Efficient and Private Scoring of Decision Trees, Support Vector Machines and Logistic Regression Models based on Pre-Computation

Martine De Cock, Rafael Dowsley, Caleb Horst, Raj Katti,
Anderson C. A. Nascimento, Stacey C. Newman and Wing-Sea Poon

***Abstract*—Many data-driven personalized services require that private data of users is scored against a trained machine learning model. In this paper we propose a novel protocol for privacy-preserving classification of decision trees, a popular machine learning model in these scenarios. Our solutions are composed out of building blocks, namely a secure comparison protocol, a protocol for obliviously selecting inputs, and a protocol for evaluating polynomials. By combining some of the building blocks for our decision tree classification protocol, we also improve previously proposed solutions for classification of support vector machines and logistic regression models. Our protocols are information theoretically secure and, unlike previously proposed solutions, do not require modular exponentiations. We show that our protocols for privacy-preserving classification lead to more efficient results from the point of view of computational and communication complexities. We present accuracy and runtime results for 7 classification benchmark datasets from the UCI repository.**

***Index Terms*—Private classification, decision trees, support vector machines, logistic regression.**

I. Introduction

Data-driven machine learning has the ability to vastly improve the quality of our daily lives and is already doing so in many ways. Healthcare providers use systems based on machine learning to diagnose patients; wearable devices are connected to fitness tracking apps that use machine learning to make personal health recommendations; search engines and social media sites rely on machine learning to decide which content to show to each individual user, including which advertisements; e-commerce companies leverage machine learning to determine which products or movies to recommend to customers based on their prior purchase behavior; online dating services use machine learning in an attempt to connect people with the love of their lives... the list goes on and on. To benefit from any of these personalized services, the personal data of users – such as personal preferences, browsing behavior or medical lab results – needs to be scored against a trained machine learning model. In this paper we propose techniques to perform this scoring in an encrypted way so that individuals do not have to share their personal data with anyone "in the clear" but may still benefit from these types of personalized services.

More specifically, we deal with scenarios where a person holding data (Alice) wants to score her data against a model in possession of another party (Bob) such that, at the end of the protocol, Bob learns nothing about Alice's data and Alice learns as little as possible about Bob's model.

Our contributions: We propose a new privacy-preserving protocol for evaluating decision trees. We also substantially improve upon previously proposed protocols for hyperplane based classifiers - we include support vector machines and logistic regression classifiers as specific cases. We provide formal definitions of security and show that our protocols match these definitions. We show that our protocols compare favorably against previous results [10], [9], [18].

Our results are proven in the so-called commodity-based model [4], [3], in which correlated data is distributed to Alice and Bob during a setup phase. Later on, during an online phase, Alice and Bob use these commodities to run the desired computation on their respective inputs. This data can be pre-distributed by a trusted authority or it can be pre-computed by the players during a setup phase using well known protocols available in the literature. These commodities do not depend on the actual inputs of Alice or Bob. Thus, in case a trusted authority is used to distribute the commodities, the trusted authority never engages in the actual computation during the online phase and never learns any information about the model held by Bob or the data possessed by Alice. The protocols in our online phase are information theoretically secure; that is, if the commodities are provided in an information theoretically secure fashion, the overall protocol will be information theoretically secure. Finally, differently from previously proposed solutions [10], [9], [41] our protocols solely use modular additions and multiplications. No modular exponentiations are ever required.

The main idea behind our solutions is to decompose the problem of obtaining privacy-preserving classifiers into the problem of obtaining secure versions of a few building blocks: distributed multiplication, distributed comparison, bit-decomposition of shares, distributed inner product and argmax computation, and oblivious input selection. We then either use the most efficient available versions of these protocols or propose more efficient ones. In more detail, the main contributions include:

Martine De Cock, Caleb Horst, Raj Katti, Anderson C. A. Nascimento, Stacey C. Newman and Wing-Sea Poon are with the Institute of Technology, University of Washington Tacoma, Tacoma, WA, USA. E-mails: mdecock@uw.edu, calebjh@uw.edu, rajkatti@uw.edu, andclay@uw.edu, newmsc8@uw.edu, wpoon93@uw.edu.

