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. Extensive benchmarks and comparisons to existing works are performed. Our 2-party logistic regression computation is 11× faster than the one in [27] on the same dataset and it uses only a single CPU core.

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 [53], protein functions and interactions [33] as well as drug discovery and molecular design [58].

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

[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`

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.

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), differential privacy (DP) among others.

Track III from the 2020 edition featured a privacy-preserving model learning 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 [61] for details). In the 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 3 genomic datasets from 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 Related work

In this section we review related literature on privacy-preserving machine learning (PPML) with emphasis to the genomic field. An extensive review and analysis of the limitations for privacy enhancing technologies used for genomic data is provided in [45].

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. 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 [36], in settings limited to two parties, or secret sharing and multiparty computation [17], 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 [48,52,21], therefore requiring some degree of trust on the hardware manufacturer, or homomorphic encryption [4,40,55,56,19,41]. 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 [40] 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 [19] 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 [41]. The authors propose several methods, which differ by the employed HE scheme, with different computational performance vs accuracy trade-offs.

The closest related work to ours are De Cock et al. [27] and SecureML [46], which also propose protocols for privately training a logistic regression model based on secure MPC. The SecureML paper proposes a new protocol for ReLu activation function and a fast MPC backend. In [27] the authors optimise the activation function by avoiding the use of full-fledged secure comparison protocols and improve the MPC computations. Like in our solution, [27] and [46]

are split into an offline and online phase. In our solution and also in [27], the offline phase is performed by an additional party (different from the computational parties) which provides so-called multiplication Beaver triples in advance. In [46] this offline phase is executed by same computational parties that run the data dependant (online) phase. To compare the three protocols, we apply the same approach as in [27]: we exclude the data-independent offline stage (different for SecureML protocol), and we keep only the data dependant phases that give us protocols in similar settings. We obtain a speed-up of 11 times for GSE2034 dataset and same execution time for BC-TCGA dataset on a weaker machine and a single-threaded implementation. More details are given in section 5.

3 Background

In this section, we introduce the privacy-enhancing technologies (PETs) and tools needed for GenoPPML.

3.1 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 [10,38,13,49,37]. MPC computations have been used in different domains with sensitive data such as genomic studies [26] and other industry verticals in a variety of use cases including machine learning applications such as linear regressions and logistic regressions [46,13], decision trees [32,28], neural networks [60], etc.

Several MPC protocols and libraries have been proposed in the literature and implemented. `SecureML` [46] 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 [51] 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 [17] is a highly efficient MPC framework operating in semi-honest security model with an offline trusted dealer (TD), with full-threshold security across an arbitrary number of players. The trusted dealer is an additional party (different from the players). The TD generates pre-computed data (also referred to as triplets and masks) and distributes secret-shares of these masks to the players. The verifiability of this phase (TD) can be achieved via standard techniques such as oblivious transfer and cut-and-choose. The TD does not participate in the online phase and thus, it does see neither the local private data, nor the communicated masked data. Furthermore, it does not collude with any of the parties and all communication between the trusted dealer and the players, as well as all communication between the players during the online phase, is end-to-end encrypted. This makes it secure against malicious external

adversaries. By building on top of Manticore, our protocol is secure in the information-theoretic setting.

The protocol and implementation details are described in [17]. 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 [17], as well XGBoost [28] 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 [13,17] 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.

We use Manticore, but our GenoPPML framework is agnostic of the choice of the MPC framework and can thus be implemented on any MPC framework supporting these primitives. In our implementation of GenoPPML protocol the data owners play the role of the players (computational parties) and the trusted dealer is either the analyst party (this work), or an additional independent party or the triplets are interactively generated by the players using technique as the one proposed in [15]. The drawback of the later is that the MPC protocol is slower, but the security model is stronger. More details are given in section 4.

3.2 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 [35]. 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 [31], BGV [16], CKKS [22] and TFHE [23,25]. 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) [23] 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 [29,23] for *gate bootstrapping* and it was further generalized to arbitrary functions in [9,24,20].

4 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 MPC, FHE and differential privacy (DP).

Figure 1 illustrates the GenoPPML framework. Three types of actors are involved in the data processing: (i) genomic dataset owners, (ii) data analyst and

