databases supporting linear queries. Proceedings on Privacy Enhancing Technologies **4**, 583–604 (2024)

22. Fan, T., Kang, Y., Ma, G., Chen, W., Wei, W., Fan, L., Yang, Q.: Fate-llm: A industrial grade federated learning framework for large language models. arXiv preprint arXiv:2310.10049 (2023)

23. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. SIAM Journal on Computing **45**(3), 882–929 (2016)

24. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the forty-first annual ACM symposium on Theory of computing. pp. 169–178 (2009)

25. Goldreich, O.: Secure multi-party computation. Manuscript. Preliminary version **78**(110), 1–108 (1998)

26. Keller, M.: Mp-spdz: A versatile framework for multi-party computation. In: Proceedings of the 2020 ACM SIGSAC conference on computer and communications security. pp. 1575–1590 (2020)

27. Kim, A., Song, Y., Kim, M., Lee, K., Cheon, J.H.: Logistic regression model training based on the approximate homomorphic encryption. BMC medical genomics **11**, 23–31 (2018)

28. Kim, M., Lee, J., Ohno-Machado, L., Jiang, X.: Secure and differentially private logistic regression for horizontally distributed data. IEEE Transactions on Information Forensics and Security **15**, 695–710 (2019)

29. Kim, M., Song, Y., Wang, S., Xia, Y., Jiang, X., et al.: Secure logistic regression based on homomorphic encryption: Design and evaluation. JMIR medical informatics **6**(2), e8805 (2018)

30. Li, M., Tian, Y., Zhang, J., Fan, D., Zhao, D.: The trade-off between privacy and utility in local differential privacy. In: 2021 International Conference on Networking and Network Applications (NaNA). pp. 373–378 (2021). https://doi.org/10.1109/NaNA53684.2021.00071

31. Li, T., Sahu, A.K., Talwalkar, A., Smith, V.: Federated learning: Challenges, methods, and future directions. IEEE signal processing magazine **37**(3), 50–60 (2020)

32. Libert, B., Titiu, R.: Multi-client functional encryption for linear functions in the standard model from LWE. pp. 520–551 (2019). https://doi.org/10.1007/978-3-030-34618-8_18

33. Ligier, D., Carpov, S., Fontaine, C., Sirdey, R.: Privacy preserving data classification using inner-product functional encryption. In: International Conference on Information Systems Security and Privacy. vol. 2, pp. 423–430. SciTePress (2017)

34. LogisticDx: Diagnostic tests for models with a binomial response. lbw: Low birth weight study data (8 2024), https://rdrr.io/rforge/LogisticDx/man/lbw.html

35. LogisticDx: Diagnostic tests for models with a binomial response. nhanes3: Nhanes iii data (8 2024), https://rdrr.io/rforge/LogisticDx/man/nhanes3.html

36. LogisticDx: Diagnostic tests for models with a binomial response. pcs: Prostate cancer study data (8 2024), https://rdrr.io/rforge/LogisticDx/man/pcs.html

37. LogisticDx: Diagnostic tests for models with a binomial response. uis: Umaru impatct study data (8 2024), https://rdrr.io/rforge/LogisticDx/man/uis.html

38. Mahawaga Arachchige, P.C., Bertok, P., Khalil, I., Liu, D., Camtepe, S., Atiquzzaman, M.: Local differential privacy for deep learning. IEEE Internet of Things Journal **7**(7), 5827–5842 (2020). https://doi.org/10.1109/JIOT.2019.2952146

39. Mera, J.M.B., Karmakar, A., Marc, T., Soleimanian, A.: Efficient lattice-based inner-product functional encryption. pp. 163–193 (2022). https://doi.org/10.1007/978-3-030-97131-1_6

40. Naseri, M., Hayes, J., Cristofaro, E.D.: Toward robustness and privacy in federated learning: Experimenting with local and central differential privacy. CoRR **abs/2009.03561** (2020), https://arxiv.org/abs/2009.03561

41. Panzade, P., Takabi, D.: Fenet: Privacy-preserving neural network training with functional encryption. In: Proceedings of the 9th ACM International Workshop on Security and Privacy Analytics. pp. 33–43 (2023)

42. Rao, S., Mahto, D., Yadav, D.K., Khan, D.: The aes-256 cryptosystem resists quantum attacks. Int. J. Adv. Res. Comput. Sci **8**(3), 404–408 (2017)

43. Ryffel, T., Dufour-Sans, E., Gay, R., Bach, F., Pointcheval, D.: Partially encrypted machine learning using functional encryption. arXiv preprint arXiv:1905.10214 (2019)
44. Slawomirski, L., et al.: Progress on implementing and using electronic health record systems: Developments in oecd countries as of 2021. OECD Health Working Papers **No. 160** (2023). https://doi.org/10.1787/4f4ce846-en
45. of Standards, N.I., (NIST), T., Dworkin, M.J., Turan, M.S., Mouha, N.: Advanced encryption standard (aes) (2023-05-09 04:05:00 2023). https://doi.org/https://doi.org/10.6028/NIST.FIPS.197-upd1, https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=936594
46. Team, D.P.: Differential privacy library (dp lib v2.0.0) (2024), https://github.com/google/differential-privacy
47. Viand, A., Knabenhans, C., Hithnawi, A.: Verifiable fully homomorphic encryption (2023), https://arxiv.org/abs/2301.07041
48. Waters, B.: A punctured programming approach to adaptively secure functional encryption. In: Annual Cryptology Conference. pp. 678–697. Springer (2015)
49. Xu, R., Joshi, J.B., Li, C.: Cryptonn: Training neural networks over encrypted data. In: 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS). pp. 1199–1209. IEEE (2019)
50. Zalonis, J., Armknecht, F., Scheu-Hachtel, L.: Differentially private functional encryption. Proceedings on Privacy Enhancing Technologies **2**, 509–530 (2024)
51. Zhao, C., Zhao, S., Zhao, M., Chen, Z., Gao, C.Z., Li, H., Tan, Y.a.: Secure multi-party computation: theory, practice and applications. Information Sciences **476**, 357–372 (2019)
52. Ziller, A., Trask, A., Lopardo, A., Szymkow, B., Wagner, B., Bluemke, E., Nounahon, J.M., Passerat-Palmbach, J., Prakash, K., Rose, N., et al.: Pysyft: A library for easy federated learning. Federated Learning Systems: Towards Next-Generation AI pp. 111–139 (2021)

A   Broader definition of NMCFE

Definition 8 (Noisy Multi-Client Functional Encryption). *Let $\{\mathcal{F}_n^m\}_{n\in\mathbb{N}}$ be a family of sets $\mathcal{F}_n$ of functions $f : \mathcal{X}_1 \times \cdots \times \mathcal{X}_n \to \mathcal{Y}$. Let $\mathcal{L} = \{0,1\}^* \cup \{\bot\}$ be a set of labels. A noisy multi-client functional encryption scheme (NMCFE) for $\mathcal{F}_n$ and $\mathcal{L}$ is a tuple of four efficient algorithms $\mathsf{NMCFE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ of the following form:*

$\mathsf{Setup}(1^\lambda, m, n)$**:** *Takes as input the security parameter λ, the number of clients n and vector length m. It outputs a set of public parameters pp, a master secret key msk as well as secret keys ek_i, for each slot $i \in [n]$. All of the remaining algorithms implicitly take pp.*

$\mathsf{Enc}(\mathsf{ek}_i, x_i, \ell)$**:** *Takes as input the secret key ek_i for a slot $i \in [n]$, a message $x_i \in \mathcal{X}_i$ and a label $\ell \in \mathcal{L}$. It outputs a ciphertext $\mathsf{ct}_{i,\ell}$.*

$\mathsf{KeyGen}(\mathsf{msk}, f, \ell, \mathbb{D})$**:** *Takes as input the master secret key msk, a function $f \in \mathcal{F}_n$, a label $\ell \in \mathcal{L}$ and a distribution $\mathbb{D}$ over some $\Delta \subseteq \mathcal{Y}$ such that $\Pr[f + \Delta \in \mathcal{F}_n] = 1$. Sample $\nu \leftarrow \mathbb{D}$ and output a decryption key $\mathsf{dk}_{f,\ell}$.*

$\mathsf{Dec}(\mathsf{dk}_{f,\ell}, \mathsf{ct}_{1,\ell}, \ldots, \mathsf{ct}_{n,\ell})$**:** *Takes as input the decryption key $\mathsf{dk}_{f,\ell}$ and n ciphertexts $\mathsf{ct}_{1,\ell}, \ldots, \mathsf{ct}_{n,\ell}$, all encrypted under the same label used for the decryption key. It outputs a function evaluation $f(x_1, \ldots, x_n) + \nu \in \mathcal{Y}$.*

The following is a generalized security definition that covers existing definitions for (noisy) MIFE and MCFE. Hence, our notation includes several options. We will briefly explain some of these to enhance the clarity of the technical definition. If we set the noise to always be zero, we obtain the classical security definition of MCFE. Additionally, if we set $\mathcal{L} = \{\perp\}$, we get the security definition for MIFE. Another consideration is the number of required encryption queries per slot $i \in [n]$, i.e., exactly one (one) or at least one (pos).