Rafael Dowsley is with the Institute of Theoretical Informatics, Karlsruhe Institute of Technology. Am Fasanengarten 5, Geb. 50.34, 76131 Karlsruhe, Germany. Email: rafael.dowsley@kit.edu.

- A novel protocol for computing private scoring of decision trees where Bob learns nothing about Alice's data and Alice learns only the depth of Bob's decision tree. Moreover, only modular additions and multiplications are required. In previous solutions [10], [9], [41], either modular exponentiations and fully homomorphic encryption are required [10], [9] or Paillier encryption-based private comparison schemes and Oblivious Transfer protocols (both requiring modular exponentiations) are required [41].
- A novel secure comparison protocol that, for input sizes normally used in practical applications, outperforms existing solutions in terms of communication and computation complexities and has round complexity comparable to the best previous solution.
- Demonstration that applying an adaptation of the bit decomposition protocol proposed in [28] and our new comparison protocols as building blocks to previously proposed protocols for hyper-plane based classifiers [18] delivers more efficient results for the computational and communication complexities. We implement the particular case of support vector machines and logistic regression.
- Application of our proposed protocols on 7 real data benchmark datasets from the UCI Machine Learning repository and presentation of the obtained accuracies and running times.

Our solutions are secure in the honest-but-curious model, consistent with the security model used in previous works [10], [9], [18]. We provide full proofs of security.

Outline: We first introduce our notation and model in Section II. Section III explains the machine learning classifiers that are considered in this work. We then present the building blocks that are used in the privacy-preserving classifiers: a secure distributed comparison protocol in Section IV, a secure argmax protocol in Section V, a secure bit-decomposition protocol in Section VI and an oblivious input selection protocol in Section VII. After that, Section VIII describes the privacy-preserving classifiers and Section IX the experiments that we performed to assess their performance. Section X explains how the pre-distributed data can be generated by the parties if no trusted initializer is available (or desirable). Finally, Section XI compares our solution with the related work and Section XII presents our concluding remarks.

II. Preliminaries

A. Notation and Security Model

We denote by $y \xleftarrow{\$} F(x)$ the act of running the probabilistic algorithm F with input x and obtaining the output y. $y \leftarrow F(x)$ is similarly used for deterministic algorithms. All logarithms are base 2. For a bit b, we let $\bar{b}$ represent its negation.

In this work additively secret sharings are used to perform computation modulo q. A value x is secretly shared over $\mathbb{Z}_q$ by picking $x_1, \ldots, x_n$ uniformly at random subject to the constraint that $x = \sum_{i=1}^{n} x_i \mod q$ and then distributing each share x_i to P_i. Let $[\![x]\!]_q$ denote this secret sharing. Given $[\![x]\!]_q$, $[\![y]\!]_q$ and a constant c, it is trivial for the parties to compute a secret sharing $[\![z]\!]_q$ corresponding to $z = x + y$, $z = x - y$, $z = cx$ or $z = x + c$. All of these operations can be performed locally by the parties without any interaction by simply adding, subtracting or multiplying the shares respectively for the first three cases, and by having a pre-agreed party add the constant in the last case. These operations will be denoted respectively by $[\![z]\!]_q \leftarrow [\![x]\!]_q + [\![y]\!]_q$, $[\![z]\!]_q \leftarrow [\![x]\!]_q - [\![y]\!]_q$, $[\![z]\!]_q \leftarrow c[\![x]\!]_q$ and $[\![z]\!]_q \leftarrow [\![x]\!]_q + c$. For a secret sharing $[\![x]\!]_q$, the parties can open the value x by revealing their shares x_i. Similarly, for a matrix X, $[\![X]\!]_q$ will denote the element-wise secret sharing of the matrix and the operations will be denoted in the same way. In order to unify the treatment of the protocols with the case in which one input x is held by a single party P_i, we write $[\![x]\!]_q \leftarrow x$ to denote the case in which P_i computes with the share x and the remaining parties with shares equal to zero.

