GenoPPML – a framework for genomic privacy-preserving machine learning

Sergiu Carpov, Nicolas Gama, Mariya Georgieva, and Dimitar Jetchev

Inpher, Switzerland

Abstract. We present a framework GenoPPML for privacy-preserving machine learning in the context of sensitive genomic data processing. The technology combines secure multiparty computation techniques based on the recently proposed Manticore secure multiparty computation framework for model training and fully homomorphic encryption based on TFHE for model inference. The framework was successfully used to solve breast cancer prediction problems on gene expression datasets coming from distinct private sources while preserving their privacy - the solution winning 1st place for both Tracks I and III of the genomic privacy competition iDASH'2020.

Keywords: privacy-preserving machine learning · multi-party computation · homomorphic encryption · genomic privacy · differential privacy

1 Introduction

Advanced machine learning modeling techniques are used extensively on genomic, epigenomic and proteomic data to address a large class of problems in the bioinformatics and biomedical fields ranging from developing robust digital diagnostic tools, better understanding of transcription factor binding and gene regulation [3], genomic wide association studies (GWAS), mRNA translation, protein folding [50], protein functions and interactions [31] as well as drug discovery and molecular design [55].

A large amount of the biomedical data processed by these techniques is publicly available (e.g., ENCODE[1], PDB[2], TCGA[3], etc.). Yet, the massive cost reduction of human genome sequencing via next generation sequencing technologies and tools largely increase the available individuals' genomic data. The latter helps build better models for the causal relations between predisposition to diseases, response to treatments, effect of drugs, and more generally, it enables personal and precision medicine. It also represents a natural candidate for outsourced machine learning that can benefit from various cloud services and resources.

[1] The Encyclopedia of DNA Elements (ENCODE) – `https://www.encodeproject.org`

[2] The Protein Data Bank (PDB) – `http://www.wwpdb.org`

[3] The Cancer Genome Atlas (TCGA) – official website `http://cancergenome.nih.gov`

A major challenge in the use of individuals' genomic data for biomedical research and diagnostic tools is the high sensitivity as well as the legal regulations. For instance, the *Genetic Information Nondiscrimination Act* of 2008 [4] prohibits genetic discrimination, thus, strictly limiting the disclosure and leakage of data to third parties (e.g., cloud service providers). To address this challenge, one needs effective data protection methods that enable accurate and efficient computation without leaking information about the individual genome sequences. The trade-off between the computational efficiency, the security level and the accuracy of the result is essential for the practical application of these methods.

The Annual iDASH Privacy and Security Workshop[5] brings together experts on privacy-enhancing technologies (PETs) and secure cryptographic computing to propose and implement privacy-preserving computation methods applied to specific problems from the genomic and the bioinformatics fields. The worldwide iDASH competition evaluates state-of-the-art cryptographic technologies for practical secure computation such as homomorphic encryption (HE), federated learning (FL), differential privacy (DP) among others.

Track III from the 2020 edition featured a federated learning model building for cancer prediction using gene expression datasets coming from different parties without exposing any of the plaintext genomic data. More specifically, numerical gene expression data for a large number of genes is collected for samples from both normal and cancerous tissues. The goal is to predict distant metastasis of breast cancer (see [58] for details). In a second track, a cancer prediction model available in plaintext is to be used to provide prediction services on encrypted genomic data.

In this work, we introduce a unified framework GenoPPML that accomplishes both tasks. A logistic regression model is trained on a joint genomic dataset composed of data coming from several private data sources. The model is subsequently used to perform predictions over individuals' encrypted genomic data samples. The privacy of the input dataset, model and the genomic data samples is guaranteed via a combination of secret sharing, differential privacy and homomorphic encryption techniques. Note that the methods and the techniques used in GenoPPML are the award-winning solutions submitted by Inpher's team for two of the three tracks of the iDASH'2020 competition, the only minor difference being that the prediction is adapted to better support individual genomic data samples (as opposed to prediction on batched samples during the contest). We tested our framework using breast cancer gene expression data from TCGA (BC-TCGA) for privacy-preserving training and predictions of a logistic regression model.

[4] `https://www.govinfo.gov/content/pkg/PLAW-110publ233/pdf/PLAW-110publ233.pdf`

[5] `http://www.humangenomeprivacy.org`

2 Background and related works

In this section, we introduce the privacy-enhancing technologies (PETs) and tools needed for GenoPPML. We then review related literature on privacy-preserving machine learning (PPML) with emphasis to the genomic field.

2.1 Multi-party computation

Multiparty computation (MPC) is a method for cryptographic computing allowing several parties holding private data to evaluate a public function on their aggregate data without revealing anything except what is logically implied by the output of the function. Recent works make these protocols practical [10,37,13,47,36]. MPC computations have been used in different domains with sensitive data such as genomic studies [25] and other industry verticals in a variety of use cases including machine learning applications such as linear regressions and logistic regressions [44,13], decision trees [30,26], neural networks [57], etc.

Several MPC protocols and libraries have been proposed in the literature and implemented. `SecureML` [44] provides a practical method for privacy-preserving machine learning in the two-party setting using a combination of arithmetic, Boolean and Yao shares. `Sharemind` [10] together with subsequent extensions such as [48] for both arithmetic, Boolean secret shares as well as garbled circuits is a framework providing 1-out-of-3 and full threshold security model.

Manticore is a highly efficient MPC framework with full threshold and semi-honest security model. The protocol and implementation details are described in [16]. It supports arithmetic with both real numbers and Booleans (via arithmetic and boolean secret sharing, as well as garbled circuits). Machine learning algorithms such as linear and logistic regression [16], as well XGBoost [26] amongst others, are implemented in Manticore at scale.

The main MPC functionalities of the Manticore framework used in this work are matrix addition/multiplication and sigmoid evaluation via Fourier approximation:

- classical approaches based on Beaver triples [7] are used for the evaluation of linear combinations and multiplications,
- the Fourier approximation method proposed in [13,16] is used for sigmoid function evaluation. Here, the sigmoid function is approximated uniformly on a given interval with Fourier series whose coefficients decay exponentially fast.

2.2 Homomorphic encryption

Homomorphic encryption (HE) is a PET that allows to perform computations directly over encrypted data without revealing inputs, outputs or any intermediary data. A major difference from MPC is that HE does not need data owner interaction during the computation phase. Additive/multiplicative HE schemes are known since the late 1970's. Yet, the first scheme supporting simultaneously

both addition and multiplication without restriction on the number of operations was proposed in 2009 by Gentry [34]. The study of HE schemes is an active area of research and recent advances bring technologies closer to being efficient in practice.

For security reasons all ciphertexts for HE schemes supporting simultaneously addition and multiplication, for any prescribed multiplicative depth, include a noise component. After each homomorphic operation, the noise increases and once it exceeds a certain threshold, the ciphertext can no longer be correctly decrypted. HE schemes supporting addition and multiplication for an arbitrary prescribed multiplicative depth of operations are called leveled homomorphic encryption (LHE), whereas schemes not limited by the complexity and depth of the operations are called fully homomorphic encryption (FHE) schemes.