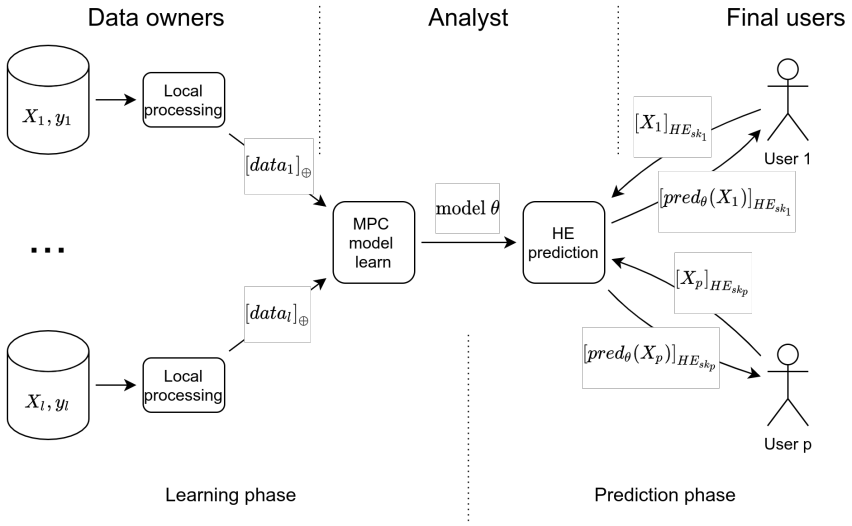

Fig. 1. GenoPPML architecture. Private datasets are processed locally for dimensionality reduction and outputs are secret shared for model training in MPC. The computed model is then used for prediction on users' data encrypted via an FHE scheme. Users receive and decrypt the computed encrypted predictions.

(iii) final users. In a first phase (learning phase), the database owners train a joint logistic regression model (requested by the analyst) via MPC. In order to speed-up the MPC protocol the analyst provides the required Beaver multiplication triplets and does not collude with any sub-set of data owners. A differential privacy mechanism is then used in order to ensure the privacy of the genomic database owners against model inversion attacks [34], [64]. 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.

4.1 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 an 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.

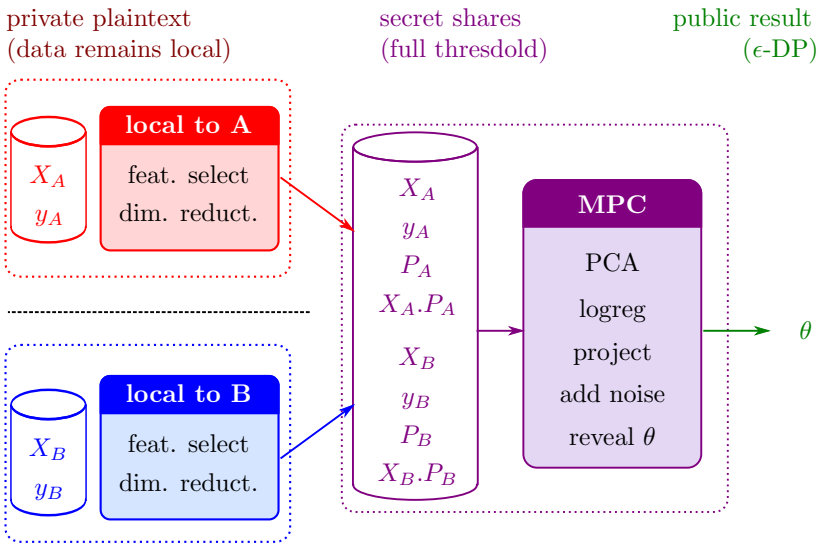

Fig. 2. Example of a 2-party MPC training data-flow chart.

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 n_A and n_B, respectively.

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 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 and 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.

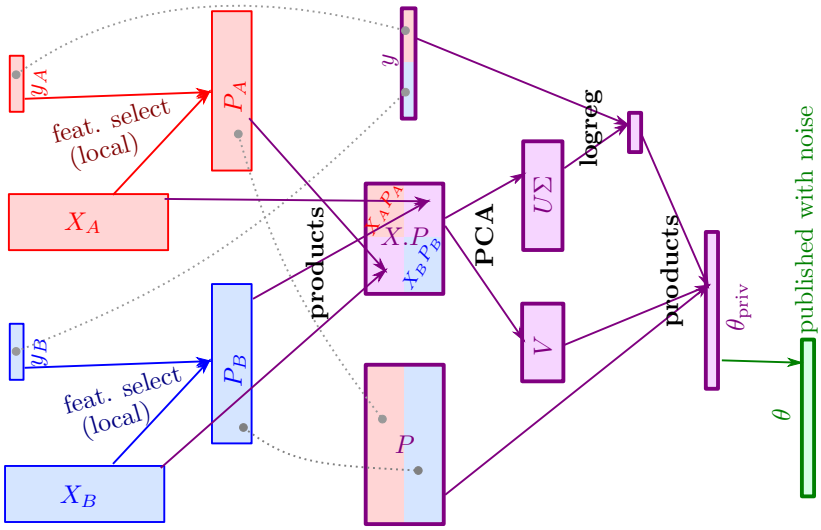

Fig. 3. Logistic regression learning data-flow program.

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 selection algorithms we have tried, the best results are obtained using a simple PCA (principal component analysis). We note that each party can choose its own value for hyperparameter h as a function of dataset characteristics. In this work, we use the same h for both parties.

