MIOPE: A Modular framework for Input and Output Privacy in Ensemble Inference

Kyrian Maat[1], Gareth T. Davies[2], Zoltán Ádám Mann[3], Joppe W. Bos[2] and Francesco Regazzoni[1,4]

[1] University of Amsterdam, Amsterdam, Netherlands
[2] NXP Semiconductors, Leuven, Belgium
[3] University of Münster, Institut für Informatik, Münster, Germany
[4] Università della Svizzera italiana, Faculty of Informatics, Lugano-Viganello, Switzerland

Abstract. We introduce a simple yet novel framework for privacy-preserving machine learning inference that allows a client to query multiple models without a trusted third party aggregator by leveraging homomorphically encrypted model evaluation and multi-party computation. This setting allows for dispersed training of models such that a client can query each separately, and aggregate the results of this 'ensemble inference'; this avoids the data leakage inherent to techniques that train collectively such as federated learning. Our framework, which we call MIOPE, allows data providers to keep the training phase local to provide tighter control over these models, and additionally provides the benefit of easily retraining to improve inference of the ensemble. MIOPE uses homomorphic encryption to keep the querying client's data private and multi-party computation to hide the individual model outputs. We illustrate the design and trade-offs of input- and output-hiding ensemble inference as provided by MIOPE and compare performance to a centralized approach. We evaluate our approach with a standard dataset and various regression models and observe that the MIOPE framework can lead to accuracy scores that are only marginally lower than centralized learning. The modular design of our approach allows the system to adapt to new data, better models, or security requirements of the involved parties.

Keywords: Privacy-enhanced Machine Learning · Fully Homomorphic Encryption · Security for Machine Learning

1 Introduction

Machine Learning has found increasing success in finding patterns that humans are not able to decipher, such as in diseases or DNA [GST+99, AAR+19]. However, to correctly find and act upon these patterns requires a large amount of high-quality data for training [DDS+09]: in many settings collecting such data at a central point is infeasible due to legal and ethical safeguards that protect individual privacy or commercial interests. Even if one or more models can be trained, if the input to the model(s) is very sensitive then similar legal/ethical boundaries may prohibit cleartext (transmission and) processing of queries (inputs) or availability of the resulting insights (outputs) during inference [GLMS23].

This research has received funding from the European Union's Horizon Research and Innovation Actions under the Grant Agreement No. 101095717 (SECURED).

E-mail: k.maat@uva.nl (Kyrian Maat), garethThomas.davies@nxp.com (Gareth T. Davies), zoltan.mann@uni-muenster.de (Zoltán Ádám Mann), joppe.bos@nxp.com (Joppe W. Bos), f.regazzoni@uva.nl (Francesco Regazzoni)

In scenarios with large volumes of horizontally partitioned data that cannot be centralized, there are two distinct avenues: training a joint model on the datasets in a way that provides some degree of data input privacy or keeping the data in their silos and performing inference over an ensemble of locally trained models. The first approach is often realized via Federated Learning (FL) [KMR15, KMY+16], which allows for multiple data owners to collectively train a single model by sharing model weights of local models to contribute to a single centralized model. Multiple rounds of communication are required to reach convergence but the final result is built upon the data of the local models, and this can achieve similar results to a non-privacy preserving centralized training [ZLL+18].

Federated Learning is a fast-developing and maturing field and has many different variants that can address specific problems. Asynchronous variants [CNSR20, XQXG23] ease the burden of coordination and can more easily handle drop-outs or slow participants. Decentralized Federated Learning [HBJ18, ZYWL25] removes the need of a central server by involving the participants with the aggregation. Secure aggregation [BIK+17, BBG+20] and differential privacy (DP) [Dwo06] can also be utilized to reduce how much a singular party can learn from the updates it sees. For many scenarios, a combination of these techniques will be strong enough to provide privacy guarantees.

However there are scenarios in which, even with the aforementioned advances and variants, FL does not fit. Consider for example a network of mental health clinics that want to jointly assess a patient's risk of a mental health crisis [SGU25]. Each clinic has locally trained a risk model on its own patient data, encapsulating demographic attributes, history of treatment and different symptoms that can differ on a per population basis. Sharing the raw data is not allowed as they contain sensitive personal information, but a clinic's local model in and of itself sensitive. Additionally the training phase of Federated Learning where model weights or model updates are shared introduce attack vectors for gradient inversion attacks [GBDM20, YZH22], where an adversary that obtains these model updates can reconstruct sensitive training data and expose individual records without obtaining the raw data. Sharing gradients is therefore also not a justifiable option for legal teams, which would exclude one of the most common FL settings regardless of how well the data itself are handled. Secure aggregation and the usage of DP could reduce this risk, but this adds complexity and non-trivial accuracy penalties in the case of DP, which are trade-offs that contradict the need for highly-accurate risk-averse institutions in sensitive domains. Besides there still remains the question of how a jointly trained model that all participants have contributed to is handled in terms of ownership and auditability. Who and how can the model be utilized among participants and how can it be retrained or audited if needed, in cases of incorrect data or poisoning attacks?

No singular variant of FL fully resolves all these different areas of privacy, governance and regulatory concerns and in many cases—such as for personally identifiable health care data—these privacy and security risks may mean that FL—even techniques such as decentralized FL [HBJ18]—would be seen as too risky in the eyes of regulation enforcers: the consequence is that the data remains in silos.

Ensemble inference[1]—inference over multiple trained local models in an ensemble, where the model outputs are aggregated or summed on the entity querying the models—can be a useful approach in these privacy-sensitive settings [ZDW13, ZLZ23, KTBG20]. Participants are free to update or replace their models independently. However, a naive implementation of ensemble inference exposes the individual model outputs to the client, which in fact *increases* the risk of model extraction or membership inference compared to federated learning.

What we want to obtain are the advantages of ensemble inference in terms of model autonomy and local training, while ensuring the client only sees aggregated outputs, not

[1]We use this terminology to distinguish from the use of an ensemble of models to collectively and iteratively *train* a better model using boosting or bagging, often referred to as *ensemble learning* [Efr92].

individual model outputs. At the same time, we also want to protect the client's query, as none of the clinics should learn about the sensitive information of the client. Ideally, we would not want to involve any participants beyond the client and clinics, no third parties that have to be agreed upon and handle processing of client data, model updates, or similar. We position MIOPE at that particular set-up, creating a a modular privacy-preserving machine learning inference framework that enables analytics on data that is kept—by policy, legal framework or simply due to risk adversity—in distinct databases and cannot be shared; models are trained locally on these databases. MIOPE keeps the advantages of flexibility from ensemble inference and provides input and output privacy, without a trusted third party. Each model owner evaluates homomorphically encrypted client queries and masks its output with values that cancel upon aggregation, so the client learns only the ensemble result, not individual model outputs[2]. To our knowledge, MIOPE is the first framework that combines homomorphic encryption for query input with MPC-style output masking for model output in a model-agnostic ensemble inference setting, utilizing so called 'output-hiding ensemble inference'.

We demonstrate the utility of our approach by providing an implementation for tree-based inference that uses public-key (fully) homomorphic encryption (FHE) by combining the CKKS [CKKS17] and FHEW [DM15] schemes in the `OpenFHE` library [BAB+22], the NIST randomness beacon [Nat25] for mask generation, and a simple method to sum all masks to zero in the query output aggregation component. We evaluate the approach on the California housing dataset [PB97] and show that the accuracy of the encrypted ensemble is only marginally lower than centralised training, with the flexibility to mix model types and update individual models independently.

1.1 Brief Related Work

The closest related literature to ours is that of Giacomelli et al. [GJK+18], which also aims to provide input privacy and output privacy for querying an ensemble of tree-based models using homomorphic encryption. Their framework fixes a machine learning structure (random forests), and requires that the model owners encrypt models for storage on a dedicated processing entity that also performs output aggregation. This dedicated server is a single point of failure and introduces a collusion risk between server and client that MIOPE avoids by design. Removing the central server requires a new mechanism for masks to cancel across independent owners which MIOPE provides.

A related line of work concerns private inference queries on tree-based models, and this is often referred to as private decision tree evaluation [WFNL16, TMZC17, KNL+19]. In such papers, the querying client and the model-holding server engage in a multi-round cryptographic protocol, using multi-party computation or homomorphic encryption or a combination of the two, to evaluate the tree-based model without leaking the client's input or the model's output to the server. While these dedicated interactive protocols can provide efficiency gains compared to our approach of generically using FHE for this step, this class of solutions requires significant setup costs and lacks flexibility, and furthermore many solutions leak the model's topology/structure to the querying client. MIOPE achieves multiple model output's privacy non-interactively with owners operating independently after a one-time setup, not requiring further communication between model owners or a server. As such, our work is more closely aligned with non-interactive private decision tree evaluation [TBK20, ALR+22, CDPP22, MNLK23], and the modular nature of our framework would allow any of these constructions to be used in the encrypted inference component.

[2]The data leakage is not necessarily zero: following much of the secure multi-party computation (MPC) literature, it is only what can be calculated from the aggregated output. We elaborate on this in detail later.

1.2 MIOPE

In a nutshell, our system requires a set of trained machine learning models that perform the same task, e.g. a regression task that predicts the risk of a specific type of cancer based on the environmental and/or genomic data of a patient. The client wants to keep their input query private, and the model owners want to hide their model architecture and their individual model outputs in order to reduce the efficacy of black-box attacks [SSSS17].

Our framework specifically aims to provide model updatability (models can be retrained at any time) and model agnosticity (any model owner can employ any model architecture) while still allowing a client to query over data spread over multiple entities.