Bootstrapping techniques are used to reduce the noise components in ciphertexts to predefined values and thus, convert LHE schemes to FHE schemes. Several HE schemes have been proposed in the literature: BFV [29], BGV [15], CKKS [21] and TFHE [22,24]. The first three schemes are suitable for ciphertexts encrypting multiple plaintext values (SIMD) in LHE mode whereas the TFHE scheme is well-adapted for ciphertexts encrypting single plaintext value in FHE mode.

TFHE (Torus Fully Homomorphic Encryption) [22] defines messages and ciphertexts over the torus $\mathbf{R}/\mathbf{Z}$ and keeps track of the noise standard deviation $\alpha \ll 1$, a dynamic parameter that changes after each operation. Therefore, plaintexts have $\ell = -\log_2(\alpha)$ fractional bits of precision. TFHE can supports different plaintext space representations. The ones used in GenoPPML are:

- TLWE encrypts individual elements of $\mathbf{R}/\mathbf{Z}$,
- TRLWE encrypts packings of N individual torus elements (or vectors in $(\mathbf{R}/\mathbf{Z})^N$) associated to the coefficients of polynomials modulo $Z^N + 1$.

Various homomorphisms and algebraic structures allow to switch between these two representations. For each of these plaintext representations, specific encryption and decryption functions are defined. Below, we describe the functionalities of TFHE scheme that are used in this work.

- Arithmetic function Add takes as input two TRLWE ciphertexts and adds the corresponding encrypted messages together in a new TRLWE ciphertext.
- The plaintext-ciphertext multiplication function Mult takes as inputs a public polynomial with integer coefficients and a TRLWE ciphertext and returns a new TRLWE ciphertext encoding their product.
- The coefficient extract operation $\mathsf{CoefExtr}_i$ takes as input a TRLWE ciphertext and outputs a TLWE ciphertext which encodes the i-th polynomial coefficient of the input ciphertext with at most the same noise variance or amplitude.
- The functional bootstrapping procedure $\mathsf{FuncBoot}_f$ [14] allows to evaluate arbitrary point-wise defined function f over an TLWE ciphertext message (in addition to noise reduction). Simple rounding functionality is used in [27,22] for *gate bootstrapping* and it was further generalized to arbitrary functions in [9,23,19].

2.3 Federated learning

Federated learning (FL) [41,43] is a distributed machine learning approach where training data is decentralized and typically comes from a large collection of client devices (e.g., mobile phones, sensors, etc.). Each client trains a local model that then gets aggregated by a central server.

Usually, the aggregating server has access to all the model updates of the clients. Knowing these model updates enables the server to often reconstruct the original clients' data via the so-called model inversion attacks [61,33]. To prevent such attacks, one requires the server aggregation to be performed in privacy-preserving manner. Various methods for secure aggregation based on differential privacy [1,42], pairwise additive masking [11,8], homomorphic encryption, additive secret sharing [54], have been proposed in the literature.

In this work, we propose a hybridization between a pure multiparty computation and a federated learning with secure aggregation based on additive secret sharing and differential privacy.

2.4 Privacy-preserving machine learning

Linear and logistic regressions are one of the most basic machine learning tools used in genomic studies, although most recently, state-of-the-art models in deep neural networks are successfully applied to prediction problems (see the introduction). During the last years, there have been numerous approaches to implement computations securely on medical data; these involve, on the one hand, (a) distributed settings where two or more parties collectively compute a function such as a linear or logistic regression, by applying technologies such as garbled circuits [35], in settings limited to two parties, or secret sharing and multiparty computation [16], through interactive protocols; these solutions require non-colluding computing parties and heavily rely on communication between them; on the other hand, (b) outsourced scenarios move the bulk of the computation to an untrusted third party in a non-interactive setting, either relying on trusted hardware such as Intel SGX [46,49,20], therefore requiring some degree of trust on the hardware manufacturer, or homomorphic encryption [4,39,52,53,18,40]. The latter does not need any assumption on the hardware, as it is solely based on the cryptographic guarantees of the used cryptosystems, so it can be seen as the most promising approach for outsourcing medical computations.

The authors of [4] proposed a method for training a logistic regression based only on additive homomorphic encryption, which requires the client to precompute some intermediate values in order to account for the limited range of operations (additions). Several works use levelled homomorphic cryptosystems, enabling both encrypted additions and a limited number of encrypted products, to implement the basic logistic regression block, i.e. a Gradient descent algorithm with an approximated Sigmoid function on an encrypted matrix of input data. Amongst them, Kim et al [39] employed a Nesterov's accelerated Gradient descent algorithm with the CKKS cryptosystem, which supports homomorphic

rescaling and approximate arithmetic; Bonte et al [12] implemented one iteration of a simplified fixed Hessian method with the BFV cryptosystem. Carpov et al [18] implemented a semi-parallel logistic regression training algorithm using the Chimera framework [14] employing a combination of TFHE and CKKS HE schemes. The easier task of logistic regression prediction using HE systems is treated in work [40]. The authors propose several methods, which differ by the employed HE scheme, with different computational performance vs accuracy trade-offs.

3 GenoPPML framework

The GenoPPML framework trains a prediction model (logistic regression) on private genomic datasets and provides private genomic data prediction service to users. The privacy of the input genomic datasets, the prediction model as well as the users' prediction data is guaranteed via a combination of FL, MPC, FHE and DP described in Section 2.

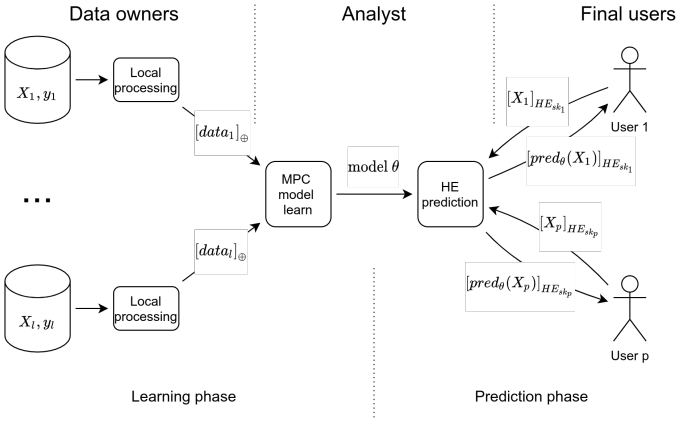

Fig. 1. GenoPPML architecture. Private datasets are processed locally for dimensionality reduction and outputs are secret shared for model training in MPC. The computed model is then used for prediction on users' data encrypted via an FHE scheme. Users receive and decrypt the computed encrypted predictions.

Figure 1 describes GenoPPML. Three types of actors are involved in the data processing: (i) genomic dataset owners, (ii) data analyst and (iii) final users. In a first phase (learning phase), the database owners and the analyst train a joing logistic regression model via MPC. A differential privacy mechanism is then used in order to ensure the privacy of the genomic database owners against model inversion attacks [32], [60]. Once the analyst has the model, it provides a private prediction service to users in a second phase (prediction phase). Homomorphic encryption is used to protect the privacy of user genomic data during this phase.

In our threat model, we assume that all communication channels between all database owners and analyst are private and authenticated (encrypted). Moreover, we assume that the database owners, the analyst and the final users provide the correct input and follow the protocol. Note that in the case where the final user does not follow the protocol and perform a large number of requests to the analyst, the only information that can learn is the model that is already protected by differential privacy.