Algorithm 1 Logistic regression MPC 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 of 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, and compute $X_A \cdot P_A$ and $X_B \cdot P_B$ respectively (U_A and U_B are $n_A \times h$ and $n_B \times h$ matrices, Σ_A and Σ_B are positive diagonal matrices of size h, P_A and P_B are orthogonal $k \times h$ 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 $X_B \cdot P_B$ respectively. 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: Player A and B interactively (via MPC) 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: Player A and B interactively (via MPC) apply principal component analysis in order to obtain the decomposition $U \cdot \Sigma \cdot V^T := \texttt{PCA}_h(X \cdot P)$.

5: Player A and B interactively (via MPC) compute $\theta = P \cdot V \cdot \texttt{logreg}_{\lambda,b}(U \cdot \Sigma, y)$ 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.

6: **return** θ

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 [39, p.25.3] and following Dwork [30], 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 [17, §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 [39] 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 [17] 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. This set-up phase is performed by the analyst.

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 "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: comparison with pure MPC and FL approaches The learning phase described above is a hybridization between a pure MPC, local computations and differential privacy. 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.

Another alternative for the learning phase is a federated learning approach. Federated learning (FL) [42,44] 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

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

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 [64,34]. 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 [1,43], pairwise additive masking [11,8], homomorphic encryption, additive secret sharing [57], have been proposed in the literature.

In our framework the only data that is published 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 [34,64].

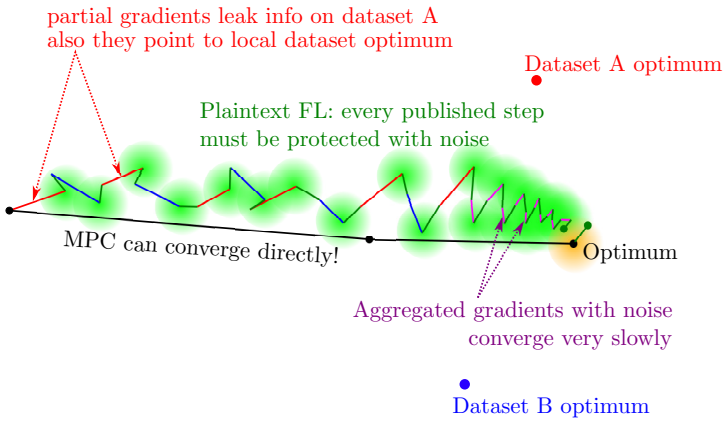

Fig. 4. Compared convergence: MPC versus plaintext-aggregation FL.

The convergence of our 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.
- 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.

4.2 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 $X \cdot \theta$.

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 $\mathsf{TLWE}\left(\mathtt{sign}\left(X \cdot \theta\right)\right)$

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

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.

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

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 [5,59,6,54,18] 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.

5 Benchmarks

In our experiments, we have used 3 datasets from the TCGA (The Cancer Genome Atlas) database, more specifically breast cancer genomic data [62]. The first dataset BC-TCGA contains gene expression data for 17,814 genes and 590 samples (61 normal tissue samples vs 529 breast cancer tissue samples). 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). The second dataset GSE2034 contains gene expression data for 12,634 genes and 286 samples (107 recurrence tumor samples vs 179 no recurrence samples). Lastly, the third dataset, which we denote by BC12-TCGA, contains genomic data (copy number variations) for 25,128 genes and 2713 samples for 11 tumor locations (201 samples of breast tissue vs 2512 samples of other 10 tumor sites). First and last datasets are highly unbalanced towards one output class making learning task theoretically harder. The 3 datasets are representative for different kind of genomic studies, more specifically classification tasks. For more details about refer to [47,**?**].

Our objective is to predict if a given tissue sample, described by its gene expression data, belongs to one of 2 groups (e.g. cancerous or normal for BC-TCGA dataset). We split each dataset (keeping a fixed ratio between the 2 sample groups) into 3 parts: A (40%), B (40%) and C (20%). A joint logistic regression model is trained between 2-parties over datasets A and B. Dataset C is used to for asserting model quality. Training and test dataset sizes are summarized in table 1.

	#features	#samples		neg. to
		train	test	pos. ratio
BC-TCGA	17,814	471	119	0.12
GSE2034	12,634	228	58	0.60
BC12-TCGA	25,128	2,169	544	0.08

Table 1. Train and test dataset sizes together with positive to negative class ratio.

5.1 Learning phase

In our tests, we use 3 values (5, 20 and 50) for the projection dimension h of local datasets and 9 configurations for the DP parameters. The privacy budget ϵ varies

from 3 to 90 and δ is either 0 (ϵ-DP) or 0.02 ((ϵ, δ)-DP). Also we have tested the model with no addition of DP-noise (corresponding to ∞-DP). In order to compare to a clear-text learned model, we have executed the `scikit-learn` library [50] logistic-regression (with default parameters) over the full dataset (concatenation of parts A and B). AUC score (Area Under the Receiver Operating Characteristic Curve), well suited for unbalanced datasets, is used to test model quality. The AUC score varies from 0.5 to 1 (higher is better) and a 0.5 value corresponds to random guessing. Besides, we log the accuracy score (ratio of correct prediction to total number of predictions) for comparison with existing works. The model quality varies at each run because the added DP-noise is random. Each test is executed 10 times. The mean AUC score and the mean accuracy is used in our results.