The main idea behind MIOPE and output-hiding ensemble inference is improving the flexibility of the participants and models involved. The participants can train their model whenever they like and do not depend on other models in the system for training and participating in the system allowing them all to train simultaneously and separately from each other. An added benefit of this design decision is that the MIOPE system has strong adaptability and resilience for participants in the system, as corrupted models or incorrect models can easily be removed, and new models can be added to the ensemble of models. As long as the input and output of the system is agreed upon, any model can be added to the MIOPE system and will not need any retraining of a global model that might be utilized in a centralized or federated learning setting. One can also consider the general model-agnosticism that MIOPE allows another perk of adaptability, as the models can be heterogeneous in form if they adhere to the pre-agreed output requirements from the system. Similarly, the resilience of the MIOPE system also benefits from the plug-and-play nature of the model participants. A faulty mode, such as a wrong calibration in measuring, or a rogue model owner can easily be removed from the pool of queried models, whereas alternative approaches, such as FL, will require some retraining of the central model to remedy this problem. Additionally, because the model owners are free to bring their own local models, training time is amortized over the multiple model owners and the system as a whole can start running earlier than systems with communication during the training phase. This implies a shorter time to the first inference of the system than a setting where training has to be done together. However, conversely, due to the set-up of MIOPE, the time for a single query is higher than Federated Learning. A single model being queried with a single communication round is cheaper than the MIOPE alternative of multiple model owners that each have to communicate with the client, making the weakest link the bottleneck of the inference.

Contributions. In this work we provide:

- a modular and model-agnostic framework output-hiding encrypted ensemble inference, combining FHE for input privacy with an MPC-style output masking for model output privacy, without a trusted third party for compute or processing.
- A concrete instantiation for inference on tree-based models in `OpenFHE` using the fully homomorphic schemes CKKS [CKKS17] and FHEW [DM15] via scheme-switching (which may be of independent interest);
- an end-to-end demonstration of our privacy-preserving inference pipeline using regressors trained with the California housing dataset [PB97], supporting heterogeneous models.

2 Preliminaries

In this section we define some standard definitions and concepts that we will use in the MIOPE framework.

2.1 Fully Homomorphic Encryption

Definition 1. *A public-key (fully) homomorphic encryption scheme* $\mathsf{HE} = \{\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec}\}$ *is defined by the following algorithms:*

- *Key generation:* $(\mathsf{pk}, \mathsf{sk}, \mathsf{evk}) \leftarrow \mathsf{KeyGen}(\lambda, \mathsf{params})$. *On input a security parameter* λ *and other parameters* params*,* KeyGen *outputs an encryption key* pk*, a decryption key* sk *and an evaluation key* evk*.*
- *Encryption:* $\mathsf{C} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{m})$. *On input an encryption key* pk *and a plaintext message* m*,* Enc *outputs a ciphertext* C*.*
- *Evaluation:* $\mathsf{C}' \leftarrow \mathsf{Eval}(\mathsf{evk}, \mathsf{C}, \mathsf{f})$. *On input an evaluation key* evk*, a ciphertext* C *and a function* f*,* Eval *outputs a new ciphertext* C'*.*
- *Decryption:* $\mathsf{m} \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{C})$. *On input a decryption key* sk *and a ciphertext* C*,* Dec *outputs a plaintext message* m *(or an error symbol such as* $\perp$*).*

The function $\mathsf{f} : \mathcal{D} \mapsto \mathcal{R}$ belongs to some viable set of functions that the scheme supports, hence this function set can be seen as a parameter of HE. Our syntax uses a generic argument params to reflect that in general it is necessary for the entity to know at least some details about the supported functions f, for example the multiplicative circuit depth of f.

All known FHE schemes, including the first one based on lattices and proposed by Gentry [Gen09], depend on the addition of randomness—called *noise* or *error*—to the ciphertext. Each HE evaluation also adds noise to the ciphertext (growing slowly for addition, and very quickly for multiplication). However, there is an upper bound on the amount of noise tolerated, and if that bound is exceeded, decryption will result in failure. To reduce noise, the scheme includes an auxiliary process called *bootstrapping* that homomorphically evaluates the decryption function with an encrypted secret key (i.e. resetting the noise to that of a single fresh encryption). Bootstrapping is usually a very expensive procedure since it requires homomorphic evaluation of what is often a high complexity function: applications often therefore define the cryptographic parameters to enable evaluation of a class of functions without ever needing to bootstrap.

HE schemes are often well suited to specific tasks due to their underlying structure, for example BGV [BGV11] and BFV [FV12, Bra12] are suited to precise computations on integers, CKKS [CKKS17] is an approximate scheme which is more suited to floating point operations where small accuracy loss is tolerable, while TFHE [CGGI16, CGGI20] and FHEW [DM15] are examples of FHE schemes working on binary circuits and therefore are well suited to functions that can be represented as (small) look-up tables (these schemes also support relatively fast bootstrapping).

In machine learning inference it is often necessary to perform a combination of operations, for example SIMD multiplication by weights for a neural network linear layer or a tree-based model's tree output pooling function and non-linear operations for activation functions or the comparisons made in decision nodes. In the encrypted domain it is therefore desirable to transform ciphertexts directly from one scheme to another: this process of *scheme switching* [BGGJ20, LHH+21] is essentially a more involved form of bootstrapping where the target scheme's decryption operation is homomorphically evaluated.

HE schemes are implemented in a variety of open-source libraries. The `OpenFHE` [BAB+22, BBB+22] library is used in this work and carries implementations of all of the schemes mentioned above. Other libraries exist, including `Concrete` [Zam22b] (and its ML wrapper `Concrete ML` [Zam22a]) which only supports the TFHE scheme.

2.2 Encrypted Inference for Machine Learning

Our work considers outsourced private inference, meaning that a machine learning model is evaluated entirely in the encrypted domain by a service provider (hereafter referred to as the model owner, but in practice these entities could be separated). Using homomorphic encryption makes it possible that the inference query inputs and outputs are not available in plaintext to this model owner [MWCB24].

A model owner is in possession of a trained model M that maps input features represented as a vector $\mathtt{qry}$ to some output space. To prepare for encrypted inference queries, the model owner represents M as a function f that is appropriate for scheme HE, extracts the necessary parameters params and sends them to potential querying clients. A client $\mathtt{C}$ can then run $(\mathsf{pk}, \mathsf{sk}, \mathsf{evk}) \leftarrow \mathsf{KeyGen}(\lambda, \mathsf{params})$ and send evk to the model owner.

In the online inference phase, $\mathtt{C}$ encrypts their input feature vector with the encryption key $\mathsf{C} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathtt{qry})$, and sends C to the model owner. The model owner then homomorphically evaluates their model on the input query by doing $\mathsf{C}' \leftarrow \mathsf{Eval}(\mathsf{evk}, \mathsf{C}, \mathsf{f})$, and sends C' back to $\mathtt{C}$. The client can decrypt the received message to get $f(\mathtt{qry}) \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{C}')$ as desired.

Note that in this description the query is encrypted and the result is decrypted by the client, and therefore the encryption key pk does not need to be made public. Later on for the output masking step we will need to let the model owner encrypt random values and homomorphically add them to the model output, therefore pk will need to be transmitted alongside evk. The HE implementation in `OpenFHE` supports this 'public-key encryption' variant of HE, whereas it is not supported by `Concrete ML` (only a symmetric mode is available).

2.3 Multi-Party Computation

Secure multi-party computation (MPC) enables a group of parties to perform collaborative computations. The system consists of *input parties* that provide inputs, *computing parties* that actually engage in the computation phase, and *output parties* that receive output. This abstraction captures settings where these sets could be fully disjoint, or where they could fully overlap. Defining security for MPC can be a challenging exercise but generally at a minimum we expect *input privacy* for at least some of the input parties, even for adversaries that see intermediate computations and the output. Participating entities can be *honest-but-curious* where they perform their role according to the protocol specification yet try to learn as much additional information as possible, or *malicious* where they actively manipulate messages to reveal additional information or force other parties to accept an incorrect result.

A formal syntax is similarly difficult to provide in full generality, but for our purposes it is not necessary since the separation of parties is straightforward and the functions being collaboratively computed are simple.

The MPC component of our pipeline is a modular block that can adapt to the required implementation environment and threat model, for instance to tolerate malicious behavior or participants that might drop out during execution.

2.4 Randomness Beacons

Our instantiation requires a set of parties to use a randomness source for (use as a seed for deterministically generating) structured and correlated outputs. The properties we require from this randomness source are *unpredictability* (predicting the next output is hard even given prior outputs), *unbiasedness* (outputs are statistically close to uniform) and *public access* (all parties can easily access its outputs). Rabin [Rab83] first formalized the concept of a verifiable public randomness beacon. Suggestions for instantiation have

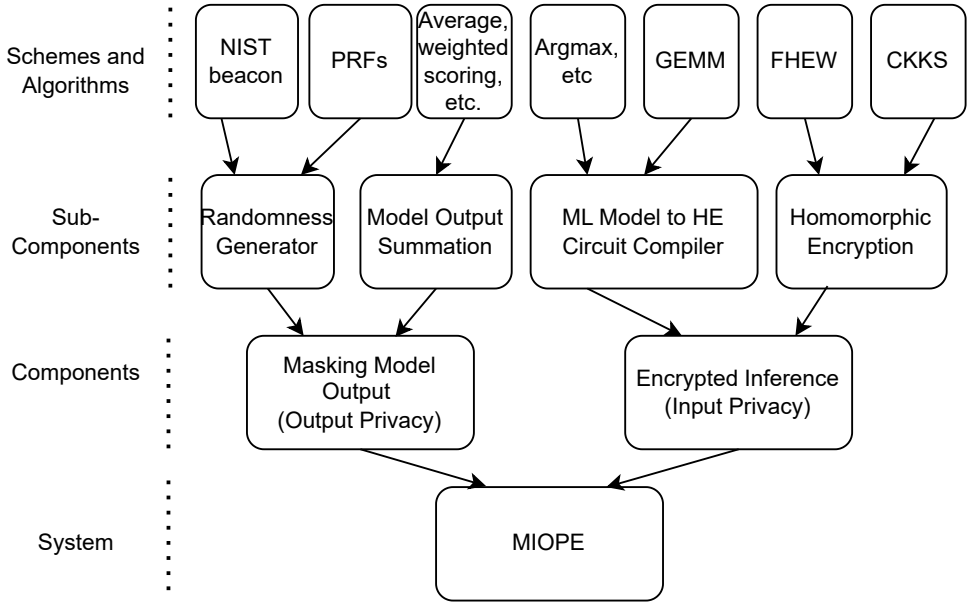

Figure 1: An illustration of the different components that are present in the realisation of the MIOPE system as a top-down approach. The top row represents our choices for our instantiation the modular sub-components in the second row.

included financial data [CH10] and the Bitcoin blockchain [BCG15]. Demand is sufficient for NIST to run their own randomness beacon [Nat25] and we will use this service in our implementation.