In what follows each phase of the GenoPPML is described in more details.

3.1 Model learning phase

The multi-party logistic regression learning is performed in two steps. In the first step the genomic database owners perform a local preprocessing and afterwards a MPC protocol is used to find the final model. The objective of the local preprocessing is to reduce the dimensionality of genomic databases by applying a feature selection procedure. Genomic databases have the same set of features.

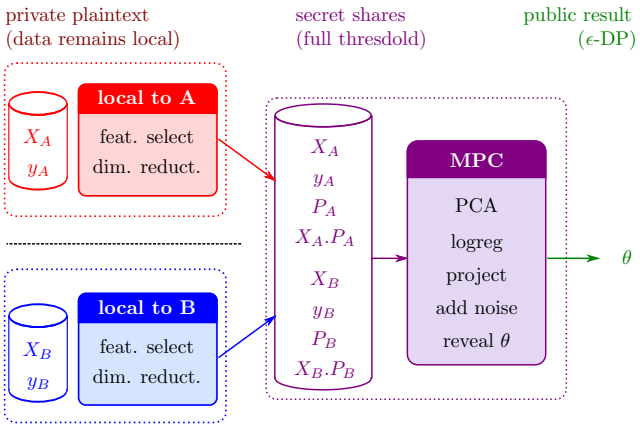

Fig. 2. Example of a 2-party MPC training data-flow chart.

As an example, and also for conciseness, we describe the learning phase restricted to 2 database owners/players. This procedure is illustrated in figure 2. Here, X_A, y_A are n_A samples known by database owner A (in red in figure 2), and X_B, y_B are n_B samples known by database owner B (in blue in figure 2). X_A, X_B are feature matrices, of dimension respectively $n_A \times k$ and $n_B \times k$, with the same same number of features k, and the class vectors y_A, y_B are binary vectors of length respectively n_A and n_B.

Each data owner first executes a local computation on their data: here, a feature selection/projection and secret-share the outcome (in a secret-shared distributed database in purple on figure 2). The result of feature selection step are projection matrices P_A, P_B which allow to project input datasets from k to

h features, where $h \ll k$. Then, both players jointly finish with a secret-shared multiparty computation using the secret-shared database. At all times, Player A and Player B keep one share of each input and intermediate variable: a share has marginal uniform distribution and does not reveal any bit of information about the variable. It is only by combining of two shares of A and B that a plaintext value can be reconstructed: such online operation requires the consent of both players, and is only applied to the final model. By standard security of MPC protocols, every exchange between player A and B during the whole training process are uniformly masked and do not carry any bit of information, except the final model once a suitable level of differential privacy (DP) noise has been added.

In addition to the dimensions n_A, n_B, k, and projection size h, training uses differential privacy parameters ϵ or δ: when $\delta = 0$ ϵ-differential privacy is obtained, whereas when $\delta > 0$ (ϵ, δ)-differential privacy is obtained. When $\epsilon = \infty$ (or a very big number), no noise is added and the result is the baseline model (nothing else is revealed about the input data). The published model θ is a vector of length $k + 1$ (intercept and model coefficients) and includes the DP-noise.

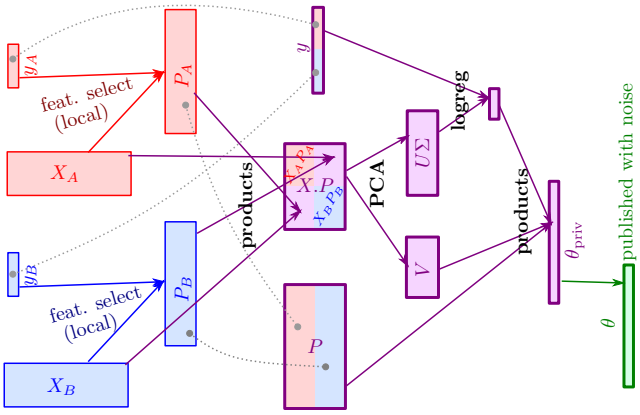

Fig. 3. Logistic regression learning data-flow program.

The procedure to obtain the (noiseless) baseline logistic regression model is the following (depicted on figure 3). The full algorithm is described in Algorithm 1.

Local plaintext computation Each player A, B runs locally on its dataset a features selection procedure and computes an orthonormal projection matrix P_A, P_B, of dimension $k \times h$ where h is the hyper-parameter. These matrices are used in the multi-party computation to project databases to a smaller feature space. This step are the red and blue arrows/blocks in figure 3. Out of all the feature

Algorithm 1 Logistic regression federated learning.

Require: Datasets (X_A, y_A) and (X_B, y_B) owned by A and B, respectively.
Require: A hyperparameter h used for local dimensionality reduction.
Require: Differential privacy parameters (ϵ, δ).
Require: L_2-regularization parameter λ satisfying (3)
Ensure: Revealed logistic regression model θ with (ϵ, δ)-differential privacy noise

Local plaintext computation

1: Players A and B apply principal component analysis locally in order to obtain the decomposition to rank h: $U_A \cdot \Sigma_A \cdot P_A^T := \texttt{PCA}_h(X_A)$ and $U_B \cdot \Sigma_B \cdot P_B^T := \texttt{PCA}_h(X_B)$ respectively an computes $X_A \cdot P_A$ and $X_B \cdot P_B$ resp. (U_A and U_B are $n_A \times h$ and $n_B \times h$ resp., Σ_A and Σ_B are positive diagonal matrices of size h, P_A and P_B are $k \times h$ matrices).

Multi-party computation

2: Players A and B secret shared X_A, y_A, P_A, $X_A \cdot P_A$, X_B, y_B, P_B and $X_B \cdot P_B$ resp. Let $X = \begin{pmatrix} X_A \\ X_B \end{pmatrix}$, $y = \begin{pmatrix} y_A \\ y_B \end{pmatrix}$ and $P = \begin{pmatrix} P_A & P_B \end{pmatrix}$.

3: Compute $X \cdot P = \begin{pmatrix} X_A \cdot P_A & X_A \cdot P_B \\ X_B \cdot P_A & X_B \cdot P_B \end{pmatrix}$, using the locally pre-computed $X_A \cdot P_A$ and $X_B \cdot P_B$.

4: Apply principal component analysis in order to obtain the decomposition $U \cdot \Sigma \cdot V^T := \texttt{PCA}_h(X \cdot P)$.

5: **return** $\theta = P \cdot V \cdot \texttt{logreg}_{\lambda,b}(U \cdot \Sigma, y)$ where b is a perturbation vector satisfying either (4) for (ϵ, δ)-differential privacy with Gaussian noise or (5) for ϵ-differential privacy with Laplacian noise.

selection algorithms we have tried, the best results are obtained using a simple PCA (principal component analysis).

Multi-party computation Let (X, y) denote the joint datasets, X is the $(n_A + n_B) \times k$ features matrix, y is the classes vector of size $(n_A + n_B)$ and let P be the joint projection matrix of dimension $k \times 2 \cdot h$. A new PCA is performed, this time in MPC settings.The benefits of the PCA are two-fold:

(i) it removes correlated features from the projected joint dataset and reduces the dimension of the feature space of the data,
(ii) the change of variables allows us to work on a dataset with orthonormal columns, which significantly increases the performance and stability of the algorithm.