The specification for handling the labels in the decryption keys is another consideration when building an NMCFE scheme. The keys can either be valid for all ciphertexts (all) or only for those, which share the same label as the key (lab). This is directly related to Equation (8), as the decryption must only protect the trivial attack, i.e., the inputs of the ciphertext queries posed by the adversary yield different function evaluations, if the key and the ciphertexts are generated under the same label. If the key is not tied to the label (all), then Equation (8) should hold for all queried functions, as long as only ciphertexts under the same label are combined.

Finally, it makes a difference at what point clients are corrupted, i.e., whether they are corrupted from the start (static (sta) corruptions) or at a later point (adaptive (ad) corruptions).

In the following, $\mathbb{D}_\nu$ defines the distribution that outputs ν with probability one.

Definition 9 (Security of NMCFE). *Let* NMCFE *be an NMCFE scheme for n clients and $\mathcal{L}$ be a label set. For specifications* $\mathrm{ww} \in \{\mathrm{sta}, \mathrm{ad}\}$, $\mathrm{xx} \in \{\mathrm{nh}, \mathrm{mh}\}$, $\mathrm{yy} \in \{\mathrm{one}, \mathrm{pos}\}$, $\mathrm{zz} \in \{\mathrm{lab}, \mathrm{all}\}$, *and any security parameter* λ, *consider the following game* $\mathrm{ww\text{-}xx\text{-}yy\text{-}zz\text{-}IND}_\beta^{\mathsf{NMCFE}}$ *between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. The game involves a set $\mathcal{HS}$ of honest clients, initialized to $\mathcal{HS} := [n]$, and a set of corrupted clients $\mathcal{CS}$, initialized to $\mathcal{CS} := \emptyset$.*

Initialization: *At the beginning of their interaction, $\mathcal{C}$ runs* $(\mathsf{pp}, \mathsf{msk}, \{\mathsf{ek}_i\}_{i \in [n]}) \leftarrow \mathsf{Setup}(1^\lambda, m, n)$. *Then, it chooses a random bit $\beta \leftarrow \{0,1\}$ and gives pp to $\mathcal{A}$.*

Encryption queries: *$\mathcal{A}$ can adaptively make encryption queries $\mathsf{QEnc}(i, x_i, \ell)$, which $\mathcal{C}$ replies with $\mathsf{ct}_{i,\ell} \leftarrow \mathsf{Enc}(\mathsf{ek}_i, x_i, \ell)$. Denote by $q_{c,i,\ell}^{\mathsf{Enc}}$ the total number of queries of the form $\mathsf{QEnc}(i, \cdot, \ell)$.*

Challenge encryption queries: *$\mathcal{A}$ can adaptively transmit challenge encryption queries $\mathsf{CQEnc}(i, x_i^0, x_i^1, \ell)$, which are answered with $\mathsf{ct}_{i,\ell} \leftarrow \mathsf{Enc}(\mathsf{ek}_i, x_i^\beta, \ell)$. Denote by $q_{c,i,\ell}^{\mathsf{CEnc}}$ the total number of queries $\mathsf{CQEnc}(i, \cdot, \cdot, \ell)$.*

Decryption key queries: *The adversary $\mathcal{A}$ can adaptively obtain functional decryption keys via queries $\mathsf{QKeyGen}(f, \ell, \nu)$. In response, $\mathcal{C}$ returns $\mathsf{dk}_{f,\ell} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f, \ell, \mathbb{D}_\nu)$. Denote by $q_{k,\ell}^{\mathsf{KeyGen}}$ the total number of queries $\mathsf{CQEnc}(\cdot, \ell, \cdot, \cdot)$.*

Challenge decryption key queries: *The adversary can adaptively pose queries $\mathsf{CQKeyGen}(f, \ell, \nu^0, \nu^1)$ to receive functional decryption keys. In response, $\mathcal{C}$ returns $\mathsf{dk}_{f,\ell} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f, \ell, \mathbb{D}_{\nu^\beta})$. Denote by $q_{k,\ell}^{\mathsf{CKeyGen}}$ the total number of queries $\mathsf{CQEnc}(\cdot, \ell, \cdot, \cdot)$.*

Corruption queries: *$\mathcal{A}$ can query the corruption oracle $\mathsf{QCor}(i)$ to obtain ek_i. $\mathcal{C}$ set $\mathcal{CS} := \mathcal{CS} \cup \{i\}$ and $\mathcal{HS} := \mathcal{HS} \setminus \{i\}$.*

Finalize: $\mathcal{A}$ *outputs a bit* $\beta' \in \{0,1\}$. $\mathcal{C}$ *checks, if* $\mathcal{A}$ *acted admissibly. If not,* $\mathcal{C}$ *sets* $\beta' = 0$. $\mathcal{A}$ *wins, if* $\beta' = \beta$.

Depending on the specification of the game defined in advance, we call $\mathcal{A}$ *admissible, if all of the following conditions hold. Denote by* $q_{c,i,\ell}$ *the total number of encryption and challenge encryption queries to slot* i *under label* ℓ, *i.e.,* $q_{c,i,\ell} = q_{c,i,\ell}^{\mathsf{Enc}} + q_{c,i,\ell}^{\mathsf{CEnc}}$. *Similarly, let* $q_{k,\ell}$ *be the total number of decryption and challenge decryption key queries under label* ℓ, *i.e.,* $q_{k,\ell} = q_{k,\ell}^{\mathsf{KeyGen}} + q_{k,\ell}^{\mathsf{CKeyGen}}$.

- *If* $i \in \mathcal{CS}$, *then for any query* $\mathsf{CQEnc}(i, x_i^0, x_i^1, \ell)$, $x_i^0 = x_i^1$.
- *For any family storing all honest and all challenge encryption queries, i.e.,* $\{\mathsf{CQEnc}(i, x_i^0, x_i^1, \ell)$ *or* $\mathsf{QEnc}(i, x_i, \ell)\}_{i \in \mathcal{HS}}$, *for any family of inputs* $\{x_i \in \mathcal{X}_i\}_{i \in \mathcal{CS}}$, *any label* $\ell \in \mathcal{L}$, *any query* $\mathsf{CQKeyGen}(f_\ell, \ell, \nu^0, \nu^1)$, *we define* $x_i^0 := x_i$ *and* $x_i^1 := x_i$ *for* $i \in \mathcal{CS}$ *and any slot queried to* $\mathsf{QEnc}(i, x_i, \ell)$ *and* $\nu^0 := \nu$ *and* $\nu^1 := \nu$ *for any query* $\mathsf{QKeyGen}(f_\ell, \ell, \nu)$, *and we require that:*

$$f_\ell(x_1^0, \ldots, x_n^0) + \nu_\ell^0 = f_\ell(x_1^1, \ldots, x_n^1) + \nu_\ell^1. \tag{8}$$