Figures 5, 6 and 7 illustrate the obtained AUC scores (each has a sub-figure for the ϵ-DP and the (ϵ, δ)-DP experiments). Solid black line is the AUC score for the `scikit-learn` logistic-regression model. Other colored solid lines are the scores for different hyperparameter h values. Dashed lines correspond to AUC score for output models with no differential privacy noise (i.e. ∞-DP).

As expected, for the same value of h, better models are obtained when the weaker (ϵ, δ)-DP notion is used and also when parameter ϵ increases. The `scikit-learn` full-dataset model obtains the best score, except for the BC12-TCGA dataset. We suppose that the default training algorithm in `scikit-learn` logistic regression is not well suited for this dataset and that it needs hyperparameter tuning.

In the case of BC-TCGA dataset the best AUC score, over 0.99, is obtained for $h = 5$ independently of DP-noise configuration. For other values of h the score is worse because the added DP-noise is larger. Observe that a projection of $h = 5$ is sufficient for this dataset as the noiseless DP model obtains the best score.

The situation is different in case of GSE2034 and BC12-TCGA datasets. The projection size $h = 5$ is no more sufficient to capture complete dataset information, the noiseless DP model obtains the lowest score. The 0.5 AUC value for GSE2034 is equivalent to random guessing. We suppose that the maximal projection size ($h = 50$) is not yet sufficient for GSE2034 because the noiseless DP model score is lower than the `scikit-learn` model score. There is no general trend in the choice of hyperparameter h. Under the same DP budget (ϵ value) the best model depends on value h and on dataset characteristics.

Tables 2, 3 and 4 give training execution times, memory usage and network communication size for the executed experiments. The tests were performed on a 4-core Intel(R) Xeon(R) CPU @ 3.10GHz machine with 32GB of RAM (`c2-standard-8` GCP instance). Total training time (local preprocessing phase and MPC phase) in the worst case (highest value of projection parameter h) for the 3 datasets is respectively less than 4, 2 and 23 seconds. It is counter-intuitive that the MPC execution time and the local processing time are comparable, one would expect the MPC phase to take the largest part of execution time. This