Let $U \cdot \Sigma \cdot V^T := \texttt{PCA}_h(X \cdot P)$ be the PCA decomposition of $X \cdot P$ to rank h. Here, U is $n \times h$ matrix whose columns form an orthonormal family, Σ is a diagonal matrix of size h with positive diagonal entries and V is $2 \cdot h \times h$ matrix.

To add differential privacy to the model, we have two choices:

output perturbation – add noise to the exact model post-optimization,
objective perturbation – add noise to the cost function pre-optimization and optimize for the perturbed cost function.

As summarized in [38, p.25.3] and following Dwork [28], for logistic regression cost functions, objective perturbation achieves a prescribed level (ϵ, δ) of differential privacy with smaller distortion to the minimizer compared to output perturbation.

For objective perturbation, we choose to add differential privacy with respect to the PCA-reduced model and input feature matrix. More precisely, letting $b \in \mathbf{R}^h$ be a perturbation vector parameter, the objective perturbation of the cost function in our setting is

$$L(\theta, \lambda, b) := \frac{1}{n} \sum_{i=1}^{n} \ell\left((U \cdot \Sigma)_i, y_i, \theta\right) + \frac{\lambda}{2n}\|\theta\| + \frac{1}{n}\langle b, \theta\rangle,$$

and the minimum for that perturbed function is

$$\texttt{logreg}_{\lambda,b}(U \cdot \Sigma, y) := \arg\,\min_\theta L(\theta, \lambda, b).$$

Here, $\ell(W_i, y_i, \theta)$ is the negative log-likelihood cost function:

$$\ell(W_i, y_i, \theta) = -y_i \ln(\sigma(W_i \cdot \theta)) - (1 - y_i)\ln(1 - \sigma(W_i \cdot \theta)),$$

where $W_i \in \mathbf{R}^h$ is the sample and $y_i \in \{0, 1\}$ is the class label. Note the presence of both the L_2-regularization term with parameter λ and a perturbation term written in terms of b. Both λ and b will be determined in terms of the differential privacy parameters ϵ and δ as well as the input matrix X.

The noiseless (baseline) model is

$$\theta_{\text{priv}} = P \cdot V \cdot \texttt{logreg}_{\lambda, b=0}(U \cdot \Sigma, y).$$

Detailed description of the logistic regression in the MPC `Manticore` framework with $b = 0$ is presented in [16, §5.1]. The MPC steps are the purple arrows on Figure 3.

For any θ, the maximal norm among all gradients of individual samples, $G_{\max}(\theta) = \max_i \left|\nabla \ell\left((U \cdot \Sigma)_i, y_i, \theta\right)\right|$ satisfies

$$G_{\max}(\theta) \leq \sqrt{h} \cdot \|U \cdot \Sigma\|_\infty \leq \sqrt{h} \cdot \|X\|_\infty \tag{1}$$

and the hessian spectral norm, $H_{\max}(\theta) = \max_i \left|\nabla^2 \ell\left((U \cdot \Sigma)_i, y_i, \theta\right)\right|$ satisfies

$$H_{\max}(\theta) \leq \frac{h}{4} \cdot \|U \cdot \Sigma\|_\infty^2 \leq \frac{h}{4} \cdot \|X\|_\infty^2. \tag{2}$$

Letting $G_{\max} := \sup_\theta G_{\max}(\theta)$ and $H_{\max} := \sup_\theta H_{\max}(\theta)$, Kifer et al [38] prove that for any regularization $\lambda \geq 2H_{\max}/\epsilon$, one achieves:

ϵ-differential privacy using a Laplacian perturbation to the objective function with b satisfying:

$$\text{density}(b) \propto \exp\left(-\frac{\epsilon\|b\|}{2G_{\max}}\right)$$

(ϵ, δ)**-differential privacy** for a perturbation vector b satisfying:

$$b \leftarrow \mathcal{N}\left(0, \text{variance} = \frac{8G_{\max}^2 \log(2/\delta + 4\epsilon)}{\epsilon^2}\right),$$

In particular, combining with the bounds (1) and (2), Kifer's result implies that for

$$\lambda \geq \frac{h}{2\epsilon} \cdot \|X\|_\infty^2 \tag{3}$$

and

$$b \leftarrow \mathcal{N}\left(0, \text{variance} = \frac{8h\|X\|_\infty^2 \log(2/\delta + 4\epsilon)}{\epsilon^2}\right) \tag{4}$$

we achieve the desired level (ϵ, δ) of differential privacy. Similarly, if we want to achieve ϵ-differential privacy, it suffices to consider a Laplacian perturbation satisfying

$$\text{density}(b) \propto \exp\left(-\frac{\epsilon\|b\|}{2\sqrt{h}\|X\|_\infty}\right). \tag{5}$$

The GenoPPML framework model learning phase can be easily extended in order fine-tune the model using different values for hyperparameters such as h and λ. One would execute several (parallel) learning steps and only the best performing model, as a function of the model cost function or any other model quality assertion metric, will be revealed.

Implementation details The learning phase is built upon Inpher XOR library[6], which is an implementation of the Manticore [16] framework. This framework provides a virtual machine allowing multiple players to execute an MPC algorithm (one virtual machine per player). The built-in compiler included in the XOR framework is particularly optimized for stable fixed point computations, such as regression algorithms, and matrix algebra. Once the computation has been set-up (i.e. compilation of a public algorithm, and data-independent setup that depends only on public dimensions and hyper-parameters), the computation phase is carried-out in peer-to-peer manner between the data owners, in the *honest-but-curious full threshold* security model, i.e. no information other than the final result is revealed without the explicit collusion of all the players.

The combined action of PCA dimension reduction, and of the L_2-regularization of logistic regression make the overall algorithm numerically stable, robust against singular features matrices, and resilient against over-fitting. High precision is obtained via a uniform approximation of the sigmoid function via Fourier series based on the ideas of [13]. Because MPC operates over the full dataset, the algorithm is stable even if the datasets of player A and B are not independent and identically distributed.

Our protocol does not reveal any intermediate variables, and in particular, no partial gradients are published. This allows to choose an algorithm that goes

[6] More information about Inpher XOR library can be found here `https://www.inpher.io/products#xor`.

"straight to the point", like gradient descents over *the full dataset*. Because MPC does not restrict the choice of aggregation function, we use accelerated convergence methods, the *IRLS (Newton–Raphson)* technique via the inverse Hessian, that make logistic regression converge in 8-10 iterations. This is much smaller than any stochastic gradient descent (SGD), and/or mini-batched approaches, that require many more iterations to converge.

Discussion The learning phase described above is a hybridization between a pure MPC and an FL+DP approach. In a pure MPC solution, the data owners secret share the full joint dataset (X, y) without projecting it using the PCA matrix and thus, perform the MPC logistic regression training on a larger matrix: $(n_1 + \cdots + n_\ell) \times k$ instead of $(n_1 + \cdots + n_\ell) \times l \cdot h$ (here, l is the number of data owners). Considering that for genomic datasets $l \cdot h$ is usually several magnitudes lower than k, our framework has a huge computational advantage over a pure MPC approach. The drawback is that our approach supposes that the input dataset features are correlated and that a projection over a lower-dimensional space preserves input dataset information. Note that this limitation does not apply to genomic datasets because their feature space is highly correlated.