  If one slot $i \in \mathcal{HS}$ *has not been queried, then there is no restriction.*
- *If* ww = sta, *the adversary should output* $\mathcal{CS}$ *at the beginning of the game and does not have access to* $\mathsf{QCor}(\cdot)$ *afterwards.*
- *If* xx = mh, *then for all queries* $\mathsf{CQKeyGen}(f, \ell, \nu^0, \nu^1)$, $\nu^0 = \nu^1 = 0$.
- *If* yy = any^{ot}, *then for any slot* $i \in [n]$ *and* $\ell \in \mathcal{L}$, $q_{c,i,\ell} \leq 1$. *In other words, the adversary can pose at most one (challenge) encryption query per slot.*
- *If* yy = one, *then for any slot* $i \in \mathcal{HS}$ *and* $\ell \in \mathcal{L}$, *it holds that if* $q_{c,i,\ell}^{\mathsf{CEnc}} > 0$ *or* $q_{k,\ell}^{\mathsf{CKeyGen}} > 0$, *then* $q_{c,j,\ell} = 1$ *for any slot* $j \in \mathcal{HS}$. *In other words, for any label, if there is one honest (challenge) encryption query or challenge decryption key query, then all honest slots have to be queried exactly once.*
- *If* yy = pos, *then for any slot* $i \in \mathcal{HS}$ *and* $\ell \in \mathcal{L}$, *it holds that if* $q_{c,i,\ell}^{\mathsf{CEnc}} > 0$ *or* $q_{k,\ell}^{\mathsf{CKeyGen}} > 0$, *then* $q_{c,j,\ell} > 0$ *for any slot* $j \in \mathcal{HS}$. *In other words, for any label, the adversary makes either at least one challenge query for each slot* $i \in \mathcal{HS}$ *or none. Moreover, for any label, if the adversary queries a challenge decryption key, they also have to pose at least one (challenge) encryption query per slot.*
- *If* zz = all, *then for any* $f \in \mathcal{F}_n$, $\mathsf{dk}_{f,\ell} = \mathsf{dk}_{f,\ell'}$ *for all* $\ell, \ell' \in \mathcal{L}$. *In other words, the decryption keys for a function* f *are the same for each label and can thus be seen as independent of the label.*

The advantage of $\mathcal{A}$ *in this game is defined as*

$$\mathsf{Adv}_{\mathsf{NMCFE},\mathcal{A}}^{\mathsf{ww\text{-}xx\text{-}yy\text{-}zz\text{-}IND}}(\lambda) = \Big| \Pr(\mathsf{ww\text{-}xx\text{-}yy\text{-}zz\text{-}IND}_0^{\mathsf{NMCFE}}(\lambda, \mathcal{A}) = 1)$$
$$- \Pr(\mathsf{ww\text{-}xx\text{-}yy\text{-}zz\text{-}IND}_1^{\mathsf{NMCFE}}(\lambda, \mathcal{A}) = 1) \Big|$$

We say that an NMCFE scheme NMCFE *provides* ww-xx-yy-zz-IND-*security, if* $\mathsf{Adv}_{\mathsf{NMCFE},\mathcal{A}}^{\mathsf{ww\text{-}xx\text{-}yy\text{-}zz\text{-}IND}}(\lambda) \leq \mathsf{negl}(\lambda)$.

Note that this security definition is the same as Definition 7 for ww-xx-yy-zz = sta-nh-one-lab and if all encryption keys are generated during setup. In fact, for our proposed scheme, DyNo, we will show that thee non-dynamic version fulfills sta-nh-one-lab-IND-security and conclude that this suffices for the IND-security of DyNo.

In case only one label is used, i.e., $\mathcal{L} = \{\perp\}$, the declaration of zz is omitted. Moreover, we omit the label index in the notation above, e.g., only write q_k instead of $q_{k,\ell}$. In a similar manner, if no corruptions are allowed, we omit ww.

To simplify our security proofs, we introduce the notion of 1-label security, as it is common in the MCFE setting [32]. In this modified game, the adversary is only allowed to pose challenge queries under a single label ℓ^*. Although this may seem as a strong restriction, we show via a sequence of hybrid arguments that is equivalent to Definition 7.

Definition 10 (1-label security of NMCFE). *Let* NMCFE *be a NMCFE scheme for n clients and $\mathcal{L}$ be a label set. For* ww, xx, yy, zz *defined as in Definition 7 and $\beta \in \{0,1\}$, we define the game* $1\text{-ww-xx-yy-zz-IND}_\beta^{\mathsf{NMCFE}}$ *as in Definition 7, except for the following changes in the oracle queries:*

Encryption queries: $\mathcal{A}$ *can adaptively make encryption queries* $\mathsf{QEnc}(i, x_i, \ell)$, *which $\mathcal{C}$ replies with* $\mathsf{ct}_{i,\ell} \leftarrow \mathsf{Enc}(\mathsf{ek}_i, x_i, \ell)$. *If one label corresponds to ℓ^* queried to* CQEnc, *the game ends and returns 0.*

Challenge encryption queries: $\mathcal{A}$ *can adaptively transmit challenge encryption queries* $\mathsf{CQEnc}(i, x_i^0, x_i^1, \ell)$, *which are answered with* $\mathsf{ct}_{i,\ell} \leftarrow \mathsf{Enc}(\mathsf{ek}_i, x_i^\beta, \ell)$. *These queries can be made on at most one label ℓ^*. Further queries with distinct labels will be ignored.*

Decryption key queries: $\mathcal{A}$ *can adaptively pose queries* $\mathsf{QKeyGen}(f, \ell, \nu)$ *to obtain functional decryption keys. $\mathcal{C}$ returns* $\mathsf{dk}_{f,\ell} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f, \ell, \mathbb{D}_\nu)$. *If one label corresponds to ℓ' queried to* CQKeyGen, *the game ends and returns 0.*

Challenge decryption key queries: *The adversary can adaptively receive functional decryption keys by posing queries* $\mathsf{CQKeyGen}(f, \ell, \nu^0, \nu^1)$. *In response, $\mathcal{C}$ returns* $\mathsf{dk}_{f,\ell} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f, \ell, \mathbb{D}_{\nu^\beta})$. *These queries can be made on at most one label ℓ'. Further queries with distinct labels will be ignored.*

The same conditions defining the admissibility of an adversary as in Definition 7 apply to an adversary $\mathcal{A}$ playing the $1\text{-ww-xx-yy-zz-IND}_\beta^{\mathsf{NMCFE}}$ *game. The advantage of $\mathcal{A}$ is defined as*

$$\mathsf{Adv}_{\mathsf{NMCFE},\mathcal{A}}^{1\text{-ww-xx-yy-zz-IND}}(\lambda) = \Big| \Pr(1\text{-ww-xx-yy-zz-IND}_0^{\mathsf{NMCFE}}(\lambda, \mathcal{A}) = 1)$$
$$- \Pr(1\text{-ww-xx-yy-zz-IND}_1^{\mathsf{NMCFE}}(\lambda, \mathcal{A}) = 1) \Big|$$

We say that an NMCFE scheme NMCFE *provides* 1-ww-xx-yy-zz-IND-*security, if*

$$\mathsf{Adv}_{\mathsf{NMCFE},\mathcal{A}}^{1\text{-ww-xx-yy-zz-IND}}(\lambda) \leq \mathtt{negl}(\lambda).$$

This definition allows $\ell^* \neq \ell'$. However, note that if this is the case, then $\nu_l^0 = \nu_l^1$ for all decryption key queries $l \in [q_{k,\ell'}]$ under the condition that $\mathcal{A}$ is an admissible adversary. Moreover, $f(x_1^0, \ldots, x_n^0) = f(x_1^1, \ldots, x_n^1)$ for all $\{x_i^b\}_{i \in [n], b \in \{0,1\}}$. Thus, the adversary would always play the game in the xx = mh setting, which is the weaker security definition. The similarity of both security definitions in that particular setting has been proven for example by Libert et al. [32]. We will focus on the case where $\ell^* = \ell'$.