We will additionally require a method for our set of parties to *extract* randomness from this source using a pseudorandom function such as HMAC-PRF or AES-CBC with zero strings for key and IV.

3 System design of MIOPE

In this section we define the constituent components of MIOPE. A graphical representation of the MIOPE system is given in Fig. 1. The core driving idea behind output-hiding ensemble inference is the usage of multiple models that are trained in a local manner by their respective data owners and thus isolated from each other to inherently gain privacy and security on the models. When performing inference, all models are queried and the outputs are combined via an output summation strategy to obtain the final result. Two building blocks are required for the input and output privacy of MIOPE, encrypted inference on the desired ML model by converting trained machine learning models to circuits (in Section 3.1) and model output masking (in Section 3.2). We define our threat model associated with MIOPE in Section 3.3.

3.1 Encrypted Inference in MIOPE

Encrypted inference for MIOPE requires homomorphic encryption (as introduced and defined in Section 2.2) to be applied over the multiple models.

The usage of encrypted inference will achieve input privacy for the client, as the features used are homomorphically encrypted and the output is never able to be observed by the model owner. For each model owner, the client runs KeyGen once and sends the evaluation key. The model owner only communicates to the client their multiplicative depth of the HE circuit, which does not reveal their model architecture. If the model owner internally uses scheme-switching, this is also not revealed to the client. The model owners can choose their HE scheme independently from the client or other model owners to optimize for their own model architecture[3]. The only shared requirement are that the model outputs are normalized to a common space (e.g. $\{0.0, 0.1, \ldots, 100.0\}$ for regressor models), which is also necessary for output masking described in Section 3.2.

3.2 Model Output Masking

Encrypted Inference has given the system input privacy: none of the inputs that are given by the user are ever revealed to the model owners or any eavesdroppers. However, the client will decrypt all the model outputs locally. This means that the individual model output privacy is not guaranteed, allowing (adversarial) clients to potentially launch black-box attacks if it wishes to glean insights into the training data of the individual data silos.

MIOPE therefore incorporates model output masking to the encrypted outputs of each model to provide input and output privacy whilst providing the correct end results to the client. To perform model output masking, the outputs obtained from the encrypted inference need to be masked with random values, and since we do not have a dedicated and active trusted third party in our system we generically require a randomness source that is accessible by all model owners. Public randomness beacons—that precisely provide the functionality that we require—were introduced in Section 2.4.

To combine the model outputs, different strategies can be utilized depending on the data used in the model and the goal of the model, such as a classification, image recognition, or regression task.

The property we need from this procedure is that the model owners can take the outputs from a randomness source and turn them into values that 'cancel out' (for some definition that corresponds to the mathematical structure of the masks) when the client attempts to reconstruct a single output from the ensemble. As an example approach we will later describe the most straightforward solution for regressor models, namely masks that together sum to zero: summation of masked model outputs and division by the number of models then leads to the average of the regressor outputs.

The process for a model owner is therefore to homomorphically evaluate the model on the ciphertext query, retrieve the output of the randomness source that corresponds to the query, then to calculate the appropriate mask from this randomness source output[4]. For example, the mask generation process could take into account the time at which the client made their query and a client identifier/pseudonym. A (potentially non-interactive) protocol will then occur to turn these values into something that can (blindly) remove all masks when the client performs reconstruction. This process could be performed by the client on all of the encrypted-and-masked model outputs, or the client could first decrypt all masked model outputs and then perform an appropriate operation.

A generic syntax for a masking scheme that can be used for output-hiding ensemble inference is provided in Definition 2.

[3]As a reminder, the input to key generation is the multiplicative depth of the HE circuit being evaluated (this is represented in generality by params in Def. 1). Therefore if all model owners use models of the same depth then the client could run KeyGen just once and send the same evaluation key material to all model owners, but this is not required by our approach.

[4]The model-output privacy-preserving techniques will be applied to the output of the circuit provided by the inference on the input, and then the technique to protect the model outputs can be considered an extra gate or circuit applied at the end.

Definition 2 (Masking of Model Outputs)**.** *Let* $\mathsf{HE}_0 = \{\mathsf{KeyGen}_0, \mathsf{Enc}_0, \mathsf{Eval}_0, \mathsf{Dec}_0\}, \ldots,$ $\mathsf{HE}_{\mathsf{N}-1} = \{\mathsf{KeyGen}_{\mathsf{N}-1}, \mathsf{Enc}_{\mathsf{N}-1}, \mathsf{Eval}_{\mathsf{N}-1}, \mathsf{Dec}_{\mathsf{N}-1}\}$ *be* N *homomorphic encryption schemes. A client* $\mathtt{C}$ *has input* $\mathtt{qry}$ *and sends it in encrypted form to* N *model owners* $(M_0, \ldots, M_{\mathsf{N}-1})$*. After homomorphic processing the encrypted model output is* $c_i = \mathsf{Enc}_i(\mathsf{pk}_i, y_i)$ *where* $y_i = f_i(\mathtt{qry})$ *is the plaintext output of model* f_i*. The model owners then engage in a protocol* MASK.GET *such that each* M_i *receives or calculates a mask* r_i *and applies it to the encrypted output in a procedure* MASK.APPLY*, i.e. performs* $z_i = \mathsf{Eval}_i(\mathsf{evk}_i, c_i \bigodot r_i)$ *for some ciphertext operation* $\bigodot$ *that corresponds to a plaintext operation* $\bigotimes$*. The client then performs output reconstruction* MASK.REMOVEALL *by either*

- *doing* $\mathtt{agg.fn}(z_0, \ldots, z_{\mathsf{N}-1})$ *over the encrypted responses to get one or more ciphertexts that it can decrypt and then combine into a single* out*, or*
- *decrypting each ciphertext* z_i *using* sk_i *and then performing a plaintext space operation* $\mathsf{out} = \mathtt{agg.fn}'(y_0 \bigoplus r_0, \ldots, y_{\mathsf{N}-1} \bigoplus r_{\mathsf{N}-1})$*.*

Protocol $\{\mathsf{MASK.GET}, \mathsf{MASK.APPLY}, \mathsf{MASK.REMOVEALL}\}$ *is said to be a correct if the single output* out *is the same as the aggregation procedure applied in plain to the model outputs* $y_0, \ldots, y_{\mathsf{N}-1}$*.*

Although this definition is cumbersome, its purpose is straightforward: random masks hide individual outputs and must be collectively removable.

To instantiate this with output summation and masks that sum to zero, the model owners are given identifiers and placed into pairs by the querying client. The model owners then calculate a random mask per pair, where one of the model owners will add the calculated value to their output and the other model owner will subtract that same value from their model's output. More formally, the aggregation function averages the model responses of a regressor with output space $\{0.0, \ldots, 100.0\}$. We construct pairs of random masks—that are also elements of $\{0.0, \ldots, 100.0\}$—that alternate their sign, since for $z_1 = y_1 + r$ and $z_2 = y_2 - r$ the average (over the integers) of the z_i values is the same as for the y_i values[5]. We cannot use arithmetic mod100 for addition, in the event that one value gets pushed across the boundary by addition and the other does not[6]. We can solve this by extending the function output space in the positive and negative direction to ensure that wraparound does not occur. This creates a new masked range $\{-100, \ldots, 200\}$ in which average aggregation will work correctly.

If a different operation than summing is required, for example using Argmax applied to classification models or incorporating contribution evaluation then if it is feasible to construct the corresponding masking scheme then this can be used in our modular framework in a straightforward manner.

Note that Definition 2 is stated with the presence of HE in mind, but output-hiding ensemble inference can also be used without encryption if input privacy is not required. In that case, encryption and decryption reduce to the identity function, and masking is applied in plaintext.

3.3 Security Goals

So far we have introduced the building blocks that can be used to provide functionality that we have designed MIOPE to achieve, namely inference to an ensemble of models. To complete the threat model we would need to precisely define adversarial capabilities but this is not the goal of our approach: we wish to provide a framework that can use

[5]This idea extends to groups of model owners that collectively append randomness such that the sum (and therefore the average) of the output values is the same with or without masking.

[6]If $y_1 = 90, y_2 = 92$ and $r = 20$ then $z_1 = 10, z_2 = 72$ and therefore the average of the z values is 41 rather than the correct value of 91. It's possible to deal with this in plaintext, but in the encrypted domain the entity adding the masks will not be aware of whether wraparound has occurred or not.

building blocks that are sufficient for a fixed set of assumptions on the adversaries in the system. We briefly and informally present the privacy goals of MIOPE, where we assume a threat model of honest-but-curious with no collusion, with more details given in the discussion in Section 7.1 for stronger adversaries. Our construction relies on the security of the underlying primitives and we therefore assume standard security properties for FHE and MPC.

For privacy, there are two distinct privacy goals that have to be achieved in input- and output-hiding ensemble inference. We define input privacy as such: An encrypted inference scheme provides input privacy if the model owners performing inference on the input data `qry` learn no information about the input data or the model output beyond the fact that a query has occurred, the view of any model owner is computationally indistinguishable from a simulation with the query. We also define output privacy as: An output-hiding ensemble inference scheme provides output privacy if the client does not learn information about an individual model output y_i beyond the combined aggregated output of the ensemble $\texttt{agg.fn}(y_0, \ldots, y_{N-1})$.