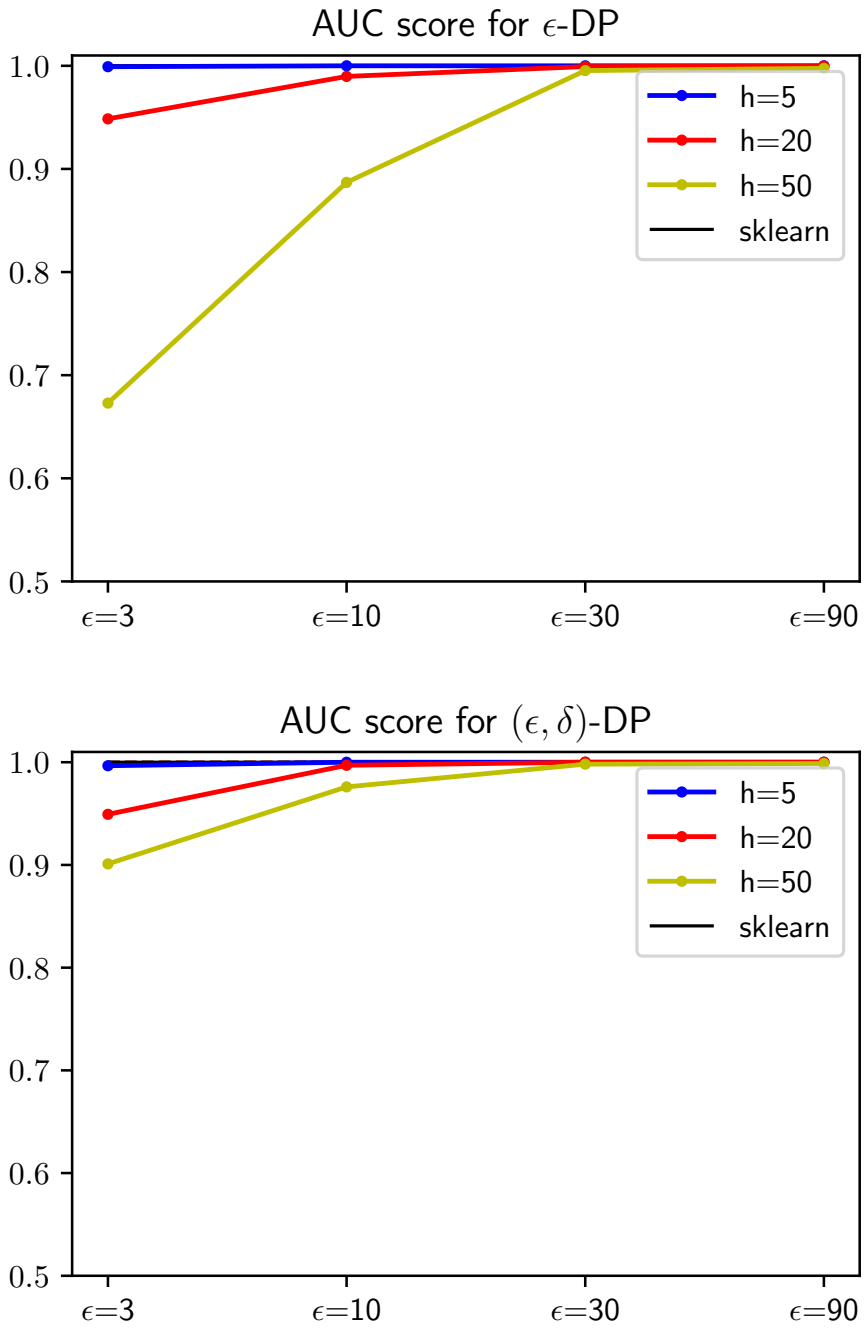

Fig. 5. BC-TCGA dataset AUC scores.

	h	5	20	50
Local processing	CPU	0.49	0.99	1.78
	RAM	266	267	270
MPC	CPU	0.91	1.30	2.17
	RAM	399	437	466
	Network	138	164	219

Table 2. BC-TCGA dataset execution times (in seconds per player), RAM usage and network communication (in MB per player).

	h	5	20	50
Local processing	CPU	0.12	0.29	0.51
	RAM	161	161	161
MPC	CPU	0.37	0.56	1.02
	RAM	167	180	276
	Network	50	68	106

Table 3. GSE2034 dataset execution times (in seconds per player), RAM usage and network communication (in MB per player).

	h	5	20	50
Local processing	CPU	4.10	6.30	11.31
	RAM	1294	1294	1294
MPC	CPU	5.87	7.43	11.47
	RAM	1892	1925	2005
	Network	854	909	1021

Table 4. BC12-TCGA dataset execution times (in seconds per player), RAM usage and network communication (in MB per player).