The only data that is published in our framework is the final model, and it is the only variable that need to be protected by differential privacy noise, the privacy of all other intermediate variables being protected by MPC in the information-theoretic security model. The privacy is the same as if the whole computation had been carried out by an oracle that just publishes the final result, which is the strongest notion of privacy achievable. In other words, the use of MPC compared to a federated learning approach allows us to:

1. use noise-less iterations during the gradient descent (much fewer rounds than solutions that require noise at each step),
2. apply rigorous one-step privacy budget bounds provided in the foundation papers on differential privacy (not heuristic iterative bounds),
3. derive all regularization and noise parameters directly from the provided ϵ and/or δ privacy levels.

As a final bonus: setting $\epsilon = \infty$ in our solution reveals only the baseline θ_{priv} model, but no other information on X_A, y_A, X_B, y_B. This is much better than traditional plaintext-aggregation-based federated learning, where $\epsilon = \infty$ would leak all the (noiseless) intermediate gradients, which carry a lot of additional information about the individual datasets, as explained in [33,61].

The convergence of our hybrid method versus a plaintext-aggregation FL approach is illustrated in figure 4. Traditional FL that publishes local updates have a few drawbacks:

– Partial gradients that only depend on dataset A leak information on A, to mitigate the privacy impact, they must be protected with larger noise and reduced learning rate: both increase the number of iterations, and may affect the final accuracy.

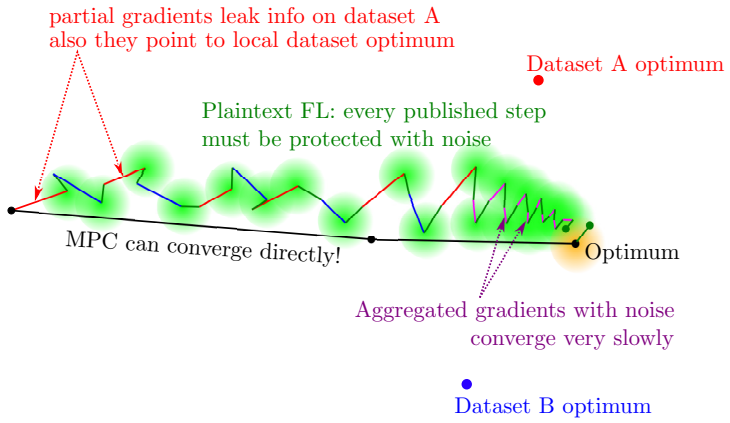

Fig. 4. Compared convergence: MPC versus plaintext-aggregation FL.

– Alternating partial updates on A and B will only converge to the barycenter of datasets A and B optima: reaching the real optimum requires advanced aggregation of partial gradients with noise, which further increase the number of iterations. In contrast, our solution of a pure MPC technique converge directly to the optimum and the noise is added at the end.

3.2 Prediction phase

Once the analyst party obtains the learned model, it can use it to do predictions for the final users. The prediction is a 3 step process:

– User encrypts and sends feature row-vector $[X]_{HE_{sk}}$ to the analyst party.
– Analyst computes the model prediction function pred_θ over X and sends the prediction result $[\text{pred}_\theta(X)]_{HE_{sk}}$ back to the user.
– User decrypts the prediction result using his secret key sk.

User data is homomorphically encrypted using user secret-key sk. The analyst party has never access in clear either to user data X, to prediction result or to any intermediate computation values.

The plain-text logistic regression prediction function is the sign of a matrix product:

$$\text{pred}_\theta(X) = \texttt{sign}(X \cdot \theta).$$

We consider that X has a constant 1 column appended which corresponds to model intercept component. In practice, no multiplication between model intercept coefficient and this column is done. Without loss of generality, we suppose that model θ has integer coefficients.

Data encryption Let user data X be a row-vector size k where k is the number of features. The row-vector X is split into $d = \lceil k/n \rceil$ contiguous parts, here n is the number of coefficients in the homomorphic encryption scheme TRLWE polynomial. Let $F^{(j)} = \sum_{i=0}^{n-1} X_{j \cdot n+i} \cdot Z^i$ be its j-th part. Each part $F^{(j)}$, $0 \leq j < d$ is encrypted in a TRLWE ciphertext using the coefficient packing strategy.

Since TRLWE ciphertexts encrypt polynomials with Torus coefficients, the part-polynomial $F^{(j)}$ is rescaled by an analyst-provided factor. The scale factor is chosen in such a way that the prediction function does not overflow modulo 1. In practice, a 25% error-margin is used.

Encrypted prediction The learned model θ is available in plain to the analyst party. Prediction is performed by a mix of linear combination and functional bootstrapping. Algorithm 2 illustrates our approach. The loop performs the dot product by blocks of n values. On line 3, polynomial $T^{(j)}$ encodes a part of n model coefficients in reverse order. Observe that the result of homomorphic multiplication on line 4 encrypts a polynomial whose highest-degree coefficient is a part of dot product $X \cdot \theta$, in particular coefficient $n-1$ is:

$$\sum_{i=0}^{n-1} X_{j \cdot n+i} \cdot \theta_{j \cdot n+i}.$$

Other coefficients are dot products with different rotations of $T^{(j)}$ and are not used. After the loop, the $(n-1)$-th coefficient of variable P encodes the full dot product $X \cdot \theta$. This part of the algorithm uses d plaintext-ciphertext multiplications and $d-1$ ciphertext additions.

Afterwards, line 7, a TLWE encryption of $X \cdot \theta$ is extracted from the TRLWE ciphertext P. In this way, a single coefficient encryption is obtained and the unnecessary coefficients of P are discarded. A functional bootstrapping procedure, line 8, evaluates the `sign` function over $X \cdot \theta$.

Implementation details The prediction phase has been implemented using the open source TFHE library[7]. We use TRLWE and TLWE samples of size $n = 1024$. The standard deviation $\alpha = 2^{-25}$ is used to sample the initial Gaussian noise in the TRLWE ciphertexts. The secret key is binary. These parameters achieve at least 128 bits of security, according to LWE estimator [2]. The library is used unmodified except for the TLWE sample size n.

Discussion For performance reasons, one could perform only the dot product part on the analyst side and let the final user do the sign part. The downside is that more information about model will leak to user, refer to [5,56,6,51,17] for different attack vectors exploiting this. A possible solution will be to add another layer of DP noise to prediction results before sending them back to user.

[7] TFHE library is available at `https://github.com/tfhe/tfhe`.

Algorithm 2 Homomorphic evaluation of logistic regression prediction.