In our instantiation of MIOPE in Section 4 queries are encrypted under HE, meaning no secure communication channels are required, so parties can freely observe network traffic without gaining information. We can also allow collusions among model owners, but not between the client and model owner, as they can work as a colluding pair to cancel the masking in Definition 2 and expose a paired model's output. We consider collusion in more detail in Section 7.1.

3.4 MIOPE: A Modular Framework

With this generic setup for output-hiding ensemble inference we have a bottom-up approach allowing selection of algorithms and schemes for performing the collective inference. We show this in Fig. 1 where the top-row is an instantiation of the algorithms and schemes used in this paper, which will be clarified in Section 4. If these components fulfill the requirements of the underlying components, the system can be used to create an ensemble inference setting with privacy for model inputs and outputs.

4 Output-Hiding Ensemble Inference on Tree-Based Models

In Section 3 we have defined our framework for input- and output-private encrypted inference in ensembles settings. We will now specifically focus on the usage of tree-based models held by multiple model owners in the rest of this paper, but as a reminder our generic framework does not require any specific machine learning architecture and this is just one example instantiation. Tree-based ML models have shown good performance on tabular and clinical data [ZZC+19, MKV+23], which closely fits with participants that can utilize MIOPE. Regardless of whether a single-tree model or a forest (trained for example using gradient boosting) is chosen, these models require only small modifications to work with encrypted inference.

4.1 Encrypted Inference on Tree-Based Models

To ensure the encrypted inference works on the tree-based models, we have to create or use a technique to allow homomorphic operations to work on the tree-based models. Generally, tree-based models consist of nodes that contain a decision or threshold that needs to be checked relative to an input feature, to traverse the tree and ultimately obtain a prediction for a given input. However, if a computing party wants to traverse the tree in a secure and encrypted fashion, they cannot take any decision in the tree itself, since this would

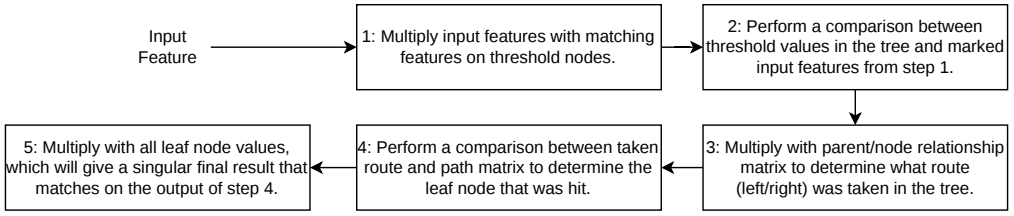

Figure 2: Graphical representation of inference for a tree-based ML model using the GEMM (GEneric Matrix Multiplication) framework, which is used in frameworks such as the `Hummingbird` library [NIS20].

reveal the path taken (and thus partial information about the input features) and, as the model is owned by the computing party, they would learn the output of the query. This necessitates a way to mask the path being traveled from the computing party, which can be done by transforming the tree-based model into a series of linear and non-linear matrix operations.

One possible solution is the GEMM approach for tree-based models which is used to represent the tree-based models in tensor format, as performed by the `Hummingbird` library [NIS20, NSY+20].

4.1.1 ML model to HE Circuit Conversion for Tree-Based models

Concretely, the GEMM inference method transforms the tree-based inference into tensors which are constructed based on the tree model's characteristics. The technique in essence simulates going over every path in the tree with the input data and localizing with tensor operations over encrypted data which final leaf node is hit, yielding only the final leaf value as encrypted output if GEMM is used in combination with homomorphic encryption.

For tree-based models, the GEMM algorithm is a five-step process, see Fig. 2. Let us assume in this description that all steps are performed via homomorphic encryption. First, a multiplication is performed to setup a tensor for feature values which are then compared in the second step with the threshold values of the model. Another multiplication is performed afterward for determining the steps to take to a certain node, after which one more comparison is made to see if a specific path matches. Lastly, a multiplication is performed to obtain the value of the leaf that was matched in the fourth step. In total, the GEMM algorithm consists of 3 tensor multiplications and 2 tensor comparisons. To enable encrypted inference over tensors, multiplication and comparison via the FHE scheme thus need to be supported to prevent the computing party from learning any intermediate outputs.

We implement GEMM in `OpenFHE` [BAB+22] instead of `Concrete ML` [Zam22a], as `OpenFHE` offers broader configurability and, crucially, supports public-key operations which we require for model output masking. While `Concrete ML` implements GEMM for tree-based models [FSB+24], it does not support PKE.

4.1.2 Implementation in OpenFHE

We require both multiplication and comparison operations in our computation under homomorphic encryption, which can be problematic as most FHE schemes are usually stronger on specific types of operations. Multiplication, for example, is cheap in the integer schemes of FHE, but these schemes usually do not cheaply perform comparisons or approximations. In contrast, binary schemes in FHE have cheaper comparisons but tend to have costlier multiplications. `OpenFHE` allows for implementations in the integer

schemes, binary schemes, and also allows for scheme-switching. Integer schemes would require multiplication by approximations of high-degree polynomials to mimic comparisons, and pure binary scheme implementations in `OpenFHE` are less well-documented and lose performance on multiplications. Because machine learning inference on tree-based models has a combination of both non-linear, such as comparisons, and linear operations, we decided to use scheme switching as we have approximately an even split in operations between linear and non-linear.

Our implementation for tree-based encrypted inference starts in the integer scheme CKKS to apply the first multiplication, and scheme-switches to the binary scheme FHEW afterwards to apply the comparison operation. We scheme-switch back to CKKS afterward and repeat for the 3rd and 4th operations in CKKS and FHEW, respectively. The last multiplication with the leaf values is again done in CKKS which is also suitable to allow any additional masking or summation to be performed in CKKS after the GEMM approach has finished.

The GEMM algorithm can be applied to any tree-based model, which allows one to do inference on the features and get a result back per tree. Specifically we stress per tree, as the models that have forests, such as the gradient-boosting and random forest approaches, require a model summation to sum up the leaf values from all the individual trees. For random forests specifically, one also needs to divide by the number of trees used in the summation. Other tree-based models that we do not mention above could require modifications depending on the nature of the tree-based model.

The final resultant outputs captured in the encrypted ciphertext(s) are masked by the output model masking technique from Section 3.2 and then sent to the client that aggregates the outputs. In our evaluation we focused on regressor models with a simple strategy of summing (i.e., no weighted scores), and the syntax used in the examples reflects this. The same encrypted GEMM pipeline and masking mechanism can be applied to tree-based classifiers with an appropriate aggregation method, which we further discuss in Section 7.2.

4.2 Our Instantiation of MIOPE

Our instantiation of MIOPE for this paper is shown in the top row of Fig. 1 which uses GEMM for encrypted inference with scheme-switching CKKS and FHEW, a PRF seeded by the NIST beacon for mask generation and simple averaging as the output summation strategy used. Any of these components can be exchanged for alternatives as long as the above components are satisfied (Section 3.4). The only requirement among owners is a shared output normalization space; everything else is local to each owner in terms of their choices.

Fig. 3 showcases an example end-to-end instantiation of the MIOPE architecture where data is first encrypted via the keys that were exchanged in the PKE between client-model pairs. The data is encrypted and after that is moved to the model domain where encrypted inference is performed via a function that suits the specific ML model. After the model outputs are created, the models apply masks generated via the randomness that was acquired prior via the NIST beacon and PRF. The masked outputs are sent back to the client who can decrypt each of the outputs without revealing the exact output of the individual models as they still have masks applied to them. Only after summing the model outputs via model output summation does the client obtain the correct final result.

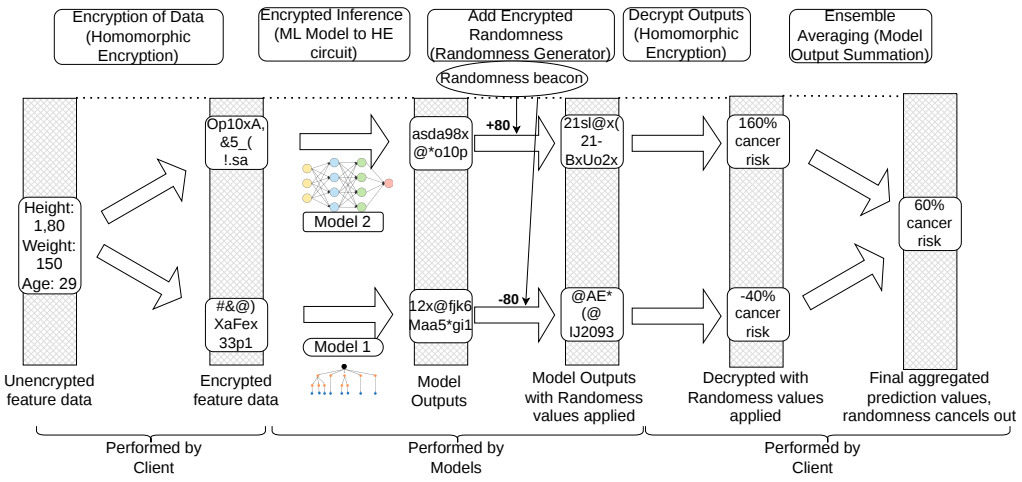

Figure 3: An example of the MIOPE framework to calculate a risk score over two models for cancer given a feature vector. Starting with the client encrypting their feature vector before sending their data to the models to query, which perform encrypted inference and add randomness to their model outputs. The results can be safely decrypted and summed on the client side.

5 Evaluation Test Case: Output-Hiding Ensemble Inference on California Housing dataset

Our evaluation uses the California housing dataset [PB97], a widely used dataset for demonstrating the effectiveness of regression tasks.[7] The dataset consists of 20,640 entries. We use the dataset to train $\mathsf{N} = X$ Regressor models to determine the median house value based on 8 different features. The features used in this dataset are mostly float values with potentially low values found in features such as AveRooms or high values in the Population. This can have an effect on the comparison operations used in Homomorphic Encryption, which leads us to perform binning and normalization on these values to fit them in our dynamic range. Model training is done using standard techniques within `scikit-learn` and model outputs are normalized to the range $\{0.0, \ldots, 100.0\}$.