We should remark that the applications considered in this paper are between two parties, but for the sake of generality some protocols are described in a more general form, running with n parties.

Adversarial Model: The adversaries considered in this paper are honest-but-curious (as in all other privacy-preserving classification protocols so far), meaning that they follow the protocol instructions correctly, but try to learn additional information. For more information about parallel composition theorems for this model, we refer the reader to chapter 4 of [13]. Our security guarantees are based on the simulation paradigm: for each adversary attacking the real protocol, there should be a simulator in the ideal world that interacts with the ideal functionality (instead of a real protocol execution) and is such that an external party cannot distinguish the real and ideal worlds. In our protocols the simulation strategy will be described very briefly as they are very simple: all the messages look uniformly random from the recipient's point of view, except for the messages that open some secret share to a party, but these ones can be easily simulated using the output of the respective functionalities.

Functionality $\mathcal{F}_{\mathsf{TI}}^{\mathcal{D}}$

$\mathcal{F}_{\mathsf{TI}}^{\mathcal{D}}$ runs with the parties $P_1, \ldots, P_n$ and is parametrized by an algorithm $\mathcal{D}$. Upon initialization run $(D_1, \ldots, D_n) \xleftarrow{\$} \mathcal{D}$. For $i = 1, \ldots, n$, deliver D_i to P_i.

Fig. 1. The Trusted Initializer functionality.

B. Commodity-based Cryptography

The commodity-based model [4], [3] is a setup assumption in which there is a trusted initializer who pre-distributes correlated data to the protocol participants during a setup phase, which is performed before the protocol execution (possibly far before the inputs are even fixed) and is independent of the protocol inputs. The trusted initializer does not take part in the protocol execution after the setup phase; in particular, he does not learn the parties' inputs. The trusted initializer is

Functionality $\mathcal{F}_{\mathsf{DMM}}$

$\mathcal{F}_{\mathsf{DM}}$ runs with parties $P_1, \ldots, P_n$ and is parametrized by the size q of the ring and the dimensions i, j and k of the matrices.

Input: Upon receiving a message from a party with its shares of $[\![X]\!]_q$ and $[\![Y]\!]_q$, verify if the share of X is in $\mathbb{Z}_q^{i\times j}$ and the share of Y is in $\mathbb{Z}_q^{j\times k}$. If it is not, abort. Otherwise, record the shares, ignore any subsequent message from that party and inform the other parties about the receipt.

Output: Upon receipt of the shares from all parties, reconstruct X and Y from the shares, compute $Z = XY$ and create a secret sharing $[\![Z]\!]_q$ to distribute to the parties: the corrupt parties fix their shares of the output to any constant values and the shares of the uncorrupted parties are then created by picking uniformly random values subject to the correctness constraint.

Fig. 2. The distributed matrix multiplication functionality.

Secure Distributed Matrix Multiplication Protocol π_{DMM}

The protocol is parametrized by the size q of the ring and the dimensions i, j and k of the matrices, and runs with the parties $P_1, \ldots, P_n$. The trusted initializer chooses uniformly random U and V in $\mathbb{Z}_q^{i\times j}$ and $\mathbb{Z}_q^{j\times k}$, respectively, computes $W = UV$ and pre-distributes secret sharings $[\![U]\!]_q, [\![V]\!]_q, [\![W]\!]_q$ to the parties. The parties have inputs $[\![X]\!]_q$, $[\![Y]\!]_q$ and interact as follows:

1) Locally compute $[\![D]\!]_q \leftarrow [\![X]\!]_q - [\![U]\!]_q$ and $[\![E]\!]_q \leftarrow [\![Y]\!]_q - [\![V]\!]_q$, then open D and E.
2) Locally compute $[\![Z]\!]_q \leftarrow [\![W]\!]_q + E[\![U]\!]_q + D[\![V]\!]_q + DE$.