Lemma 1. *Let* MCFE *be a scheme that is* 1-ww-xx-yy-zz-IND-*secure. Then it is also secure against a* PPT *adversary* $\mathcal{A}$ *against its* ww-xx-yy-zz-IND *security. Namely, for any* PPT *adversary* $\mathcal{A}$*, there exists a* PPT *adversary* $\mathcal{B}$ *such that.*

$$\mathsf{Adv}_{\mathsf{MCFE},\mathcal{A}}^{\text{ww-xx-yy-zz-IND}}(\lambda) \leq q_\ell \cdot \mathsf{Adv}_{\mathsf{MCFE},\mathcal{B}}^{\text{1-ww-xx-yy-zz-IND}}(\lambda),$$

where q_ℓ *denotes the number of distinct labels queried by* $\mathcal{A}$ *to* CQEnc *and* CQKeyGen *in the original security game.*

Proof. Let $\mathcal{A}$ be an efficient adversary in the ww-xx-yy-zz-IND-security game. We show that $\mathcal{A}$ implies an MCFE adversary in the 1-ww-xx-yy-zz-IND. Assume $\mathcal{A}$ makes encryption and decryption key queries for q_ℓ distinct labels. Denote by $\ell_j, j \in [q_\ell]$ the jth distinct label that was queried to one of the challenge oracle. We use a standard hybrid argument over the distinct labels that $\mathcal{A}$ queries throughout the attack. More precisely, let G_k with $k \in \{0, \ldots, q_\ell\}$ be the game in which the challenger responds in the following way to encryption and decryption key queries $\mathsf{QEnc}(i, x_i^0, x_i^1, \ell_j)$ and $\mathsf{QKeyGen}(f, \ell_j, \nu^0, \nu^1)$:

  – If $j \leq k$, reply with $\mathsf{Enc}(\mathsf{ek}_i, x_i^0, \ell_j)$ and $\mathsf{KeyGen}(f, \ell_j, \nu^0)$, respectively.
  – If $j > k$, reply with $\mathsf{Enc}(\mathsf{ek}_i, x_i^1, \ell_j)$ and $\mathsf{KeyGen}(f, \ell_j, \nu^1)$, respectively.

We assume that all of the restrictions imposed by the admissibility of the adversary remain the same. By construction, an adversary $\mathcal{A}$ against the ww-xx-yy-zz-IND-security of Definition 7 yields a distinguisher between G_0 and G_{q_ℓ}. On the other hand, for any $k \in \{0, 1, \ldots, q_\ell\}$, an efficient distinguisher $\mathcal{A}_k$ between G_k and G_{k+1} implies the existence of an efficient adversary $\mathcal{B}_k$ in the 1-ww-xx-yy-zz-IND$_\beta^{\mathsf{MCFE}}$ game such that $\mathsf{Adv}_{\mathcal{A}_k}^{k,k+1}(\lambda) = \mathsf{Adv}_{\mathsf{MCFE},\mathcal{B}_k}^{\text{1-ww-xx-yy-zz-IND}}(\lambda)$. The reason is that $\mathcal{B}_k$ can simulate $\mathcal{A}_k$'s view as follows.

Initialization: $\mathcal{B}_k$ obtains pp from its own challenger and hands it to $\mathcal{A}_k$.
Encryption queries: For each query $\mathsf{QEnc}(i, x_i^0, x_i^1, \ell_j)$ from $\mathcal{A}_k$, $\mathcal{B}_k$ acts as follows.
  – If $j \leq k$, $\mathcal{B}_k$ poses the encryption query $\mathsf{QEnc}(i, x_i^0, \ell_j)$ to its own oracle and forwards the response to $\mathcal{A}_k$.
  – If $j = k+1$, $\mathcal{B}_k$ poses the challenge encryption query $\mathsf{CQEnc}(i, x_i^0, x_i^1, \ell_j)$ to its own challenger and relays the response back to $\mathcal{A}_k$.
  – Otherwise, $\mathcal{B}_k$ transmits the query $\mathsf{QEnc}(i, x_i^1, \ell_j)$ and returns the answer to $\mathcal{A}_k$.
Decryption key queries: $\mathcal{B}_k$ does the following for each decryption key query $\mathsf{CQKeyGen}(f, \ell_j, \nu^0, \nu^1)$ from $\mathcal{A}_k$.

- If $j \leq k$, $\mathcal{B}_k$ sends the query $\mathsf{QKeyGen}(f, \ell_j, \nu^0)$ to its own challenger and relays the answer back to $\mathcal{A}_k$.
- If $j = k{+}1$, $\mathcal{B}_k$ poses the challenge encryption query $\mathsf{CQKeyGen}(f, \ell_j, \nu^0, \nu^1)$ and sends the response to $\mathcal{A}_k$.
- Otherwise, $\mathcal{B}_k$ queries $\mathsf{QKeyGen}(f, \ell_j, \nu^1)$ and transmits the answer to $\mathcal{A}_k$.

Corruption queries: For each query $\mathsf{QCor}(i)$ from $\mathcal{A}_k$, $\mathcal{B}_k$ makes the same query to its own corruption oracle and transmits the answer to $\mathcal{A}_k$.

Finalize: $\mathcal{B}_k$ outputs the same bit β' as $\mathcal{A}_k$.

Summing up the underlying probabilities concludes the proof.

B   Security proofs

Throughout this section, denote by $\mathsf{P}(\mathcal{A}, \mathsf{G}_i)$ the probability that an adversary $\mathcal{A}$ outputs the correct β in G_i, i.e., that $\mathcal{A}$ wins G_i.

B.1   Security of $\mathsf{NMIFE}^{\mathrm{ot}}$

To prove the security of $\mathsf{NMIFE}^{\mathrm{ot}}$, we need the information-theoretical security of $\mathsf{MIFE}^{\mathrm{ot}}$. Abdalla et al.[3] showed:

Theorem 3. *([3]) The $\mathsf{MIFE}^{\mathrm{ot}}$ scheme as described above fulfills* ad-mh-one-IND-*security. In other words, for any adversary $\mathcal{A}$, $\mathsf{Adv}^{\text{ad-nh-one-IND}}_{\mathsf{MIFE}^{\mathrm{ot}}, \mathcal{A}}(\lambda) = 0$.*

Using this theorem, we can now show the security of $\mathsf{NMIFE}^{\mathrm{ot}}$.

Proof (Proof of Theorem 1). Consider the adversary $\mathcal{A}$ against the ad-nh-one-IND-security of $\mathsf{NMIFE}^{\mathrm{ot}}$. We prove the theorem using two reductions. Let sel-sta-nh-one-$\mathsf{IND}^{\mathsf{MIFE}^{\mathrm{ot}}}_{\beta}(\lambda)$ be a variant of the sta-nh-one-$\mathsf{IND}^{\mathsf{MIFE}^{\mathrm{ot}}}_{\beta}(\lambda)$ game, where the selective adversary has to additionally specify the encryption challenges $\{\mathbf{x}^b_i\}_{i \in [n], b \in \{0,1\}}$ together with the corrupted set $\mathcal{CS}$ at the beginning of its interaction with the challenger. First, consider an adversary $\mathcal{B}_1$ against the sel-sta-nh-one-IND security of $\mathsf{NMIFE}^{\mathrm{ot}}$. $\mathcal{B}_1$ can simulate $\mathcal{A}$'s view in the following manner.

Upon interaction with $\mathcal{A}$, $\mathcal{B}_1$ guesses $\mathcal{A}$'s ciphertext queries $(\mathbf{x}^0_i, \mathbf{x}^1_i)$ as well as the set of corrupted clients $\mathcal{CS}$ and commits them along with all key queries $(f^l_{\mathbf{y}}, \nu^0_l, \nu^1_l)$, which are admissible with respect to $\{(\mathbf{x}^0_i, \mathbf{x}^1_i)_{i \in [n]}\}$ to their own selective oracle.

Whenever $\mathcal{A}$ poses a ciphertext query $(\mathbf{x}^0_i, \mathbf{x}^1_i)$, $\mathcal{B}_1$ checks if their guess was correct. If so, they query their own oracle and send the result to $\mathcal{A}$. Otherwise, they output 0.

Whenever $\mathcal{A}$ poses a corruption query for $i \in [n]$, $\mathcal{B}_1$ checks if $i \in \mathcal{CS}$. If so, they give ek_i to $\mathcal{A}$. Otherwise, they output 0.