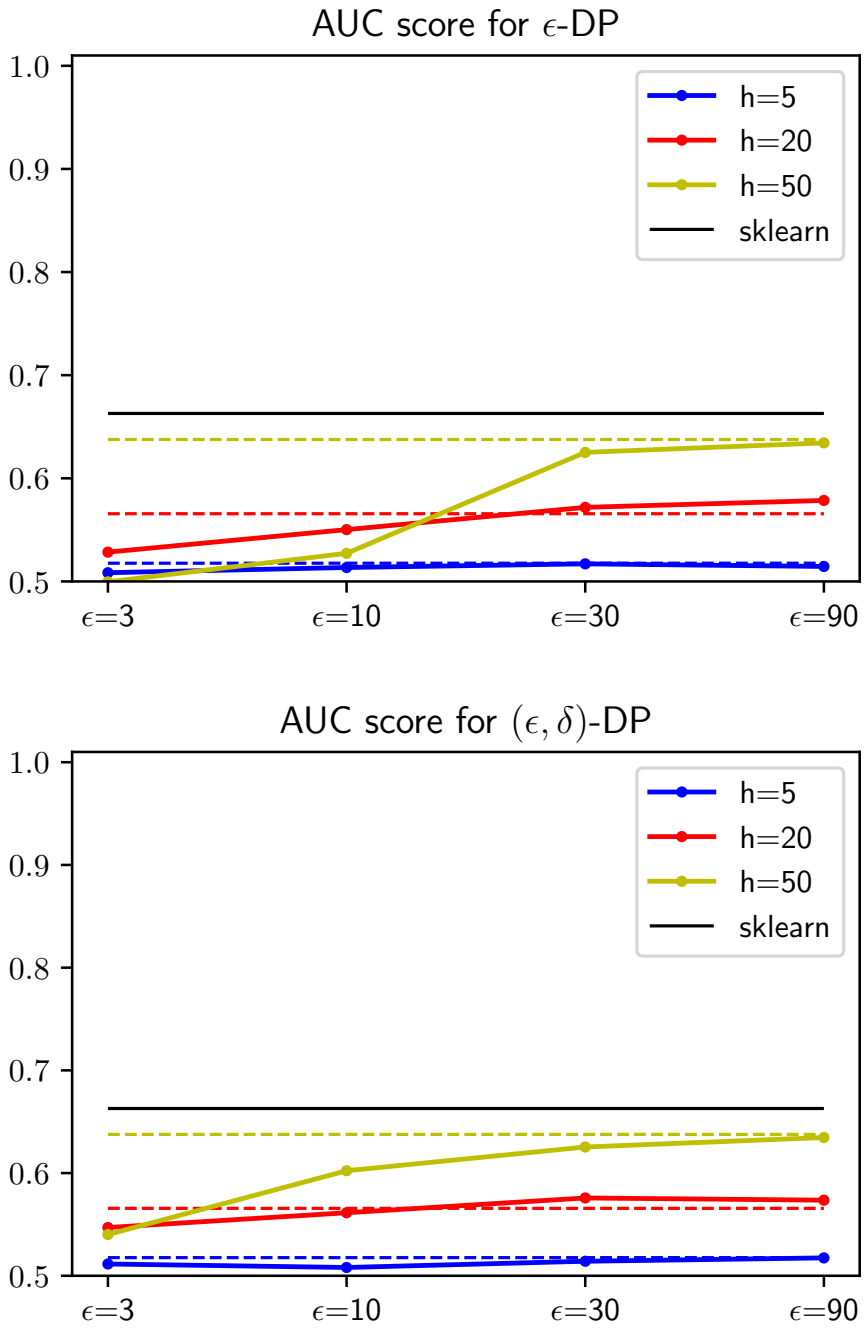

Fig. 6. GSE2034 dataset AUC scores.

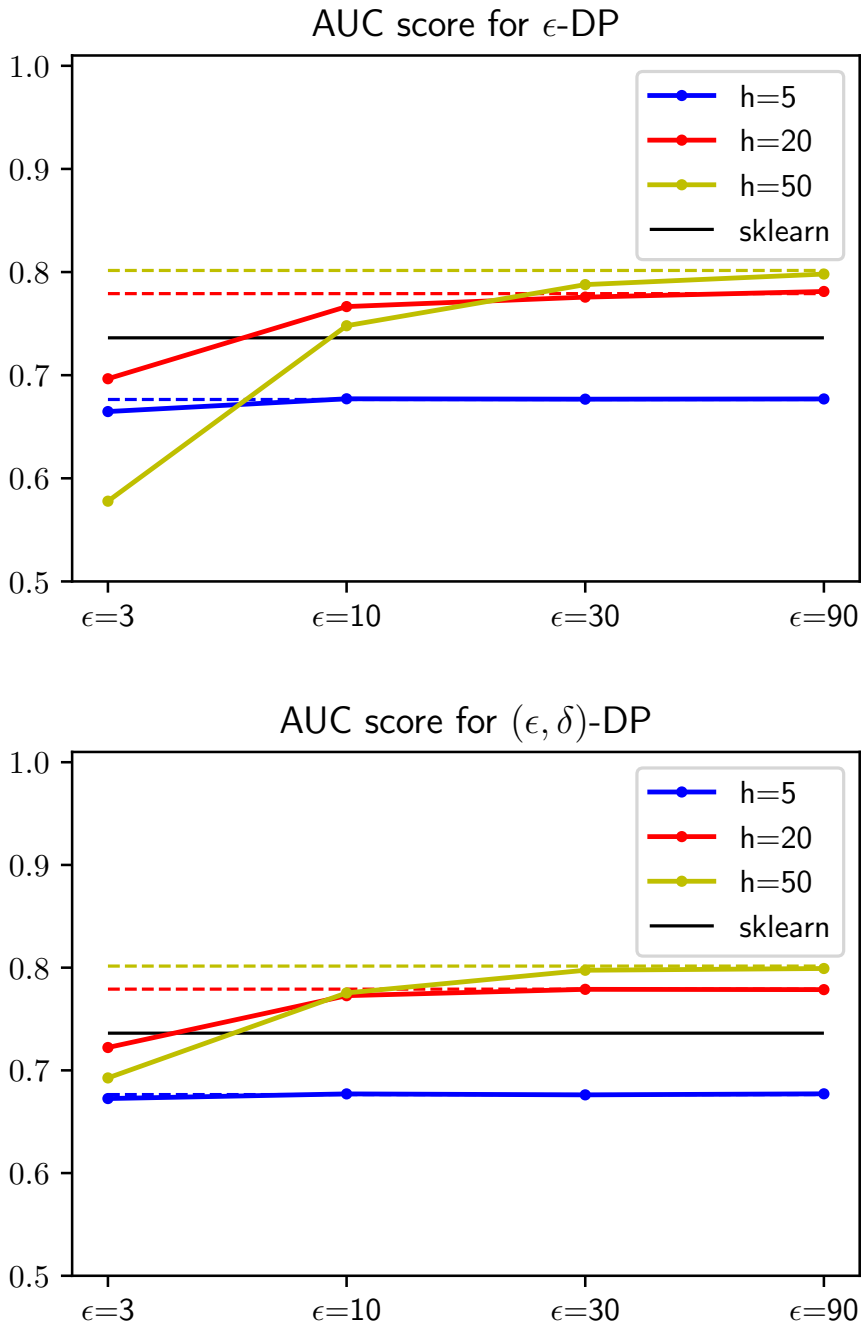

Fig. 7. BC12-TCGA dataset AUC scores.

is due to the fact that the local processing is a python implementation without regard to performance.