To perform output-hiding ensemble inference and compare against centralized learning, we use the train/test split from the centralized learning results and further split the training set according to the amount of model owners we wish to examine. The centralized learning model takes 90% of the available data from the California housing dataset and keeps 10% for testing purposes. Similarly we divide that same 90% of data over our chosen $\mathsf{N} = 4$ amount of models in an IID manner. The models are individually trained over their smaller portion of training data, and the resultant models are queried with the test data. These inference results are then summed and averaged and used to calculate the Mean Absolute Error (MAE) of the output-hiding ensemble inference of MIOPE. The goal of our evaluation is however not to create the best models for the task, but access the viability and accuracy performances of output-hiding ensemble inference when compared to the centralized learning. As such, our training process is not optimized for ideal accuracy.

For this evaluation, we consider Regressors in the form of a Decision Tree (DT) model, a GradientBooster model (GBR) and a eXtreme Gradient Boosting (XGBoost) model[8]. This

[7] Our code for the experimental set-up is available at: https://github.com/kyrianmaat/MIOPE-EPRINT

[8] We tested both AdaBoost and Random Forest models, but the achieved accuracy was considerably

Table 1: Parameters of the different tree-based ML models considered for the California Housing dataset. As the decision trees are a singular model, the depth here was varied while we kept the depth static for the gradient-boosting models that rely on building a forest from shallow decision trees. -S/M/L denote small, medium, large respectively. For the small and medium GBR we use learning rate of 0.2 and 0.1 for large.

Model	DT-S	DT-M	DT-L	XGB-S	XGB-M	XGB-L	GBR-S	GBR-M	GBR-L
Tree-depth	5	8	11	4	4	4	4	4	4
Estimator	N/A	N/A	N/A	25	100	400	25	100	400
DecisionNode	61	379	1281	775	3058	12288	765	3020	23050

provides us with a variety of different tree-based models for testing, where the high-depth singular tree of a Decision tree can be compared to the multiple tree evaluation of both gradient boosting models that train by building a forest of shallow trees. We performed a hyperparameter search for the models on tree depth and additionally amount of estimators for the gradient-boosting models. The combinations of parameters used for the small, medium and large versions of these models can be found in Fig. 1. The set-up described in Fig. 1 was used for training the 4 local models of the MIOPE framework, such that we have clusters of 4 similar tree-based models for each combination of parameters. Additionally, we also trained two mixed-model setups to simulate different models being trained in an ensemble, where we trained one mixture with 2 XGB, 1 GBR and 1 DT model and one mixture with 2 XGB and 2 GBR models. Lastly, we trained three XGBoost models using federated learning with 4 clients which mimics the amount of estimators for each of the models by running as many rounds as needed to create enough estimators. We chose to only compare XGB models as this was the most approachable via the Flower framework [BTM+20] but also represented the gradient-boosting model we wanted to compare on most.

To perform inference, the client and model owners use the `OpenFHE` library to setup cryptographic parameters and allow the client to encrypt the data. The data is sent over to the model owners, and the models perform encrypted inference using the GEMM algorithm as described in Section 4.1.1. The model summation for the XGB and GBR models is a simple internal addition of the ciphertext, and the final result is obtained by summation of all model outputs and dividing by the number of model owners. If a Regressor model has a `base_score` for prediction[9], then this has to be added on as an addition in the homomorphic circuit. The random masks can also added via homomorphic addition, which are provided beforehand to the model owners and applied to the encrypted output at inference time.

For the random masks we use the NIST public randomness beacon [Nat25] as our source of external randomness (see Section 2.4), where the time to be queried as input to the beacon is defined as the UTC time at which the client made the inference query request. Constructing the normalized random masks from this public random value, the client's identifier and model identifiers is implemented using either AES-CTR or HMAC as its PRF.

5.1 Evaluation

We compare the centralized learning models against the split models from MIOPE by using the Mean Absolute Error (MAE) as our main test statistic, comparing how it would perform better against a baseline Linear Regression model. A MAE under 0.5 is better

worse than the other models in our testing setup. Our encrypted inference pipeline still provides support for these model types.

[9]Specific for tree-based models, they can help steer a prediction to a certain base prediction

Table 2: Performance of models obtained via centralized learning and via summation over 4 split trained models, with lower MAE indicating better accuracy. Simple LR is a basic baseline model: a linear regressor that only takes into account the first feature. For the split learned models, the average decision nodes had at most a 1 to 2% difference in amount of decision nodes. HE results of large Decision Trees could not be measured due to memory requirements.

Model	Average DecisionNodes	MAE	R2	HE accuracy loss %
Simple LR	N/A	0.528	0.620	< 0.01
Small DT Centralized	61	0.511	0.609	< 0.01
Small DT 4-Split	61	0.479	0.658	< 0.01
Medium DT Centralized	379	0.428	0.701	0.05
Medium DT 4-Split	335	0.409	0.733	0.19
Large DT Centralized	1281	0.393	0.722	N/A
Large DT 4-Split	1117	0.373	0.756	N/A
Small XGB Centralized	775	0.446	0.732	< 0.01
Small XGB 4-Split	771	0.445	0.733	< 0.01
Small XGB FL	358	0.439	0.738	< 0.01
Medium XGB Centralized	3058	0.341	0.824	0.12
Medium XGB 4-Split	2910	0.349	0.817	0.4
Medium XGB FL	1383	0.363	0.805	0.12
Large XGB Centralized	12166	0.313	0.849	0.17
Large XGB 4-Split	11424	0.317	0.843	0.7
Large XGB FL	5500	0.369	0.786	0.15
Small GBR Centralized	765	0.378	0.791	< 0.01
Small GBR 4-Split	753	0.374	0.795	< 0.01
Medium GBR Centralized	3020	0.329	0.835	0.16
Medium GBR 4-Split	2954	0.333	0.831	0.42
Large GBR Centralized	12228	0.314	0.847	0.22
Large GBR 4-Split	11770	0.319	0.841	0.77
Small Mix (DT/GBR/XGB/XGB)	164	0.433	0.732	0.04
Small Mix (GBR/GBR/XGB/XGB)	762	0.399	0.766	0.05
Medium Mix (DT/GBR/XGB/XGB)	2277	0.358	0.805	0.35
Medium Mix (GBR/GBR/XGB/XGB)	2932	0.340	0.824	0.38
Large Mix (DT/GBR/XGB/XGB)	8934	0.331	0.825	N/A
Large Mix (GBR/GBR/XGB/XGB)	11597	0.315	0.844	0.81

than the baseline model, and the smaller the MAE the better. We also show the R2 score, where the closer to 1 the value is the better. We denote the amount of average number of decision nodes the model uses, the MAE associated with that model and the accuracy degradation during inference of using our HE scheme-switching implementation in `OpenFHE`.

As can be seen from Table. 2 , the larger models overall perform better than the starting linear regression algorithm. As the size goes up, the MAE also falls with the best performing models being the GradientBoosting and XGB large models, lowering the MAE to 0.313 from the simple linear regression model's 0.528.

When utilizing MIOPE and performing inference on the newly created split models, we see that the performance is close to the centralized datasets, with minor improvements on the smaller models or slightly lower accuracy. The amount of decision nodes on average for some of these models are higher than the centralized datasets, as the hyperparameter tuning on these individual lower data samples turned out to give different end results.

Comparing the mixed models shows us that they are comparable in accuracy if we omit the decision tree models, as they seem to drag down the average MAE. However, it also means that they can be utilized when lower-resource participants are part of the ensemble. The mixture of GBR and XGB models also seems to outperform an ensemble of only GBR or XGB models, but it is unlikely that this would be reliably reproducible in other experiments.

We ran the federated learning setup for 6, 25 and 100 rounds to obtain 24, 100 and 400 estimators respectively, but we observe that the performance of the FL set-up already peaks around 100 estimators, as the MAE actually drops from the medium to the large set-up. This can partially be attributed to less hyperparameter optimization in FL, but the resultant model also has significantly fewer decision nodes, which might imply that the local models stagnated at some point.

The overall accuracy loss that we observe when utilizing FHE in our approach never goes above 1% of the original accuracy, showcasing that the technique does not add on significant extra accuracy loss. We usually observed a small perturbation due to the nature of HE noise on the accuracy, and the usage of binning in our feature space means that the HE perturbations sometimes push our result in a slightly different bin, which explains the larger than 0.01% HE losses. When we use the MIOPE approach, we see that the error scales relatively linearly with the amount of models that are being used.

We chose to omit results on a split with 8 models trained in Table. 2 as they showcased worse MAE than the 4-split on all occasions except small and medium decision trees. Splitting more than 8 times on the data set would most likely lead to lower MAE scores and would require different model output summation techniques like weighted scoring, to obtain better results.

6 Related Literature

6.1 Encrypted Training of Machine Learning models

Encrypted training of machine learning is currently very infeasible in terms of computation and cost, but recent work has shown progress to try to bridge this gap for neural networks [NRPH19, LFFJ20, ZZS+24] and tree-based models [BSCG24]. Encrypted training gives strong advantages and can alleviate the problems of sharing data between model owners, but the training still need to be re-done if models need to be updated or added to the cluster. Our approach still benefits on model updateability at a data owner level, to avoid full re-training at every instance of new data or new model owners joining the setting.

6.2 Federated Learning and Secure Aggregation

There have been many advances in the Federated Learning domain creating a plethora of variations that each tackle different security, privacy and efficiency properties.

Asynchronous Federated Learning (AFL) [CNSR20, XQXG23] aims to reduce communication costs during FL: whenever updates are sent to the global model, aggregation will immediately happen on new local models. This means that the training time will be dependent on the weakest/slowest data owner. AFL also improves on the robustness of the system in terms of handling drop-outs, but overall still has a lengthier training time than MIOPE and relies on an aggregation server.