Whenever $\mathcal{A}$ poses a key query $(f^l_{\mathbf{y}}, \nu^0_l, \nu^1_l)$, $\mathcal{B}_1$ checks if it belongs to its set of key queries. If so, they query their own oracle and send the result to $\mathcal{A}$. Otherwise, they output 0.

If all guesses were correct, $\mathcal{B}_1$ outputs the same bit as $\mathcal{A}$.

Note that if all queries were guessed successfully in advance, $\mathcal{B}_1$ perfectly simulates $\mathcal{A}$'s view. The probability that all ciphertext queries are guessed correctly is exactly q^{-2nm}. On the other hand, the probability that $\mathcal{B}_1$ guesses $\mathcal{CS}$ correctly is 2^{-n}. If $\mathcal{B}_1$'s guess coincides with $\mathcal{A}$'s initial, then all of the key queries $\mathcal{A}$ poses are covered by the set of key queries $\mathcal{B}_1$ committed to their oracle as it contains all admissible key queries with respect to the same ciphertexts. Thus, it holds that

$$\mathsf{Adv}^{\mathsf{ad\text{-}nh\text{-}one\text{-}IND}}_{\mathsf{NMIFE}^{\mathsf{ot}},\mathcal{A}}(\lambda) \leq 2^{-n} q^{-2nm} \mathsf{Adv}^{\mathsf{sta\text{-}nh\text{-}one\text{-}sel\text{-}IND}}_{\mathsf{NMIFE}^{\mathsf{ot}},\mathcal{B}_1}(\lambda).$$

Next, let $\mathcal{B}_2$ be an adversary against the ad-mh-one-IND-security of $\mathsf{MIFE}^{\mathsf{ot}}$ for the family $\mathcal{F}_{q,n+\lceil q_k/m \rceil}$, where q_k is the amount of key queries $\mathcal{B}_1$ poses. They simulate the view of $\mathcal{B}_1$ as follows.

$\mathcal{B}_2$ obtains the set of ciphertext queries $\{\mathbf{x}_i^b\}_{i \in [n], b \in \{0,1\}}$ and function queries $\{(f_{\mathbf{y}}^l, \nu_l^0, \nu_l^1)\}_{l \in [q_k]}$ from $\mathcal{B}_1$. For $j \in [\lceil q_k/m \rceil]$, $\mathcal{B}_2$ sets $\mathbf{x}_{n+i}^\beta = (\nu_{1+(i-1)m}^\beta, \ldots, \nu_{m+(i-1)m}^\beta)$, where $\nu_l^\beta = 0$ for all $l > q_k$. $\mathcal{B}_2$ also receives the set of corrupted clients, $\mathcal{CS}$. They query their own corruption oracle on any index $i \in \mathcal{CS}$ to obtain ek_i. $\{\mathsf{ek}_i\}_{i \in \mathcal{CS}}$ is handed to $\mathcal{B}_1$.

$\mathcal{B}_2$ answers $\mathcal{B}_1$'s encryption queries for $i \in [n]$ by invoking their own oracle to obtain ct_i. This is directly sent to $\mathcal{B}_1$. Further, $\mathcal{B}_2$ also queries their oracle on $(\mathbf{x}_i^0, \mathbf{x}_i^1)$ for $i \in [\lceil q_k/m \rceil]$.

To answer the lth decryption key query of $(f_{\mathbf{y}}^l, \nu_l^0, \nu_l^1)$ of $\mathcal{B}_1$, $\mathcal{B}_2$ sets $\widetilde{\mathbf{y}} = (\mathbf{y}^\top, \mathbf{e}_l^\top)^\top$, where $\mathbf{e}_l$ is the vector of unity of dimension dm with $d \in \mathbb{N}$ being the smallest number such that $q_k \leq dm$. Parsing $\widetilde{\mathbf{y}} = (\widetilde{\mathbf{y}}_1, \ldots, \widetilde{\mathbf{y}}_{n+\lceil q_k/m \rceil}) \in \mathbb{Z}^{m(n+\lceil q_k/m \rceil)}$, they query their oracle for $f_{\widetilde{\mathbf{y}}}^l$ to receive z. Finally, $\mathcal{B}_2$ calculates $\widetilde{z} = z - \sum_{i \in [\lceil q_k/m \rceil]} \langle \mathsf{ct}_{n+i}, \widetilde{\mathbf{y}}_{n+i} \rangle$ and sends it to $\mathcal{B}_1$.

At the end of their interaction, $\mathcal{B}_2$ outputs the same bit as $\mathcal{B}_1$.

Clearly, the ciphertext queries are equivalent in both cases. $\mathcal{B}_2$'s key queries are admissible with respect to (8) as by construction,

$$\sum_{i \in [n+\lceil q_k/m \rceil]} \langle \mathsf{ct}_i, \widetilde{\mathbf{y}}_i^l \rangle - z = \sum_{i \in [n]} \langle \mathbf{x}_i^\beta, \mathbf{y}_i^l \rangle + \nu_l^\beta = \sum_{i \in [n]} \langle \mathbf{x}_i^0, \mathbf{y}_i^l \rangle + \nu_l^0$$

for all $\beta \in \{0,1\}$ by the restrictions imposed on $\mathcal{B}_1$.

Moreover, it holds that

$$\begin{aligned}\widetilde{z} &= \sum_{i \in [n+\lceil q_k/m \rceil]} \langle \mathsf{ek}_i, \widetilde{\mathbf{y}}_i^l \rangle - \sum_{i \in [\lceil q_k/m \rceil]} \langle \mathbf{x}_{n+i} + \mathsf{ek}_{n+i}, \widetilde{\mathbf{y}}_{n+i}^l \rangle \\ &= \sum_{i \in [n]} \langle \mathsf{ek}_i, \mathbf{y}_i^l \rangle - \nu_l^\beta.\end{aligned}$$

Thus, $\mathcal{B}_2$ perfectly simulates $\mathcal{B}_1$'s view. But as $\mathsf{Adv}^{\mathsf{ad\text{-}mh\text{-}one\text{-}IND}}_{\mathsf{MIFE}^{ot},\mathcal{B}_2}(\lambda) = 0$ by Theorem 3, the claim follows.

B.2   Security of **DyNo**

First, let us recall the formal definition of a PRF.

Definition 11. *For any PRF* PRF *from $\mathcal{D}$ to $\mathcal{R}$ and any security parameter λ we define the experiment* $\mathsf{IND}_\beta^{\mathsf{PRF}}$ *between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ in the following way.*

Initialization: *$\mathcal{C}$ samples $\mathsf{K} \xleftarrow{\$} \{0,1\}^\lambda$ and $\beta \xleftarrow{\$} \{0,1\}$.*
Challenge phase: *For a query $\ell \in \mathcal{D}$ posed by $\mathcal{A}$, $\mathcal{C}$ returns $\mathsf{PRF}(\mathsf{K},\ell)$ if $\beta = 0$ and $\mathsf{RF}(\ell)$ otherwise. Here, RF denotes a function computed on the fly.*
Finalize: *$\mathcal{A}$ outputs a bit $\beta' \in \{0,1\}$. If $\beta' = \beta$, $\mathcal{A}$ wins.*

The advantage of an adversary $\mathcal{A}$ is defined as

$$\mathsf{Adv}_{\mathsf{PRF},\mathcal{A}}(\lambda) = \left| \Pr(\mathsf{IND}_0^{\mathsf{PRF}}(\lambda,\mathcal{A}) = 1) - \Pr(\mathsf{IND}_1^{\mathsf{PRF}}(\lambda,\mathcal{A}) = 1) \right|.$$

PRF *is called secure, if for any* PPT *adversary $\mathcal{A}$, there exists a negligible function* `negl` *such that $\mathsf{Adv}_{\mathsf{PRF},\mathcal{A}}(\lambda) \leq \mathtt{negl}(\lambda)$.*

For our security proof, we need the non-dynamic version of DyNo, which is defined as follows.