The total training time does not count for network communication because we execute player A and B virtual machines on the same host machine. As an example, the communication time for parameter $h = 50$ and dataset BC-TCGA is less than 2 seconds for a LAN network (1Gbps) and ≈ 18 seconds for a WAN network (100Mbps). The communication time goes up to ≈ 82 seconds for the BC12-TCGA dataset in WAN settings.

Figure 8 illustrates the best model accuracy scores obtained for most-secure $(3, 0)$-DP, least-secure $(90, 0.02)$-DP and noiseless ∞-DP settings. In paper [63] the authors compared different dimensionality reduction techniques before the classification task for BC-TCGA and GSE2034 datasets. We obtain comparable accuracy for the BC-TCGA (1.0 vs 0.99) and somewhat better accuracy for the GSE2034 dataset (0.66 vs 0.62). We compare our noiseless DP setting to [63] best obtained accuracy.

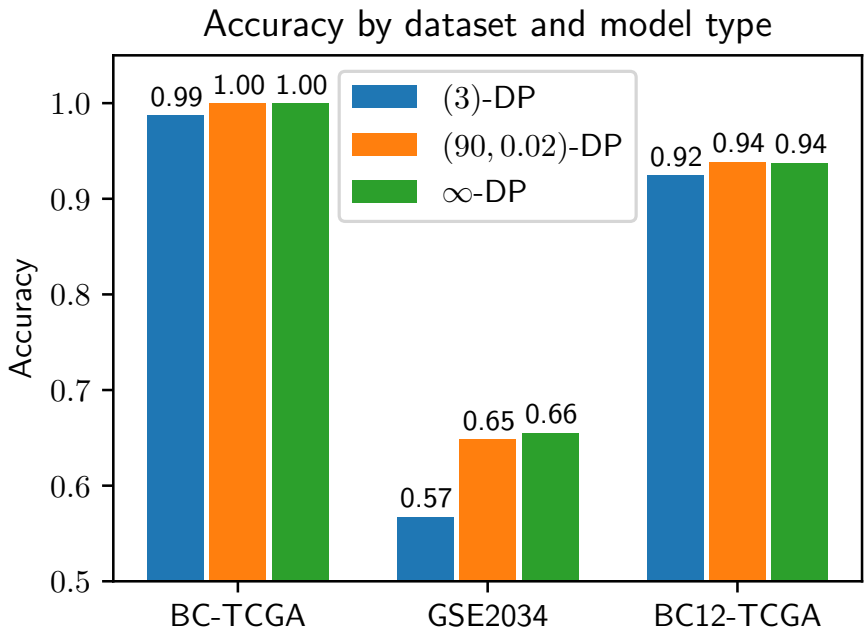

Fig. 8. Best obtained model accuracy by dataset and model type.

We compare the learning phase execution performance with results presented in [27, table 2,3]. Execution times and model accuracy scores are given in table 5. We choose the hyperparameter value h for which the best noiseless model

accuracy is obtained ($h = 5$ and $h = 50$ for BC-TCGA and GSE2034 respectively). Model accuracy scores obtained in our work are better, although the difference is not that big. The results from [27] include the communication time over a LAN network (1Gbps), which is not included in the execution times presented earlier. For fairness, in table 5 we add an estimate of communication time based on network transfer sizes: network communication field from tables 2, 3 divided by LAN network speed. To summarize, our execution time include local processing, MPC and network time estimation. Compared to [27] we use less powerfull machines: a 4-core 3.1GHz CPU vs a 18-core 3.5GHz CPU (according to AWS `c5.9xlarge` specifications). Another difference is that our implementation is single-threaded. The execution times for BC-TCGA are comparable in both cases and our protocol is ~ 11 times faster for the GSE2034 dataset. The learning task on GSE2034 dataset is harder because the authors of [27] use 223 iterations compared to only 10 for the BC-TCGA dataset. An explanation of this discrepancy would be that using approximate ReLu instead of sigmoid function does not work well on the GSE2034 dataset. In our learning phase we use the same logistic regression parameters for both datasets. We suppose that we can further increase the execution performance of our implementation by using multi-threading and by tuning the learning phase parameters (e.g. iteration count). Another reason why our protocol is faster is because of local processing which decreases the MPC phase data sizes.

	Execution time		Accuracy	
	BC-TCGA	GSE2034	BC-TCGA	GSE2034
Our work	2.50	2.38	1.000	0.655
[27]	2.52	26.90	0.996	0.648
SecureML	12.73	49.95	-	-

Table 5. Execution times (seconds) and model accuracy comparison.