Require: Encrypted parts $\mathsf{TRLWE}\left(F^{(j)}\right)$ where $F^{(j)} = \sum_{i=0}^{n-1} X_{j \cdot n+i} \cdot Z^i$ – for $0 \leq j < d$
Require: Model coefficients θ_i, for $0 \leq i < k$
Ensure: Encrypted prediction $\mathsf{TLWE}\left(\mathtt{sign}\left(X \cdot \theta\right)\right)$

$P \leftarrow 0$
for $j = 0, \ldots, d-1$ **do**
  $T^{(j)} = \sum_{i=0}^{n-1} \theta_{j \cdot n+n-1-i} \cdot Z^i$
  $tmp \leftarrow \mathsf{Mult}\left(T^{(j)}, \mathsf{TRLWE}\left(F^{(j)}\right)\right)$
  $P \leftarrow \mathsf{Add}\left(P, tmp\right)$
end for
$y \leftarrow \mathsf{CoefExtr}_{n-1}\left(P\right)$
return $\mathsf{FuncBoot}_{\mathtt{sign}}\left(y\right)$

4 Benchmarks

In our experiments, we use the BC-TCGA dataset [59,45]. The dataset contains gene expression data for 17,814 genes and 590 samples (including 61 normal tissue samples and 529 breast cancer tissue samples). The BC-TCGA dataset is highly unbalanced towards positive samples making learning task theoretically harder. Gene expression value measures the intensity of a gene in a given sample. Depending on the sequencing type, gene expression data has different ways to be measured (e.g. normalized intensity values or normalized ratios between measured and control sample intensities).

Our objective is to predict if a given tissue sample, described by its gene expression values, is cancerous or not. We split the dataset (keeping a fixed ratio between normal cases and tumor cases) into 3 parts: A, B and C. A joint logistic regression model is trained between 2-parties, datasets A and B respectively, to answer this question. Dataset C is used to for model testing purposes.

4.1 Learning phase

We run all tests 10 times and we compute the ROC AUC score (Area Under the Receiver Operating Characteristic Curve) which is well suited for unbalanced datasets. We use three values (5, 15 and 25) for the hyperparameter h (i.e., the projection dimension for the local datasets) in our tests. As the DP-noise is random, the model quality varies between each run. The mean AUC score over the 10 runs is illustrated in figure 5. The best AUC score, between 0.99 and 1.0, is obtained for $h = 5$. The privacy budget ϵ varies from 3 to 90 and $\delta = 0$ (ϵ-DP). Better models are obtained when the weaker (ϵ, δ)-DP is used. In the case of $h = 15$ and 25, the AUC score is worse because the added DP-noise is larger.

In table 1 the training execution times, memory usage and network communication size are given. The tests were performed on an Intel Core i7 1065G7

CPU @ 1.30GHz machine with 16GB of RAM. Total training time (including local preprocessing phase and MPC phase) is under 5 seconds. The total training time does not count for network communication because we execute player A and B virtual machines on one host machine. The communication time for $h = 25$ case is less than 2 seconds for a LAN network (1Gbps) and 15 seconds for a WAN network (100Mbps).

	Hyper-parameter h	5	15	25
Local processing	CPU	2.26	2.75	2.94
	RAM	280	281	285
MPC	CPU	0.76	0.93	1.26
	RAM	423	438	476
	Network	144	162	181

Table 1. Execution times (in seconds per player), RAM usage and network communication (in MB per player).

4.2 Prediction phase

The TFHE scheme instantiated with parameters described earlier has the following data sizes:

- secret-key – 128B,
- functional boostrapping key – 48MB
- TLWE ciphertext – 4kB
- TRLWE ciphertext – 8kB

The analyst needs a functional bootstrapping key for each final user. An user enrollment phase is performed during which the analyst receives the 48MB key. User enrollment is a one-time process. Uploading the bootstrapping key requires less than 4 seconds on a WAN network (100Mbps).

In order to encrypt an user genomic data vector (17,814 values) we need $d = 18$ TRLWE ciphertexts. The request size the analyst receives from user is of only $144kB$ ($= d \times$ TRLWE ciphertext size). Ciphertext expansion factor is 8 for the request, which is low in an homomorphic encryption context. Once the analyst has executed the prediction algorithm it sends the prediction result of size 4kB (a TLWE ciphertext) as response to the final user. Prediction algorithm execution is very fast and it takes less than 0.1 seconds on the same machine as before. Encryption and decryption times are instantaneous (≈ 0.01 seconds).

In the prediction phase we have tested a logistic regression model with hyperparameter $h = 5$. Using another model will make no difference in execution time because the prediction phase uses same size inputs. We have evaluated the HE prediction on part C of the split dataset. Even though the logistic regression model has been scaled-up/rounded to integer coefficients and features were

noisy due to HE, the accuracy of the prediction did not change when compared to plaintext version. We have also tested the prediction accuracy of other models (with $h = \{15, 25\}$ and two types of differential privacy) and no significant changes were observed in the AUC score.

5 Conclusions and perspectives

In this paper, we have introduced GenoPPML; an end-to-end framework for privacy-preserving machine learning applied to genomic data. Multi-party computation is used for model training over datasets from several private genomic database owners. A differential private joint model is revealed to an analyst party. No other information about the private genomic databases is revealed. The analyst uses this model to provide prediction service on homomorphically encrypted genomic data samples. We have shown that MPC and HE techniques are practical for genomic data machine learning use-cases and that GenoPPML framework is particularly well suited for large feature-space data, like genetic datasets.

The BC-TCGA dataset is used in our experiments to train a breast cancer prediction model (logistic-regression). The obtained model is highly-accurate and training times are low. Even if prediction over homomorphically encrypted data is used, the prediction phase is almost instantaneous and can be used in high-throughput applications. The GenoPPML framework is not limited to logistic-regression models only. More complex machine learning algorithms can be used at a higher computational cost for the training and the prediction phases.

One weak point of the GenoPPML is that model accuracy intrinsically worsens with the increase of the projection size (refer to database owners local preprocessing); which is due to the fact that differential privacy noise amplitude is proportional to the projection size. In a future work, we envisage to get rid of the differential privacy part for the joint model protection. The joint model shall stay encrypted after the learning phase and some kind of proxy-reencryption shall be used to switch from model encryption domain to user encryption domain.

References

1. 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)
2. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. Journal of Mathematical Cryptology **9**(3), 169–203 (2015)
3. Alipanahi, B., Delong, A., Weirauch, M.T., Frey, B.J.: Predicting the sequence specificities of dna-and rna-binding proteins by deep learning. Nature biotechnology **33**(8), 831–838 (2015)
4. Aono, Y., Hayashi, T., Trieu Phong, L., Wang, L.: Scalable and secure logistic regression via homomorphic encryption. In: Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy. pp. 142–144 (2016)