Decentralized Federated Learning (DFL) approaches remove the need for a central aggregation server during the computation [HBJ18, KMA+21, BPS+23], which means no singular entity will be able to observe all the model updates during the training time of the model and the singular entity holding the model at the end is also removed. Due to the nature of FL it still requires re-training for new data or data corruption and the overall training costs more time than MIOPE.

Utilising masks to safeguard updates in FL aggregation was explored in [BIK+17, BBG+20], which define how to do secure aggregation in a privacy-preserving manner in settings most applicable to Federated Learning. Their focus is primarily on keeping the

protocol robust under drop-outs, which is less significant in our setting where the participants are assumed to stay online, but the masking approach used in secure aggregation is very similar to our usage in MIOPE.

6.3 Related subworks to MIOPE

Training an ensemble of tree-based models has been assessed in the context of critical healthcare use cases such as heart disease prediction [SAA+25] and the conclusions can be directly applied to model training for MIOPE ensembles.

Giacomelli et al. [GJK+18] assume a central server that collects encrypted tree-based models (of a fixed size and structure) allowing this server to homomorphically evaluate the ensemble of models and provide a single output, therefore giving output privacy as MIOPE does. In comparison MIOPE aims to remove this singular node of failure and allow greater diversity in models/architectures.

Xu et al. [XZG24] consider federated learning with heterogeneous models instead of a single central model. They enable local models to train together via collective voting on an auxiliary dataset. Samples from this dataset are then labeled according to the votes and aggregated towards all the models if a consensus is reached. This work offers opportunities for the participating clients to maintain their own local models, however it still requires multiple communication rounds to aggregate votes via a central server. Using this approach for training could also be combined with the inference process of MIOPE for input and output privacy.

Another line of work has considered splitting inference query inputs to multiple entities that collaboratively perform inference for a single ML model, including the work of Ma et al. [MLL+19]. This approach follows prior work performing private face recognition using multi-party computation [EFG+09, SZ13], those prior works used dedicated and MPC-friendly classifiers rather than generic ML models.

6.4 Privacy-Preserving Inference on Machine-Learning models

PRICURE [JE21] also considers the setting of privacy-preserving collaborative inference among multiple model owners and a client, using secret-sharing with differential private aggregation to protect client input and neural-network model parameters against membership inference attacks. PRICURE allows for model owners to secret share their models with workers and relies on a trusted aggregator to aggregate the predictions and add differential noise, leading to a centralized inference model. MIOPE keeps the models local to the owners and use FHE and additive masking to allow for an ensemble inference without a trusted third-party aggregator.

7 Discussion and Future Work

Table 3: Comparison of MIOPE against other works. MIOPE specifically excels at allowing modular design and changes in the system, but has the most expensive average query time out of all available systems.

	This work	[GJK+18]	FL	Decentralized FL	FL + HE	[XZG24]
Adaptability to changes in used data	●	◐	◐	◐	◐	◐
Local training possible	●	●	○	◐	○	●
Functionality does not require third party	◐	○	○	●	○	○
No significant processing on third party	●	○	◐	●	◐	◐
Short time per query	○	◐	●	●	◐	●
Short time to first query	●	●	○	◐	○	◐
Model Agnosticity	●	●	○	○	○	●

We compare our framework with existing approaches from the literature in Table 3, showing the difference in the paradigms being used and the trade-offs the systems chose to use.

We want to stress that most of the performance impact of our approach is the utilization of Homomorphic Encryption for inference and thus input privacy. However, if a finalised model via Federated Learning is to be queried from a model owner perspective, they will also have to incur the HE penalty for input privacy. We can describe this in the following equations, albeit simplified:
$T_{MIOPE} = T_{HE} + T_{MASK} + T_{AGG} + T_{comm}$
$T_{FL+HE} = T_{HE} + T_{comm}$

Both of these are dominated by the execution time of Homomorphic Encryption which are 3 orders of magnitude higher (1000x) than plaintext processing [SCL+24]. MIOPE does run multiple instances of HE, but these are asynchronous and can be run in parallel among model owners, so the time taken is the slowest model owner in terms of compute, which makes the approach relatively comparable to the single HE inference for an FL setting. Note that input privacy inherently appears to require homomorphic encryption for processing of the ML model and therefore in the setting where multiple database owners train a single model with FL (or any other privacy-preserving training procedure), querying clients seeking input privacy will still have to pay the cost to execute inference using HE.

These trade-offs and advantages, coupled with the added constraint of settings where approaches such as federated learning are deemed too invasive on model sharing or where certain legal factors do not allow the usage of federated learning can make Output-Hiding Ensemble Inference a suitable solution, such as when entities from different countries have to work together. However if other techniques are able to be used in a specific scenario, the downsides of the average inference time might be too significant to offer MIOPE as a suitable alternative.

There are trade-offs made between MIOPE's output-hiding ensemble inference and federated learning in terms of the privacy of the models involved. In approaches using Federated Learning the model updates have to be guarded against malicious data owners, but the final model is less vulnerable to model inference attacks than MIOPE. Output-Hiding Ensemble Inference only has to protect the final model against malicious clients that could learn something from the summation, but the system does not have to safeguard local model updates.

We chose to perform our encrypted GEMM inference algorithm via scheme-switching as we had a mixture of linear and non-linear operations, but one could implement the entirety of the algorithm in an integer scheme such as CKKS, or a binary scheme such as FHEW or TFHE, instead. We chose not to explore this direction since it would require very careful noise management and to find parameters that retain acceptable model accuracy, but consider it future work that can be performed to test the accuracy and performance on these methods to compare to scheme switching.

In this work we have implemented our approach using the `OpenFHE` library over the `Concrete ML` library which is designed to support encrypted inference on machine learning models, mostly because PKE is not supported in `Concrete ML` and thus we cannot sum encrypted masks with the output of the models. If `Concrete ML` supported PKE outright, we would be able to train our models via the `Concrete ML` library or convert the trained models to run in `Concrete ML`. We let `Concrete ML` generate compiled models and cryptoparameters for the client/model pairings, and by using PKE we would add additional operations to add the masks on the output of the models. By adding support for summation on the client with the multiple models, we would be able to support the approach in `Concrete ML`. Support for PKE is currently only supported in the back-end of TFHE-rs, and not directly accessible by purely using the top-level `Concrete ML` library.

Homomorphic encryption invokes significant overheads in terms of inference processing

time, and also in terms of query bandwidth and key material. However, dedicated hardware solutions for encrypted processing are rapidly advancing [SFK+22, GBP+23]. Performance for tree-based encrypted inference is significantly affected by high-depth trees (affecting multiplicative depth of the function being evaluated homomorphically) and forest width (parallel computation is possible, as trees are essentially independent in the FHE sense). Performance overheads can be mitigated by prioritizing shallower and less dense forests. Early exit conditions or trimming the forests during training can help in lowering the computational requirements for FHE.

7.1 Stronger Adversaries and Collusion

In our instantiation of MIOPE we assume that all parties are honest-but-curious, which is a realistic assumption in settings where participants are entities with potential risks to their reputations and the system architecture already requires minimal trust assumptions between the model owners and querying clients. This means that we could tolerate model owners learning the masks of all other model owners (since HE still hides the inputs and outputs of the clients against honest-but-curious model owners). In practice this makes collusion between a client and any model owner a problem: for this reason we use a pair-wise masking technique to minimize the amount of data that is available to the model owners.

A malicious client can perform arbitrary attacks based on input-output pairs and deviate from the protocol to send out-of-range inputs. To mitigate this threat, MIOPE could be extended either by using a combination of homomorphic encryption scheme and model that ensures that the processing of these inputs always leads to an error output value, or by enforcing a check on the encrypted input, e.g. by asking the client to append a zero-knowledge proof that the input contains an allowed number of features, all within the correct ranges.

A malicious model owner can always withhold the response which is essentially a dropout for MIOPE, a client can count the responses to detect this trivially but not prevent it. Malicious models owners can also provide as masked model output an encryption of an invalid value, or apply an invalid mask to invalidate the aggregated inference process. In the mask generation protocol it could also misbehave, for example by trying to force another model owner to end up with a specific mask (such as 0 in the output space of the model). An adversary that controls multiple model owners can of course do these attacks and potentially have more chance of succeeding. For this reason a tailored approach is needed if usage of MIOPE requires resilience against malicious model owners[10].

In the event that collusion does happen, only combinations of client and model owner collusion would yield information, depending on the model output summation strategy used. If an adversary corrupts model owners, then only in a malicious setting can they influence the outcome of the protocol, as they cannot gain new information on the client due to homomorphic encryption. If an adversary does corrupt a client and model owner(s), then they can cancel out the masks on the outputs in the summation to infer information about the models of the other model owners whose masks cancel out. With a polynomial-time bounded machine, the steps through homomorphic encryption will still take time before a large amount of information of models can be leaked.

If an adversary can control or predict the outputs of the randomness source, then this could have an impact on the privacy leakage within our system. In a setting where the random masks are generated using a symmetric key that is not known to the querying client, then an adversary that controls the client and can predict beacon outputs does not gain any advantage in revealing masks (and therefore individual model outputs).

[10] It is also unclear how realistic this setting of a malicious model owner is, since a system with many clients could potentially disseminate information about the malicious behavior and remove that model owner from the ensemble.

7.2 Future Directions

One future avenue for improvement is to incorporate a mechanism for supporting weighted averages. This could for example allow a client to give a higher weight to models trained with more data points, similar to the concept of *contribution evaluation* in federated learning [XGiAP25]. This is not straightforward since the mask generation process needs to take these weights into account and the client's masked output aggregation process must ensure that the masks really do 'cancel out': the client cannot be the one to apply these weights since this is not possible without some leakage of individual model outputs. We consider it an open problem to efficiently incorporate weighted averages to the mask generation process.