5.2 Prediction phase

The TFHE scheme instantiated with parameters described in section 4.2 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).

	BC-TCGA	GSE2034	BC12-TCGA
d	18	13	25
Request size	144	104	200

Table 6. Number of TRLWE ciphertexts d and encrypted genomic data size in kB (request size) for one user.

In order to encrypt user genomic data vector (n values) we need $d = \lceil n/1024 \rceil$ TRLWE ciphertexts. The request size the analyst receives from an user has $d \times$ TRLWE ciphertexts. Table 6 illustrates the request sizes for the 3 datasets. Ciphertext expansion factor is approximately 8 for a request, which is low in the context of homomorphic encryption. 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. 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 sized 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 $h = \{20, 50\}$ and other differential privacy settings) and also no significant changes were observed in the AUC score.

6 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 genomic datasets.

Three datasets from TCGA are used in our experiments to train a logistic regression model. The obtained models achieve state-of-the-art accuracy and training times are lower when compared to existing works. Even if prediction over homomorphically encrypted data is used, the prediction phase is almost instantaneous and can be used in low-latency applications.

The GenoPPML framework is not limited to logistic-regression models only. More complex machine learning algorithms can be used at a higher computa-

tional cost for the training and the prediction phases. More precisely in figure 3, the `logreg` part can be replaced by another classification algorithm, such as XGBoost described in [28]. Although after a PCA, linear models tend to provide the best trade-off between accuracy and privacy budget.

One weak point of the GenoPPML is that model accuracy asymptotically worsens with the increase of the projection size h; 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.

Acknowledgement

The results published here are in whole or part based upon data generated by the TCGA Research Network: `https://www.cancer.gov/tcga`.

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. Boyle, E., Gilboa, N., Ishai, Y., Nof, A.: Sublinear gmw-style compiler for mpc with preprocessing. In: Annual International Cryptology Conference. pp. 457–485. Springer (2021)
16. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. ACM Transactions on Computation Theory (TOCT) **6**(3), 1–36 (2014)
17. 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)
18. 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)
19. 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)
20. 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)
21. Carpov, S., Tortech, T.: Secure top most significant genome variants search: idash 2017 competition. BMC medical genomics **11**(4), 47–55 (2018)
22. 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)
23. 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)
24. 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)
25. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Tfhe: fast fully homomorphic encryption over the torus. Journal of Cryptology **33**(1), 34–91 (2020)
26. Cho, H., Wu, D.J., Berger, B.: Secure genome-wide association analysis using multiparty computation. Nature biotechnology **36**(6), 547–551 (2018)

27. De Cock, M., Dowsley, R., Nascimento, A.C., Railsback, D., Shen, J., Todoki, A.: High performance logistic regression for privacy-preserving genome analysis. BMC Medical Genomics **14**(1), 1–18 (2021)
28. 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`
29. 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)
30. 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)
31. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. IACR Cryptol. ePrint Arch. **2012**, 144 (2012)
32. 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)
33. 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)
34. Geiping, J., Bauermeister, H., Dröge, H., Moeller, M.: Inverting gradients - how easy is it to break privacy in federated learning? In: Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual (2020)
35. 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)
36. 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)
37. 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)
38. 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)
39. 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)
40. 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)
41. 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)
42. Konečnỳ, J., McMahan, H.B., Ramage, D., Richtárik, P.: Federated optimization: Distributed machine learning for on-device intelligence. arXiv preprint arXiv:1610.02527 (2016)

43. 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)
44. 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)
45. Mittos, A., Malin, B., De Cristofaro, E.: Systematizing genome privacy research: A privacy-enhancing technologies perspective. Proceedings on Privacy Enhancing Technologies (PoPETs) **2019**(1), 87–107 (2019)
46. 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)
47. Network, C.G.A., et al.: Comprehensive molecular portraits of human breast tumours. Nature **490**(7418), 61 (2012)
48. 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)
49. Patra, A., Suresh, A.: BLAZE: blazing fast privacy-preserving machine learning. In: 27th Annual Network and Distributed System Security Symposium, NDSS (2020)
50. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
51. 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)
52. 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)
53. 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)
54. 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)
55. 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)
56. 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)
57. 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)
58. 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)
59. 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)

60. 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)
61. 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)
62. Xie, H., Li, J., Jatkoe, T., Hatzis, C.: Gene expression profiles of breast cancer (2017), http://dx.doi.org/10.17632/v3cc2p38hb.1
63. Xie, H., Li, J., Zhang, Q., Wang, Y.: Comparison among dimensionality reduction techniques based on random projection for cancer classification. Computational Biology and Chemistry **65**, 165–172 (Dec 2016). https://doi.org/10.1016/j.compbiolchem.2016.09.010, http://dx.doi.org/10.1016/j.compbiolchem.2016.09.010
64. Zhu, L., Han, S.: Deep leakage from gradients. In: Federated Learning, pp. 17–31. Springer (2020)