Construction 3. (Non-dynamic DyNo**)**   *Let $\mathcal{F}_{q,n}^m$ be the class of multi-input inner products over $\mathbb{Z}_q$. The non-dynamic* DyNo *scheme for $\mathcal{F}_{q,n}^m$ consists of the following algorithms:*

$\mathsf{Setup}(1^\lambda, m, n)$**:** *On input of the security parameter λ, number of clients n and vector length m, sample $\mathsf{ek}_i \xleftarrow{\$} \{0,1\}^\lambda$ for all slots $i \in [n]$. Set $\mathsf{msk} := (\{\mathsf{ek}_i\}_{i \in [n]})$ and output both as well as the public parameters pp.*

$\mathsf{Enc}(\mathsf{ek}_i, \mathbf{x}_i, \ell)$**:** *On input of the secret key ek_i, a plaintext $\mathbf{x}_i \in \mathbb{Z}_q^m$ for slot $i \in [n]$ and a label $\ell \in \mathcal{L}$, calculate $\zeta = \mathsf{PRF}(\mathsf{ek}_i, \ell)$ and return $\mathsf{ct}_{i,\ell} = (i, \ell, \mathbf{c}_i := \mathbf{x}_i + \zeta \mod q)$.*

$\mathsf{KeyGen}(\mathsf{msk}, f_{\mathbf{y}}, \ell, \mathbb{D})$**:** *On input of the master secret key msk, a function $f_{\mathbf{y}}$, a label $\ell \in \mathcal{L}$ and a noise distribution $\mathbb{D}$, sample $\nu \leftarrow \mathbb{D}$ and calculate $\zeta_i = \mathsf{PRF}(\mathsf{ek}_i, \ell)$ for all $i \in [n]$. Return $\mathsf{dk}_{\mathbf{y},\ell} = (\ell, \mathbf{y}, z := \sum_{i \in [n]} \langle \zeta_i, \mathbf{y}_i \rangle - \nu \mod q)$.*

$\mathsf{Dec}(\mathsf{dk}_{\mathbf{y},\ell}, \{\mathsf{ct}_{i,\ell}\}_{i \in [n]})$**:** *On input of the decryption key $\mathsf{dk}_{\mathbf{y},\ell}$ and ciphertexts $\mathsf{ct}_{i,\ell} = (i, \ell, \mathbf{c}_i)$, parse $\mathsf{dk}_{\mathbf{y},\ell} = (\ell, \{\mathbf{y}_i\}_{i \in [n]}, z)$ and return $\sum_{i \in [n]} \langle \mathbf{c}_i, \mathbf{y}_i \rangle - z \mod q$.*

Theorem 4. *Let* NMCFE *be as in Construction 3 and* DyNo *as in Construction 2. Then, for any* PPT *adversary $\mathcal{A}$ against the* IND*-security of* DyNo*, there exists a* PPT *adversary $\mathcal{B}$ such that*

$$\mathsf{Adv}_{\mathsf{DyNo},\mathcal{A}}^{\mathsf{IND}}(\lambda) \leq \mathsf{Adv}_{\mathsf{NMCFE},\mathcal{B}}^{\mathsf{sta\text{-}nh\text{-}one\text{-}lab\text{-}IND}}(\lambda).$$

Proof. We denote by $\mathbf{0}_k$ the all-zero vector of length k. $\mathcal{B}$ simulates $\mathcal{A}$'s view as follows. Upon interaction, $\mathcal{B}$ sets $n := n_{\max}$ and $m := m_{\max}$, which works as both are at most polynomial in λ. Moreover, $\mathcal{B}$ sets $\mathcal{E} := \emptyset$ as the set of indices for which an encryption key has been queried.

If $\mathcal{A}$ poses a corruption query for slot $i \in [n]$, $\mathcal{B}$ queries their own corruption oracle to obtain ek_i, which is forwarded to $\mathcal{A}$. $\mathcal{B}$ updates $\mathcal{E} := \mathcal{E} \cup \{i\}$.

Whenever $\mathcal{A}$ queries its encryption key oracle on index i, $\mathcal{B}$ sets $\mathcal{E} := \mathcal{E} \cup \{i\}$.

Whenever $\mathcal{A}$ poses an encryption query $(i, \mathbf{x}_i^0, \mathbf{x}_i^1, \ell)$, where $\mathbf{x}_i^\beta$ is of length m_ℓ $\mathcal{B}$ checks if $i \in \mathcal{E}$. If so, they query there own oracle $\mathsf{CQEnc}(i, \widetilde{\mathbf{x}}_i^{\,0}, \widetilde{\mathbf{x}}_i^{\,1}, \ell)$, where $\widetilde{\mathbf{x}}_i^{\,\beta} = (\mathbf{x}_i^\beta, \mathbf{0}_{m-m_\ell}) \in \mathbb{Z}_q^m$ to obtain $\mathsf{ct}_{i,\ell}$. They hand $\mathsf{ct}'_{i,\ell} = (\mathsf{ct}_{i,\ell}[1], \ldots, \mathsf{ct}_{i,\ell}[m_\ell])$ to $\mathcal{A}$. Otherwise, they return nothing.

Whenever $\mathcal{A}$ poses a decryption key query $(S, f, \ell, \nu^0, \nu^1)$, $\mathcal{B}$ checks if $S \subseteq \mathcal{E}$. If not, return $\bot$. Otherwise, $\mathcal{B}$ interprets $f = \{\mathbf{y}_i\}_{i \in S}$. They set $\widetilde{\mathbf{y}}_i = \mathbf{y}_i$ for $i \in S$, $\widetilde{\mathbf{y}}_i = \mathbf{0}_{m_\ell}$ for $i \in [n] \setminus S$ and interpret $f' = \{\widehat{\mathbf{y}}_i\}_{i \in [n]}$. $\mathcal{B}$ queries their own oracle on (f', ℓ, ν^0, ν^1) and forwards the result.

In the end, $\mathcal{B}$ outputs the same bit as $\mathcal{A}$. This perfectly simulates $\mathcal{A}$'s view. Note that $\mathcal{B}$ is admissible by the admissibility of $\mathcal{A}$.

Now, in order to prove the security of Construction 2, i.e., Theorem 1, we simply have to prove the following theorem.

Theorem 5. *If* $\mathsf{NMIFE}^{\mathrm{ot}}$ *from Construction 1 is* sta-nh-one-IND *secure and* PRF *is secure, then* NMCFE *from Construction 3 is* sta-nh-one-lab-IND*-secure. In particular, for any* PPT *adversary* $\mathcal{A}$, *there exist* PPT *adversaries* $\mathcal{B}_1$ *and* $\mathcal{B}_2$ *such that*

$$\mathsf{Adv}_{\mathsf{NMCFE},\mathcal{A}}^{\text{sta-nh-one-lab-IND}}(\lambda) \leq 2nq_\ell \cdot \mathsf{Adv}_{\mathsf{PRF},\mathcal{B}_1}(\lambda)$$
$$+ q_\ell \cdot \mathsf{Adv}_{\mathsf{NMIFE}^{\mathrm{ot}},\mathcal{B}_2}^{\text{sta-nh-one-IND}}(\lambda),$$

where q_ℓ *denotes the number of distinct labels queried to* CQEnc *and* $\mathsf{CQKeyGen}$.

Proof. For simplicity, we consider the case where $\mathcal{A}$ only queries CQEnc and $\mathsf{CQKeyGen}$ on label ℓ^* and never on any other label. We show that for PPT adversaries $\mathcal{B}$ and $\mathcal{B}'$, it holds that

$$\mathsf{Adv}_{\mathsf{NMCFE},\mathcal{A}}^{\text{1-sta-nh-one-lab-IND}}(\lambda) \leq 2n \cdot \mathsf{Adv}_{\mathsf{PRF},\mathcal{B}}(\lambda)$$
$$+ \mathsf{Adv}_{\mathsf{NMIFE}^{\mathrm{ot}},\mathcal{B}'}^{\text{sta-nh-one-IND}}(\lambda).$$

By Lemma 1, this concludes the theorem.