5. Ateniese, G., Mancini, L.V., Spognardi, A., Villani, A., Vitali, D., Felici, G.: Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. International Journal of Security and Networks **10**(3), 137–150 (2015)
6. Backes, M., Berrang, P., Humbert, M., Manoharan, P.: Membership privacy in microrna-based studies. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 319–330 (2016)
7. Beaver, D.: Efficient multiparty protocols using circuit randomization. In: Annual International Cryptology Conference. pp. 420–432. Springer (1991)
8. Bell, J.H., Bonawitz, K.A., Gascón, A., Lepoint, T., Raykova, M.: Secure single-server aggregation with (poly) logarithmic overhead. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. pp. 1253–1269 (2020)
9. Biasse, J.F., Ruiz, L.: Fhew with efficient multibit bootstrapping. In: International Conference on Cryptology and Information Security in Latin America. pp. 119–135. Springer (2015)
10. Bogdanov, D., Laur, S., Willemson, J.: Sharemind: A framework for fast privacy-preserving computations. In: European Symposium on Research in Computer Security. pp. 192–206. Springer (2008)
11. Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H.B., Patel, S., Ramage, D., Segal, A., Seth, K.: Practical secure aggregation for privacy-preserving machine learning. In: proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 1175–1191 (2017)
12. Bonte, C., Vercauteren, F.: Privacy-preserving logistic regression training. BMC medical genomics **11**(4), 13–21 (2018)
13. Boura, C., Chillotti, I., Gama, N., Jetchev, D., Peceny, S., Petric, A.: High-precision privacy-preserving real-valued function evaluation. In: International Conference on Financial Cryptography and Data Security. pp. 183–202. Springer (2018)
14. Boura, C., Gama, N., Georgieva, M., Jetchev, D.: Chimera: Combining ring-lwe-based fully homomorphic encryption schemes. Journal of Mathematical Cryptology **14**(1), 316–338 (2020)
15. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. ACM Transactions on Computation Theory (TOCT) **6**(3), 1–36 (2014)
16. Carpov, S., Deforth, K., Gama, N., Georgieva, M., Jetchev, D., Katz, J., Leontiadis, I., Mohammadi, M., Sae-Tang, A., Vuille, M.: Manticore: Efficient framework for scalable secure multiparty computation protocols. Tech. rep., Cryptology ePrint Archive, Report 2021/200 (2021)
17. 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)
18. Carpov, S., Gama, N., Georgieva, M., Troncoso-Pastoriza, J.R.: Privacy-preserving semi-parallel logistic regression training with fully homomorphic encryption. BMC Medical Genomics **13**(7), 1–10 (2020)
19. Carpov, S., Izabachène, M., Mollimard, V.: New techniques for multi-value input homomorphic evaluation and applications. In: Cryptographers' Track at the RSA Conference. pp. 106–126. Springer (2019)
20. Carpov, S., Tortech, T.: Secure top most significant genome variants search: idash 2017 competition. BMC medical genomics **11**(4), 47–55 (2018)

21. Cheon, J.H., Kim, A., Kim, M., Song, Y.: Homomorphic encryption for arithmetic of approximate numbers. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 409–437. Springer (2017)
22. Chillotti, I., Gama, N., Georgieva, M., Izabachene, M.: Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In: international conference on the theory and application of cryptology and information security. pp. 3–33. Springer (2016)
23. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Faster packed homomorphic operations and efficient circuit bootstrapping for tfhe. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 377–408. Springer (2017)
24. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Tfhe: fast fully homomorphic encryption over the torus. Journal of Cryptology **33**(1), 34–91 (2020)
25. Cho, H., Wu, D.J., Berger, B.: Secure genome-wide association analysis using multiparty computation. Nature biotechnology **36**(6), 547–551 (2018)
26. Deforth, K., Desgroseilliers, M., Gama, N., Georgieva, M., Jetchev, D., Vuille, M.: Xorboost: Tree boosting in the multiparty computation setting. Cryptology ePrint Archive, Report 2021/432 (2021), `https://eprint.iacr.org/2021/432`
27. Ducas, L., Micciancio, D.: Fhew: bootstrapping homomorphic encryption in less than a second. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 617–640. Springer (2015)
28. Dwork, C., Gopalan, P., Lin, H., Pitassi, T., Rothblum, G., Smith, A., Yekhanin, S.: An analysis of the chaudhuri and monteleoni algorithm. Innovations in Computer Science (poster) (2009)
29. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. IACR Cryptol. ePrint Arch. **2012**, 144 (2012)
30. Feng, Z., Xiong, H., Song, C., Yang, S., Zhao, B., Wang, L., Chen, Z., Yang, S., Liu, L., Huan, J.: Securegbm: Secure multi-party gradient boosting. In: 2019 IEEE International Conference on Big Data (Big Data). pp. 1312–1321. IEEE (2019)
31. Gainza, P., Sverrisson, F., Monti, F., Rodolà, E., Bronstein, M., Correia, B.: Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning. Nature Methods **17** (2020)
32. Geiping, J., Bauermeister, H., Dröge, H., Moeller, M.: Inverting gradients - how easy is it to break privacy in federated learning? In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual (2020), `https://proceedings.neurips.cc/paper/2020/hash/c4ede56bbd98819ae6112b20ac6bf145-Abstract.html`
33. Geiping, J., Bauermeister, H., Dröge, H., Moeller, M.: Inverting gradients–how easy is it to break privacy in federated learning? arXiv preprint arXiv:2003.14053 (2020)
34. 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)
35. Jagadeesh, K.A., Wu, D.J., Birgmeier, J.A., Boneh, D., Bejerano, G.: Deriving genomic diagnoses without revealing patient genomes. Science **357**(6352), 692–695 (2017)
36. Keller, M.: MP-SPDZ: A versatile framework for multi-party computation. In: CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security. pp. 1575–1590 (2020)

37. Keller, M., Orsini, E., Scholl, P.: Mascot: faster malicious arithmetic secure computation with oblivious transfer. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 830–842 (2016)
38. Kifer, D., Smith, A.D., Thakurta, A.: Private convex optimization for empirical risk minimization with applications to high-dimensional regression. In: COLT - The 25th Annual Conference on Learning Theory. JMLR Proceedings, vol. 23, pp. 25.1–25.40 (2012)
39. 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**(4), 23–31 (2018)
40. Kim, M., Harmanci, A., Bossuat, J.P., Carpov, S., Cheon, J.H., Chillotti, I., Cho, W., Froelicher, D., Gama, N., Georgieva, M., et al.: Ultra-fast homomorphic encryption models enable secure outsourcing of genotype imputation. bioRxiv (2020)
41. Konečnỳ, J., McMahan, H.B., Ramage, D., Richtárik, P.: Federated optimization: Distributed machine learning for on-device intelligence. arXiv preprint arXiv:1610.02527 (2016)
42. Li, W., Milletarì, F., Xu, D., Rieke, N., Hancox, J., Zhu, W., Baust, M., Cheng, Y., Ourselin, S., Cardoso, M.J., et al.: Privacy-preserving federated brain tumour segmentation. In: International Workshop on Machine Learning in Medical Imaging. pp. 133–141. Springer (2019)
43. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence and Statistics. pp. 1273–1282. PMLR (2017)
44. Mohassel, P., Zhang, Y.: SecureML: A system for scalable privacy-preserving machine learning. In: 2017 IEEE Symposium on Security and Privacy (SP). pp. 19–38. IEEE (2017)
45. Network, C.G.A., et al.: Comprehensive molecular portraits of human breast tumours. Nature **490**(7418), 61 (2012)
46. Ohrimenko, O., Schuster, F., Fournet, C., Mehta, A., Nowozin, S., Vaswani, K., Costa, M.: Oblivious multi-party machine learning on trusted processors. In: 25th USENIX Security Symposium. pp. 619–636 (2016)
47. Patra, A., Suresh, A.: BLAZE: blazing fast privacy-preserving machine learning. In: 27th Annual Network and Distributed System Security Symposium, NDSS (2020)
48. Pullonen, P., Siim, S.: Combining secret sharing and garbled circuits for efficient private ieee 754 floating-point computations. In: International Conference on Financial Cryptography and Data Security. pp. 172–183. Springer (2015)
49. Sadat, M.N., Al Aziz, M.M., Mohammed, N., Chen, F., Jiang, X., Wang, S.: Safety: secure gwas in federated environment through a hybrid solution. IEEE/ACM transactions on computational biology and bioinformatics **16**(1), 93–102 (2018)
50. Senior, A.W., Evans, R., Jumper, J., Kirkpatrick, J., Sifre, L., Green, T., Qin, C., Žídek, A., Nelson, A.W., Bridgland, A., et al.: Improved protein structure prediction using potentials from deep learning. Nature **577**(7792), 706–710 (2020)
51. Shokri, R., Stronati, M., Song, C., Shmatikov, V.: Membership inference attacks against machine learning models. In: 2017 IEEE Symposium on Security and Privacy (SP). pp. 3–18. IEEE (2017)
52. Singh, K., Sirdey, R., Artiguenave, F., Cohen, D., Carpov, S.: Towards confidentiality-strengthened personalized genomic medicine embedding homomorphic cryptography. In: ICISSP. pp. 325–333 (2017)
53. Singh, K., Sirdey, R., Carpov, S.: Practical personalized genomics in the encrypted domain. In: 2018 Third International Conference on Fog and Mobile Edge Computing (FMEC). pp. 139–146. IEEE (2018)