The machine learning models, and particularly the tree-based models with gradient boosting, can support not only numerical or float values for features, but can also utilize categorical data or work on missing values in the features or dataset. However, categorical data and missing feature input values are currently not supported in the GEMM algorithmic approach from the `Hummingbird` library [NIS20, NSY+20], and our approach also does not support categorical or missing data. One way to add support for categorical data would be to change the model to change the singular categorical split on multiple category values to separate threshold splits in the tree, but this is non-trivial and adds significant overhead to every tree.

Supporting a wider array of machine learning model types is also a challenge. Regressors are particularly well suited, while for classifiers it is necessary for the clients in the system to judge how to treat the outputs (masking is straightforward, since classes can be encoded as integers), for example how to proceed if two classes are the result of an equal number of models in the ensemble. Binary classifiers can focus on returning one of the class probabilities and averaging over that, or purely summing up all the predictions and using this as a majority voting, which servers could do with similar HE settings [CJO23]. Multi-class classifiers can use a summation function that takes into account the prediction scores for the different labels to sum over. For other models such as ranking or clustering models it might be more challenging to define an appropriate function to interpret and sum the model outputs. Generative models can be executed in homomorphic encryption to give input privacy [ZWS+26, BAPV26] however they are not well suited to direct usage in MIOPE since their outputs cannot be aggregated in a straightforward manner. An option is to therefore use generative models foregoing full output privacy, where the client receives the outputs of each model: the output aggregation step could be replaced with a model owner anonymization step where a mixnet [Cha81] or similar is executed between model owners to hide which output came from which model owner. A full analysis of such an approach is beyond the scope of this work: querying multiple generative models under FHE for a single input is likely computationally/financially infeasible for the majority of practical use cases.

7.3 Concluding

This paper presented MIOPE, a modular framework that uses output-hiding encrypted ensemble inference to provide input privacy for clients and output privacy for individual model owners without relying on a trusted third-party aggregator. By utilizing a combination of FHE and MPC-style output masking MIOPE allows pre-trained local models from model owners to participate in collaborative and heterogeneous inference without giving up any control of their models. The modular design of MIOPE allows any components, such as encryption scheme or aggregation strategy, to be substituted independently between client and model owner pair, giving MIOPE an adaptable and privacy-sensitive deployment setting and a practical alternative to scenarios where federated learning might not be viable.

References

[AAR+19] Roohallah Alizadehsani, Moloud Abdar, Mohamad Roshanzamir, et al. Machine learning-based coronary artery disease diagnosis: A comprehensive review. *Comput. Biol. Medicine*, 111, 2019.

[ALR+22] Adi Akavia, Max Leibovich, Yehezkel S. Resheff, Roey Ron, Moni Shahar, and Margarita Vald. Privacy-preserving decision trees training and prediction. *ACM Trans. Priv. Secur.*, 25(3):24:1–24:30, 2022.

[BAB+22] Ahmad Al Badawi, Andreea Alexandru, Jack Bates, et al. OpenFHE: Open-source fully homomorphic encryption library. GitHub Repository, 2022. URL: https://github.com/openfheorg/openfhe-development.

[BAPV26] Ahmad Al Badawi, Andreea Alexandru, Yuriy Polyakov, and Vinod Vaikuntanathan. SoK: Private LLM inference using approximate homomorphic encryption. Cryptology ePrint Archive, Paper 2026/935, 2026. URL: https://eprint.iacr.org/2026/935.

[BBB+22] Ahmad Al Badawi, Jack Bates, Flávio Bergamaschi, et al. OpenFHE: Open-source fully homomorphic encryption library. In *Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, pages 53–63. ACM, 2022.

[BBG+20] James Henry Bell, Kallista A. Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. Secure single-server aggregation with (poly)logarithmic overhead. In *ACM SIGSAC CCS*, pages 1253–1269, 2020.

[BCG15] Joseph Bonneau, Jeremy Clark, and Steven Goldfeder. On bitcoin as a public randomness source. *IACR Cryptol. ePrint Arch.*, page 1015, 2015. URL: http://eprint.iacr.org/2015/1015.

[BGGJ20] Christina Boura, Nicolas Gama, Mariya Georgieva, and Dimitar Jetchev. CHIMERA: combining ring-LWE-based fully homomorphic encryption schemes. *J. Math. Cryptol.*, 14(1):316–338, 2020.

[BGV11] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. Fully homomorphic encryption without bootstrapping. *Electron. Colloquium Comput. Complex.*, TR11-111, 2011. URL: https://eccc.weizmann.ac.il/report/2011/111, arXiv:TR11-111.

[BIK+17] Kallista A. Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *ACM SIGSAC CCS*, pages 1175–1191, 2017.

[BPS+23] Enrique Tomás Martínez Beltrán, Mario Quiles Pérez, Pedro Miguel Sánchez Sánchez, Sergio López Bernal, Gérôme Bovet, Manuel Gil Pérez, Gregorio Martínez Pérez, and Alberto Huertas Celdrán. Decentralized federated learning: Fundamentals, state of the art, frameworks, trends, and challenges. *IEEE Commun. Surv. Tutorials*, 25(4):2983–3013, 2023.

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In *CRYPTO 2012*, pages 868–886, 2012.

[BSCG24] Divyanshu Bhardwaj, Sandhya Saravanan, Nishanth Chandran, and Divya Gupta. Securely training decision trees efficiently. In *ACM SIGSAC CCS*, pages 4673–4687, 2024.

[BTM+20] Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Hei Li Kwing, Titouan Parcollet, Pedro PB de Gusmão, and Nicholas D Lane. Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390*, 2020.

[CDPP22] Kelong Cong, Debajyoti Das, Jeongeun Park, and Hilder V. L. Pereira. SortingHat: Efficient private decision tree evaluation via homomorphic encryption and transciphering. In *ACM SIGSAC CCS*, pages 563–577, 2022.

[CGGI16] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In *ASIACRYPT 2016*, pages 3–33, 2016.

[CGGI20] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. TFHE: fast fully homomorphic encryption over the torus. *J. Cryptol.*, 33(1):34–91, 2020.

[CH10] Jeremy Clark and Urs Hengartner. On the use of financial data as a random beacon. In *EVT/WOTE*, 2010.

[Cha81] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–88, 1981. doi:10.1145/358549.358563.

[CJO23] Harsh Chaudhari, Matthew Jagielski, and Alina Oprea. Safenet: The unreasonable effectiveness of ensembles in private collaborative learning. In *2023 IEEE Conference on Secure and Trustworthy Machine Learning, SaTML 2023, Raleigh, NC, USA, February 8-10, 2023*, pages 176–196. IEEE, 2023. URL: https://doi.org/10.1109/SaTML54575.2023.00021, doi:10.1109/SATML54575.2023.00021.

[CKKS17] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yong Soo Song. Homomorphic encryption for arithmetic of approximate numbers. In *ASIACRYPT 2017*, pages 409–437, 2017.

[CNSR20] Yujing Chen, Yue Ning, Martin Slawski, and Huzefa Rangwala. Asynchronous online federated learning for edge devices with non-iid data. In *IEEE International Conference on Big Data (IEEE BigData 2020),*, pages 15–24. IEEE, 2020.

[DDS+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition (CVPR 2009)*, pages 248–255. IEEE Computer Society, 2009.

[DM15] Léo Ducas and Daniele Micciancio. FHEW: bootstrapping homomorphic encryption in less than a second. In *EUROCRYPT 2015*, pages 617–640, 2015.

[Dwo06] Cynthia Dwork. Differential privacy. In *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006*, pages 1–12, 2006.

[EFG+09] Zekeriya Erkin, Martin Franz, Jorge Guajardo, Stefan Katzenbeisser, Inald Lagendijk, and Tomas Toft. Privacy-preserving face recognition. In *PETS*, pages 235–253, 2009.

[Efr92] Bradley Efron. Bootstrap methods: another look at the jackknife. In *Breakthroughs in statistics: Methodology and distribution*, pages 569–593. Springer, 1992.

[FSB+24] Jordan Frery, Andrei Stoian, Roman Bredehoft, Luis Montero, Celia Kherfallah, Benoit Chevallier-Mames, and Arthur Meyre. Privacy-preserving tree-based inference with TFHE. In *Mobile, Secure, and Programmable Networking*, pages 139–156, 2024.

[FV12] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptol. ePrint Arch.*, page 144, 2012. URL: http://eprint.iacr.org/2012/144.

[GBDM20] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients - how easy is it to break privacy in federated learning? In *NeurIPS*, 2020.

[GBP+23] Robin Geelen, Michiel Van Beirendonck, Hilder V. L. Pereira, Brian Huffman, Tynan McAuley, Ben Selfridge, Daniel Wagner, Georgios D. Dimou, Ingrid Verbauwhede, Frederik Vercauteren, and David W. Archer. BASALISC: programmable hardware accelerator for BGV fully homomorphic encryption. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(4):32–57, 2023. URL: https://doi.org/10.46586/tches.v2023.i4.32-57, doi:10.46586/TCHES.V2023.I4.32-57.

[Gen09] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, USA, 2009. URL: https://searchworks.stanford.edu/view/8493082.

[GJK+18] Irene Giacomelli, Somesh Jha, Ross Kleiman, David Page, and Kyonghwan Yoon. Privacy-preserving collaborative prediction using random forests, 2018. URL: http://arxiv.org/abs/1811.08695, arXiv:1811.08695.

[GLMS23] Alejandro Guerra-Manzanares, L. Julian Lechuga Lopez, Michail Maniatakos, and Farah E. Shamout. Privacy-preserving machine learning for healthcare: Open challenges and future perspectives. In *Trustworthy Machine Learning for Healthcare (TML4H)*, pages 25–40, 2023.

[GST+99] Todd R Golub, Donna K Slonim, Pablo Tamayo, Christine Huard, Michelle Gaasenbeek, Jill P Mesirov, Hilary Coller, Mignon L Loh, James R Downing, Mark A Caligiuri, et al. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):531–537, 1999.