Assume that there are h honest clients, which are by definition of the security game known by the challenger before $\mathcal{A}$ can make any queries and let $\mathcal{HS} = \{i_1, \ldots, i_h\}$. For our proof, we utilize a sequence of hybrid games. Denote by $\mathsf{P}(\mathcal{A}, \mathsf{G}_i)$ the probability that $\mathcal{A}$ outputs the correct β in game G_i, i.e., that $\mathcal{A}$ wins G_i. The games are as follows:

G_0: This game corresponds to the case $\beta = 0$.

G_1: This game is the same as G_0, except to answer any encryption or decryption key queries for label $\ell \in \mathcal{L}$, $\mathcal{C}$ sets $\zeta_{i_\eta} = \mathsf{RF}(i_\eta, \ell)$ for all $\eta \in [h]$ with RF being a random function computed on the fly. In other words, the pseudorandom values of the honest clients are changed to truly random values. In Lemma 2, we exhibit a PPT adversary $\mathcal{B}_0$ such that:

$$|\mathsf{P}(\mathcal{A}, G_0) - \mathsf{P}(\mathcal{A}, G_1)| \leq h \cdot \mathsf{Adv}_{\mathsf{PRF}, \mathcal{B}_0}(\lambda).$$

G_2: This game is the same as G_1, except that $\mathcal{C}$ answers all challenge queries with $\beta = 1$, i.e., $\mathsf{ct}_{i,\ell} = \zeta_i + \mathbf{x}_i^1$ as the output of our encryption for all $i \in [n], \ell \in \mathcal{L}$ and $z = \sum_{i \in [n]} \langle \zeta_i, \mathbf{y}_i \rangle - \nu^1$ as the output of the key generation algorithm. In Lemma 3, we exhibit a PPT adversary $\mathcal{B}_1$ such that:

$$|\mathsf{P}(\mathcal{A}, G_0) - \mathsf{P}(\mathcal{A}, G_1)| \leq \mathsf{Adv}_{\mathsf{NMIFE}^{\mathrm{ot}}, \mathcal{B}_1}^{\mathsf{sta\text{-}nh\text{-}one\text{-}IND}}(\lambda).$$

G_3: This game corresponds to the case where $\mathcal{C}$ sets $\beta = 1$ in the original game, i.e., it is the same as G_1 except that ζ_i are generated by PRF instead of RF for all $i \in [n]$. By the same argument as in Lemma 2, it holds that there exists a PPT adversary $\mathcal{B}_2$ such that:

$$|\mathsf{P}(\mathcal{A}, G_2) - \mathsf{P}(\mathcal{A}, G_3)| \leq h \cdot \mathsf{Adv}_{\mathsf{PRF}, \mathcal{B}_2}(\lambda).$$

Putting everything together with the fact that there can be at most n honest clients, we obtain the theorem.

Lemma 2. *There exists a* PPT *adversary* $\mathcal{B}_0$ *such that*

$$|\mathsf{P}(\mathcal{A}, G_0) - \mathsf{P}(\mathcal{A}, G_1)| \leq h \cdot \mathsf{Adv}_{\mathsf{PRF}, \mathcal{B}_0}(\lambda),$$

where h denotes the number of honest clients.

Proof. We prove the lemma by using a hybrid argument over the h honest clients, relying on the security of the PRF. Let $\mathcal{HS} = \{i_1, \ldots, i_h\}$. Without loss of generality, we assume $i_1 < \cdots < i_h$. Define $G_{0,\eta}$, $\eta \in [h]$, to be the same game as $G_{0,\eta-1}$ where $G_{0,0} = G_0$, except to answer any encryption or decryption key queries for label $\ell \in \mathcal{L}$, $\mathcal{C}$ sets $\zeta_{i_\eta} = \mathsf{RF}(i_\eta, \ell)$ with RF being a random function computed on the fly.

It is clear that

$$|\mathsf{P}(\mathcal{A}, G_0) - \mathsf{P}(\mathcal{A}, G_1)| = \sum_{\eta \in [h]} |\mathsf{P}(\mathcal{A}, G_{0,\eta-1}) - \mathsf{P}(\mathcal{A}, G_{0,\eta})| \qquad (9)$$

as $G_{0,0} = G_0$ and $G_1 = G_{0,h}$.

We show that for every $\eta \in [h]$, there exists PPT adversary $\mathcal{B}_{0,\eta}$ against the security of the PRF such that

$$|\mathsf{P}(\mathcal{A}, G_{0,\eta-1}) - \mathsf{P}(\mathcal{A}, G_{0,\eta})| \leq \mathsf{Adv}_{\mathsf{PRF}, \mathcal{B}_{0,\eta}}(\lambda).$$

We build $\mathcal{B}_{0,\eta}$ so that it simulates $G_{0,\eta-1+\beta}$ to $\mathcal{A}$ when interacting with the experiment as described in Definition 11. Given $\mathcal{CS}$ sent by $\mathcal{A}$, run $(\mathsf{pp}, \mathsf{msk} =$

$\{\mathsf{ek}_i\}_{i\in[n]}) \leftarrow \mathsf{Setup}(1^\lambda, m, n)$ and give pp and $\{\mathsf{ek}_i\}_{i\in\mathcal{CS}}$ to $\mathcal{A}$. For all $i \in \mathcal{CS}$, $\mathcal{B}_{0,\eta}$ can answer queries $\mathsf{QEnc}(i, \mathbf{x}_i, \ell)$ and $\mathsf{CQEnc}(i, \mathbf{x}_i^0, \mathbf{x}_i^1, \ell^*)$ using ek_i to calculate $\mathsf{ct}_{i,\ell} = \mathsf{Enc}(\mathsf{ek}_i, \mathbf{x}_i, \ell)$ and $\mathsf{ct}_{i,\ell} = \mathsf{Enc}(\mathsf{ek}_i, \mathbf{x}_i^0, \ell^*)$, respectively. The ciphertext $\mathsf{ct}_{i,\ell}$ is returned to $\mathcal{A}$.

In order to answer (challenge) encryption queries for $i \in \mathcal{HS}$ or (challenge) decryption key queries, $\mathcal{B}_{0,\eta}$ sets ζ_i in the following manner:

- $i < i_\eta$: $\zeta_i = \mathsf{RF}(i, \ell)$,
- $i = i_\eta$: Query own oracle on input (i_η, ℓ) to obtain $\zeta_i = \zeta_{i_\eta}$,
- $i > i_\eta$: $\zeta_i = \mathsf{PRF}(\mathsf{ek}_i, \ell)$.

Any $\mathsf{QEnc}(i, \mathbf{x}_i, \ell)$ and $\mathsf{CQEnc}(i, \mathbf{x}_i^0, \mathbf{x}_i^1, \ell^*)$ queries are answered using this ζ_i, i.e., $\mathcal{B}_{0,\eta}$ returns $\mathsf{ct}_{i,\ell} = (i, \ell, \mathbf{x}_i + \zeta_i \mod q)$ and $\mathsf{ct}_{i,\ell} = (i, \ell, \mathbf{x}_i^0 + \zeta_i \mod q)$, respectively. To answer queries to $\mathsf{QKeyGen}(f_{\mathbf{y}}, \nu, \ell)$ and $\mathsf{CQKeyGen}(f_{\mathbf{y}}, \nu^0, \nu^1, \ell^*)$, first, $\mathcal{B}_{0,\eta}$ calculates $\zeta_i = \mathsf{PRF}(\mathsf{ek}_i, \ell)$ for all $i \in \mathcal{CS}$. Then, for all $i \in \mathcal{HS}$, $\mathcal{B}_{0,\eta}$ calculates ζ_i as above. Identify $f_{\mathbf{y}}$ as $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n)$ and compute $z = \sum_{i\in[n]}\langle\zeta_i, \mathbf{y}_i\rangle - \nu$, where $\nu := \nu^0$ in challenge decryption key queries. Finally, send $\mathsf{dk}_{\mathbf{y},\ell} = (\ell, \mathbf{y}, z)$ to $\mathcal{A}$.