54. So, J., Güler, B., Avestimehr, A.S.: Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning. IEEE Journal on Selected Areas in Information Theory **2**(1), 479–489 (2021)
55. Stokes, J.M., Yang, K., Swanson, K., Jin, W., Cubillos-Ruiz, A., Donghia, N.M., MacNair, C.R., French, S., Carfrae, L.A., Bloom-Ackermann, Z., et al.: A deep learning approach to antibiotic discovery. Cell **180**(4), 688–702 (2020)
56. Tramèr, F., Zhang, F., Juels, A., Reiter, M.K., Ristenpart, T.: Stealing machine learning models via prediction apis. In: 25th {USENIX} Security Symposium ({USENIX} Security 16). pp. 601–618 (2016)
57. Wagh, S., Gupta, D., Chandran, N.: SecureNN: 3-party secure computation for neural network training. Proceedings on Privacy Enhancing Technologies **2019**(3), 26–49 (2019)
58. Wang, Y., Klijn, J.G., Zhang, Y., Sieuwerts, A.M., Look, M.P., Yang, F., Talantov, D., Timmermans, M., Meijer-van Gelder, M.E., Yu, J., et al.: Gene-expression profiles to predict distant metastasis of lymph-node-negative primary breast cancer. The Lancet **365**(9460), 671–679 (2005)
59. Xie, H., Li, J., Jatkoe, T., Hatzis, C.: Gene expression profiles of breast cancer (2017), `http://dx.doi.org/10.17632/v3cc2p38hb.1`
60. Zhu, L., Han, S.: Deep leakage from gradients. In: Yang, Q., Fan, L., Yu, H. (eds.) Federated Learning - Privacy and Incentive, Lecture Notes in Computer Science, vol. 12500, pp. 17–31. Springer (2020). https://doi.org/10.1007/978-3-030-63076-8_2, `https://doi.org/10.1007/978-3-030-63076-8_2`
61. Zhu, L., Han, S.: Deep leakage from gradients. In: Federated Learning, pp. 17–31. Springer (2020)

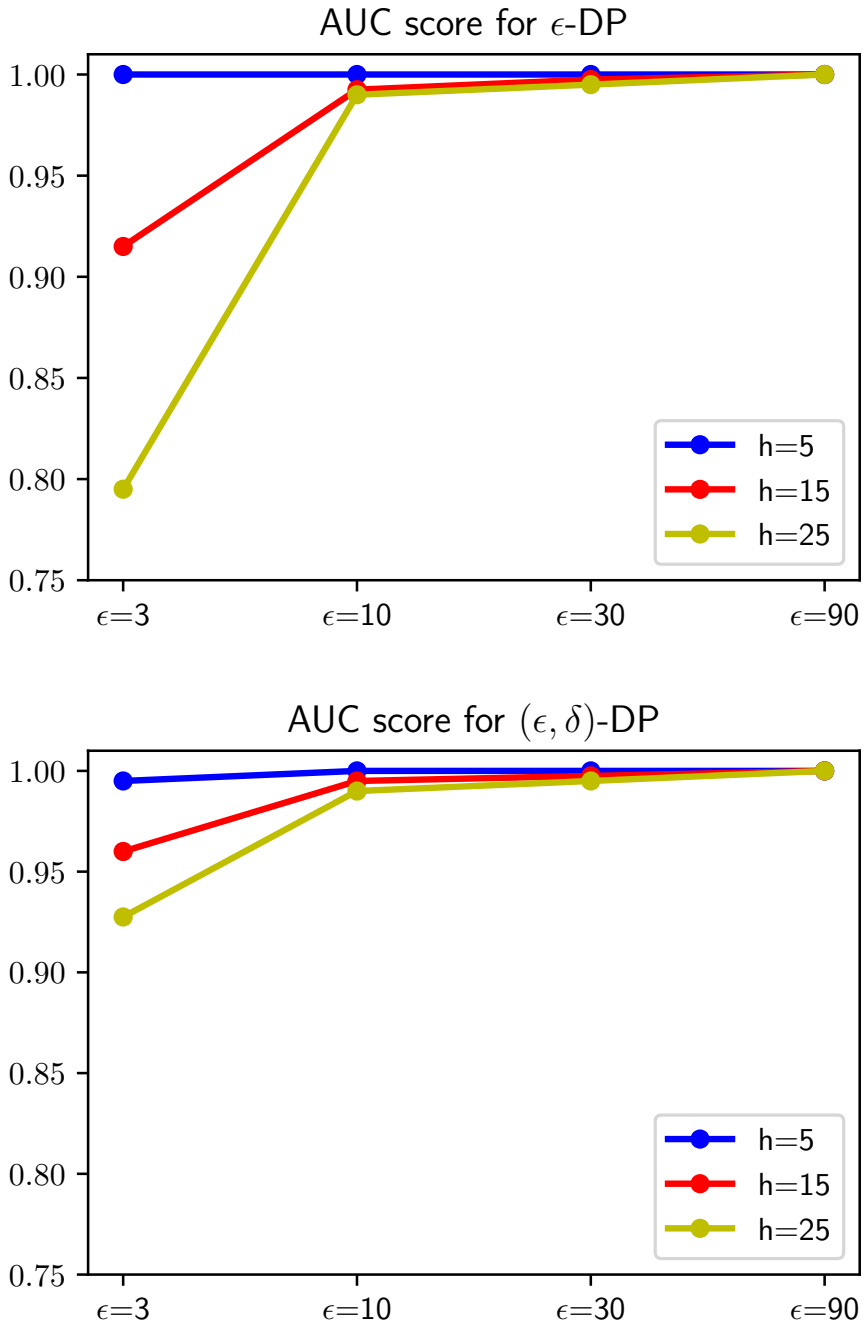

Fig. 5. AUC metric. Upper-side corresponds to ϵ-differential privacy for $\epsilon \in \{3, 10, 30, 90\}$ and lower-side corresponds to (ϵ, δ)-differential privacy for $\epsilon \in \{3, 10, 30, 90\}$ and $\delta = 0.02$. Each line corresponds to a hyper-parameter $h \in \{5, 15, 25\}$.