[HBJ18] Lie He, An Bian, and Martin Jaggi. COLA: decentralized linear learning. In *NeurIPS*, pages 4541–4551, 2018.

[JE21] Ismat Jarin and Birhanu Eshete. PRICURE: privacy-preserving collaborative inference in a multi-party setting. In *ACM Workshop on Security and Privacy Analytics*, pages 25–35, 2021.

[KMA+21] Peter Kairouz, H. Brendan McMahan, Brendan Avent, et al. Advances and open problems in federated learning. *Found. Trends Mach. Learn.*, 14(1-2):1–210, 2021.

[KMR15] Jakub Konečný, Brendan McMahan, and Daniel Ramage. Federated optimization: Distributed optimization beyond the datacenter, 2015. URL: http://arxiv.org/abs/1511.03575, arXiv:1511.03575.

[KMY+16] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency, 2016. URL: http://arxiv.org/abs/1610.05492, arXiv:1610.05492.

[KNL+19] Ágnes Kiss, Masoud Naderpour, Jian Liu, N. Asokan, and Thomas Schneider. SoK: Modular and efficient private decision tree evaluation. *Proc. Priv. Enhancing Technol.*, 2019(2):187–208, 2019.

[KTBG20] Dan Kondratyuk, Mingxing Tan, Matthew Brown, and Boqing Gong. When ensembling smaller models is more efficient than single large models, 2020. URL: https://arxiv.org/abs/2005.00570, arXiv:2005.00570.

[LFFJ20] Qian Lou, Bo Feng, Geoffrey Charles Fox, and Lei Jiang. Glyph: Fast and accurately training deep neural networks on encrypted data. In *NeurIPS*, 2020.

[LHH+21] Wen-jie Lu, Zhicong Huang, Cheng Hong, Yiping Ma, and Hunter Qu. PEGASUS: bridging polynomial and non-polynomial evaluations in homomorphic encryption. In *42nd IEEE Symposium on Security and Privacy, SP 2021*, pages 1057–1073. IEEE, 2021.

[MKV+23] Duncan McElfresh, Sujay Khandagale, Jonathan Valverde, Vishak Prasad C, Ganesh Ramakrishnan, Micah Goldblum, and Colin White. When do neural nets outperform boosted trees on tabular data? *Advances in Neural Information Processing Systems*, 36:76336–76369, 2023.

[MLL+19] Zhuo Ma, Yang Liu, Ximeng Liu, Jianfeng Ma, and Kui Ren. Lightweight privacy-preserving ensemble classification for face recognition. *IEEE Internet Things J.*, 6(3):5778–5790, 2019.

[MNLK23] Rasoul Akhavan Mahdavi, Haoyan Ni, Dimitry Linkov, and Florian Kerschbaum. Level up: Private non-interactive decision tree evaluation using levelled homomorphic encryption. In *ACM SIGSAC CCS*, pages 2945–2958, 2023.

[MWCB24] Zoltán Ádám Mann, Christian Weinert, Daphnee Chabal, and Joppe W Bos. Towards practical secure neural network inference: the journey so far and the road ahead. *ACM Computing Surveys*, 56(5), 2024.

[Nat25] National Institute of Standards and Technology. Interoperable randomness beacons: Beacon 2.0. https://csrc.nist.gov/projects/interoperable-randomness-beacons/beacon-20, 2025.

[NIS20] Supun Nakandala, Matteo Interlandi, and Karla Saur. Hummingbird. GitHub Repository, 2020. URL: https://github.com/microsoft/hummingbird.

[NRPH19] Karthik Nandakumar, Nalini K. Ratha, Sharath Pankanti, and Shai Halevi. Towards deep neural network training on encrypted data. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 40–48, 2019.

[NSY+20] Supun Nakandala, Karla Saur, Gyeong-In Yu, Konstantinos Karanasos, Carlo Curino, Markus Weimer, and Matteo Interlandi. A tensor compiler for unified machine learning prediction serving. In *14th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2020*, pages 899–917, 2020.

[PB97] R. Kelley Pace and Ronald Barry. Sparse spatial autoregressions. *Statistics and Probability Letters*, 33:291–297, 1997.

[Rab83] Michael O. Rabin. Transaction protection by beacons. *J. Comput. Syst. Sci.*, 27(2):256–267, 1983.

[SAA+25] Enoch Sakyi-Yeboah, Edmund Fosu Agyemang, Vincent Agbenyeavu, Akua Osei-Nkwantabisa, Priscilla Kissi-Appiah, Lateef Moshood, Lawrence Agbota, and Ezekiel N. N. Nortey. Heart disease prediction using ensemble tree algorithms: A supervised learning perspective. *Appl. Comput. Intell. Soft Comput.*, 2025(1), 2025.

[SCL+24] Hojune Shin, Jina Choi, Dain Lee, Kyoungok Kim, and Younho Lee. Fully homomorphic training and inference on binary decision tree and random forest. In Joaquín García-Alfaro, Rafal Kozik, Michal Choras, and Sokratis K. Katsikas, editors, *Computer Security - ESORICS 2024 - 29th European Symposium on Research in Computer Security, Bydgoszcz, Poland, September 16-20, 2024, Proceedings, Part III*, volume 14984 of *Lecture Notes in Computer Science*, pages 217–237. Springer, 2024. doi:10.1007/978-3-031-70896-1_11.

[SFK+22] Nikola Samardzic, Axel Feldmann, Aleksandar Krastev, Nathan Manohar, Nicholas Genise, Srinivas Devadas, Karim Eldefrawy, Chris Peikert, and Daniel Sánchez. Craterlake: a hardware accelerator for efficient unbounded computation on encrypted data. In Valentina Salapura, Mohamed Zahran, Fred Chong, and Lingjia Tang, editors, *ISCA '22, New York, New York, USA, June 18 - 22, 2022*, pages 173–187. ACM, 2022. doi:10.1145/3470496.3527393.

[SGU25] Aparna Saju, Shivaprasad Gundibail, and Ujjwal. Cloud-enabled predictive modeling of mental health using ensemble machine learning models and AES-256 security. *IEEE Access*, 13:152948–152961, 2025.

[SSSS17] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy, SP 2017*, pages 3–18. IEEE Computer Society, 2017.

[SZ13] Thomas Schneider and Michael Zohner. GMW vs. Yao? efficient secure two-party computation with low depth circuits. In *Financial Cryptography and Data Security (FC)*, pages 275–292, 2013.

[TBK20] Anselme Tueno, Yordan Boev, and Florian Kerschbaum. Non-interactive private decision tree evaluation. In *DBSec*, pages 174–194, 2020.

[TMZC17] Raymond K. H. Tai, Jack P. K. Ma, Yongjun Zhao, and Sherman S. M. Chow. Privacy-preserving decision trees evaluation via linear functions. In *22nd European Symposium on Research in Computer Security (ESORICS)*, pages 494–512, 2017.

[WFNL16] David J. Wu, Tony Feng, Michael Naehrig, and Kristin E. Lauter. Privately evaluating decision trees and random forests. *Proc. Priv. Enhancing Technol.*, 2016(4):335–355, 2016.

[XGiAP25] Marvin Xhemrishi, Alexandre Graell i Amat, and Balazs Pejo. Detect & score: Privacy-preserving misbehavior detection and contribution evaluation in federated learning. In *Intl. Workshop on Secure and Efficient Federated Learning*, FL-AsiaCCS '25, 2025.

[XQXG23] Chenhao Xu, Youyang Qu, Yong Xiang, and Longxiang Gao. Asynchronous federated learning on heterogeneous devices: A survey. *Comput. Sci. Rev.*, 50:100595, 2023.

[XZG24] Yukai Xu, Jingfeng Zhang, and Yujie Gu. Privacy-preserving heterogeneous federated learning for sensitive healthcare data, 2024. URL: https://doi.org/10.48550/arXiv.2406.10563, arXiv:2406.10563.

[YZH22] Xuefei Yin, Yanming Zhu, and Jiankun Hu. A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions. *ACM Comput. Surv.*, 54(6):131:1–131:36, 2022.

[Zam22a] Zama. Concrete ML, 2022. https://github.com/zama-ai/concrete-ml.

[Zam22b] Zama. Concrete: TFHE Compiler that converts Python programs into FHE equivalent, 2022. https://github.com/zama-ai/concrete.

[ZDW13] Yuchen Zhang, John C. Duchi, and Martin J. Wainwright. Communication-efficient algorithms for statistical optimization. *J. Mach. Learn. Res.*, 14(1):3321–3363, 2013.

[ZLL+18] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-IID data, 2018. URL: http://arxiv.org/abs/1806.00582, arXiv:1806.00582.

[ZLZ23] Haili Zhang, Zhaobo Liu, and Guohua Zou. Least squares model averaging for distributed data. *J. Mach. Learn. Res.*, 24:215:1–215:59, 2023.

[ZWS+26] Linru Zhang, Xiangning Wang, Jun Jie Sim, Zhicong Huang, Jiahao Zhong, HUAXIONG WANG, Pu Duan, and Kwok-Yan Lam. MOAI: Module-optimizing architecture for non-interactive secure transformer inference. In *The Fourteenth International Conference on Learning Representations*, 2026. URL: https://openreview.net/forum?id=qJn4HtTzhH.

[ZYWL25] Zhiyuan Zhai, Xiaojun Yuan, Xin Wang, and Geoffrey Ye Li. Decentralized federated learning with distributed aggregation weight optimization. *arXiv preprint arXiv:2511.03284*, 2025.

[ZZC+19] Zhongheng Zhang, Yiming Zhao, Aran Canes, Dan Steinberg, Olga Lyashevska, et al. Predictive analytics with gradient boosting in clinical medicine. *Annals of Translational Medicine*, 7(7):152, 2019.

[ZZS+24] Yancheng Zhang, Mengxin Zheng, Yuzhang Shang, Xun Chen, and Qian Lou. HEPrune: Fast private training of deep neural networks with encrypted data pruning. In *NeurIPS*, 2024.