In the end, $\mathcal{B}_{0,\eta}$ outputs the same bit as $\mathcal{A}$. Note that $\mathcal{B}_{0,\eta}$ perfectly simulates $\mathcal{A}$'s view and is admissible with respect to $\mathcal{A}$'s restrictions. Thus, by Equation (9), the claim follows.

Lemma 3. *There exists a* PPT *adversary* $\mathcal{B}_1$ *such that*

$$|\mathsf{P}(\mathcal{A}, \mathsf{G}_0) - \mathsf{P}(\mathcal{A}, \mathsf{G}_1)| \leq \mathsf{Adv}^{\text{sta-nh-one-IND}}_{\mathsf{NMIFE}^{\text{ot}}, \mathcal{B}_1}(\lambda).$$

Proof. We build an adversary $\mathcal{B}_1$ against the sta-nh-one-IND security of $\mathsf{NMIFE}^{\text{ot}}$ as follows.

Given $\mathcal{CS}$ sent by $\mathcal{A}$, $\mathcal{B}_1$ samples $\mathsf{K}_i \leftarrow \{0,1\}^\lambda$ and hands $\{\mathsf{ek}_i\}_{i\in\mathcal{CS}}$ to $\mathcal{A}$. Any encryption queries $\mathsf{QEnc}(i, \mathbf{x}_i, \ell)$ from $\mathcal{A}$ are answered with $\mathsf{ct}_{i,\ell} = (i, \ell, \mathbf{x}_i + \zeta_i \mod q)$, where $\zeta_i = \mathsf{PRF}(\mathsf{ek}_i, \ell)$ if $i \in \mathcal{CS}$ and $\zeta_i = \mathsf{RF}(i, \ell)$ else, where RF denotes a random function. Similarly, all decryption key queries $\mathsf{QKeyGen}(f_{\mathbf{y}}, \nu, \ell)$ are answered by first calculating $\zeta_i = \mathsf{PRF}(\mathsf{ek}_i, \ell)$ if $i \in \mathcal{CS}$ and $\zeta_i = \mathsf{RF}(i, \ell)$ otherwise. Then, $\mathcal{B}_1$ sets $z = \sum_{i\in[n]}\langle\zeta_i, \mathbf{y}_i\rangle - \nu$ by identifying $f_{\mathbf{y}}$ as $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n)$ and returns $\mathsf{dk}_{i,\ell}$ to $\mathcal{A}$.

Whenever $\mathcal{B}_1$ obtains a query $\mathsf{CQEnc}(i, \mathbf{x}_i^0, \mathbf{x}_i^1, \ell^*)$, they query their own encryption oracle on input $(i, \mathbf{x}_i^0, \mathbf{x}_i^1)$. If $i \in \mathcal{HS}$, the received ciphertext is directly forwarded to $\mathcal{A}$. Otherwise, $\mathcal{B}_1$ ignores the output of its oracle and calculates $\mathbf{c}_i = \mathsf{PRF}(\mathsf{ek}_i, \ell^*) + \mathbf{x}_i^0 \mod q$ instead. The ciphertext $\mathsf{ct}_{i,\ell^*} = (i, \ell^*, \mathbf{c}_i)$ is then given to $\mathcal{A}$. Note that in this case, it has to hold that $\mathbf{x}_i^0 = \mathbf{x}_i^1$ in order for $\mathcal{A}$ to be admissible. Thus, $\mathcal{B}_1$ does not make any assumptions about the value of β by choosing to encrypt x_i^0.

Upon receiving a challenge decryption key query $\mathsf{CQKeyGen}(f_{\mathbf{y}}, \nu^0, \nu^1, \ell^*)$, $\mathcal{B}_1$ views $f_{\mathbf{y}}$ as $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n)$ and calculates $z_1 = \sum_{i\in\mathcal{CS}}\langle\zeta_i, \mathbf{y}_i\rangle$. Then, they queries their own decryption key oracle on input $(f_{\widetilde{\mathbf{y}}}, \nu^0, \nu^1)$, where $\widetilde{\mathbf{y}}_i = \mathbf{y}_i$ if $i \in \mathcal{HS}$ and 0 else, to obtain $\mathsf{dk}_{\widetilde{\mathbf{y}}} = (\widetilde{\mathbf{y}}, z_2)$. Send $\mathsf{dk}_{\mathbf{y},\ell^*} := (\ell^*, \mathbf{y}, z_1 + z_2)$ to $\mathcal{A}$.

At the end of their interaction, $\mathcal{B}_1$ outputs the same bit as $\mathcal{A}$. If $\mathcal{A}$ is admissible, then so is $\mathcal{B}_1$. First, $\mathcal{B}_1$ queries each slot $i \in [n]$ of $\mathsf{NMIFE}^{\mathrm{ot}}$ exactly once, if $\mathcal{A}$ poses exactly one challenge query $\mathsf{CQEnc}(i, \mathbf{x}_i^0, \mathbf{x}_i^1, \ell^*)$ and uses only one label ℓ^*. Second, note that for $\mathcal{B}_1$'s decryption key queries, it holds that

$$\sum_{i \in [n]} \langle \mathbf{x}_i^0, \widetilde{\mathbf{y}}_i \rangle + \nu^0 = \sum_{i \in \mathcal{HS}} \langle \mathbf{x}_i^0, \mathbf{y}_i \rangle + \nu^0$$

$$= \sum_{i \in \mathcal{HS}} \langle \mathbf{x}_i^1, \mathbf{y}_i \rangle + \nu^1 = \sum_{i \in [n]} \langle \mathbf{x}_i^1, \widetilde{\mathbf{y}}_i \rangle + \nu^1$$

because $\sum_{i \in \mathcal{CS}} \langle \mathbf{x}_i^0, \mathbf{y}_i \rangle = \sum_{i \in \mathcal{CS}} \langle \mathbf{x}_i^1, \mathbf{y}_i \rangle$ by admissibility of $\mathcal{A}$.

Further, $\mathcal{B}_1$ perfectly simulates $\mathcal{A}$'s view. This is clear for any (challenge) encryption and regular decryption key queries. In case of challenge decryption key queries, note that

$$z_1 + z_2 = \sum_{i \in \mathcal{CS}} \langle \mathsf{PRF}(\mathsf{ek}_i, \ell^*), \mathbf{y}_i \rangle + \sum_{i \in \mathcal{HS}} \langle \zeta_i, \mathbf{y}_i \rangle - \nu^\beta,$$

where ζ_i is a random value. Thus, the lemma follows.

C   Derivation of Formulas

To implement the logistic regression, we have to know what the coefficients of $\mathbf{y}$ are for each element in $\mathbf{x}_i$. For brevity, let $\mathbf{x}_i[m+1] = y_i$. Writing Equation (6) for $j \in [0; m]$ as a linear function for fixed coefficients up to degree 4 yields

$$\theta[j] := \theta[j] + \sum_{i \in [n]} \frac{\alpha}{n} \left(\left(-\frac{1}{2} + (a_1 \theta[0]^3 - a_2 \theta[0]) + y_i \right) \mathbf{x}_i[j] \right.$$

$$+ \sum_{k \in [m]} (-a_2 \theta[k] + 3a_1 \theta[0]^2) \mathbf{x}_i[k] \mathbf{x}_i[j]$$

$$+ \sum_{\substack{k_1,..,k_m \geq 0 \\ k_1 + .. + k_m = 2}} 3a_1 \theta[0] \binom{2}{k_1,..,k_m} \prod_{l \in [m]} \theta[l]^{k_l} \mathbf{x}_i[l]^{k_l} \mathbf{x}_i[j]$$

$$+ \sum_{\substack{k_1,..,k_m \geq 0 \\ k_1 + .. + k_m = 3}} 3a_1 \binom{3}{k_1,..,k_m} \prod_{l \in [m]} \theta[l]^{k_l} \mathbf{x}_i[l]^{k_l} \mathbf{x}_i[j] \right),$$

where

$$\binom{n}{k_1, k_2, .., k_m} = \frac{n!}{k_1! k_2! \cdots k_m!}$$

is the multinomial coefficient. This way, we have a formula for $\mathbf{y}$ for each entry in the extended $\widetilde{\mathbf{x}}_i$.