Fig. 3. The protocol for secure distributed matrix multiplication.

modeled in this work by an ideal functionality $\mathcal{F}_{\mathsf{TI}}^{\mathsf{D}}$, which is parametrized by an algorithm $\mathcal{D}$ that samples the correlated data to be pre-distributed to the parties. See Figure 1 for details.

The main advantage of using this model is that, for many problems, it allows very efficient solutions with unconditional security (in some cases even perfect security). This follows from the fact that, in these problems, the trusted initializer can pre-distribute instances computed on random inputs, which the parties later on only derandomize to match their actual inputs. This model was already used to obtain very efficient solutions for primitives such as commitments [36], [7], [31], oblivious transfer [4], [3], inner-product [20], [25], linear algebra [17], string equality [25], verifiable secret sharing [32], [19], set intersection [25] and oblivious polynomial evaluation [39]. In the context of privacy-preserving machine learning, this model was used in [18], [12].

In practice, this correlated data can be obtained in different ways: (1) it can be distributed by a single trusted center that runs the setup phase and delivers the data to the participants; (2) it can be pre-distributed by many not entirely trusted centers that do not interact with (or even know) each other. In this case only a majority of honest centers is needed [4], [6]; (3) it can be pre-computed by the parties themselves, using a multi-party computation protocol in order to emulate the trusted initializer (in this case the main advantage is offloading the heavy computational steps to an offline phase that can be executed at any idle time).

C. Secure Distributed Matrix Multiplication

Given the operations that can be performed locally with the secret sharings, one remaining important operation that is still missing is the multiplication of secret sharings. While this operation can be complicated to perform in the plain model, in the commodity-based model there is a very simple and efficient solution from Beaver [5]. Here we present an extension of his idea for basic multiplication to performs distributed matrix multiplication. The parties have as input $[\![X]\!]_q$ and $[\![Y]\!]_q$, for matrices $X \in \mathbb{Z}_q^{i\times j}$ and $Y \in \mathbb{Z}_q^{j\times k}$, and want to obtain shares of the product. The trusted initializer pre-distributes a random matrix multiplication triple to the parties, i.e., secret sharings $[\![U]\!]_q, [\![V]\!]_q$ and $[\![W]\!]_q$ for U and V uniformly random in $\mathbb{Z}_q^{i\times j}$ and $\mathbb{Z}_q^{j\times k}$, respectively, and $W = UV$. The parties then derandomize the random matrix multiplication triple during the protocol execution in order to compute a secret sharing $[\![Z]\!]_q$ corresponding to $Z = XY$ without leaking any information about the input values X and Y or the output value Z. Figure 2 describes the distributed matrix multiplication functionality $\mathcal{F}_{\mathsf{DMM}}$ that is considered and Figure 3 presents the protocol π_{DMM} that implements such functionality.

Theorem II.1. *The protocol π_{DMM} is correct and securely implements the distributed matrix multiplication functionality $\mathcal{F}_{\mathsf{DMM}}$ against honest-but-curious adversaries in the commodity-based model.*

Proof. **Correctness:** For verifying correctness, first notice that $Z = XY = (U + D)(V + E) = UV + UE + DV + DE = W + UE + DV + DE$ and therefore $[\![Z]\!]_q \leftarrow [\![W]\!]_q + E[\![U]\!]_q + D[\![V]\!]_q + DE$ obtains a secret sharing corresponding to $Z = XY$. The fact that the resulting shares are uniformly random with the constraint that $Z = XY$ follows trivially from the fact that the pre-distributed multiplication triple has this property.

Security: The simulation is very simple and proceeds as follows. The simulator $\mathcal{S}$ runs internally a copy of the adversary $\mathcal{A}$ and reproduces the real world protocol execution perfectly for $\mathcal{A}$. For that, it simulates the protocol execution with dummy inputs for the uncorrupted parties. The leverage of the simulator is the fact that it can simulate the trusted initializer functionality $\mathcal{F}_{\mathsf{TI}}^{\mathsf{D}}$ for $\mathcal{A}$. Using this leverage,

whenever a corrupted party announces its shares of D and E in the simulated protocol execution, $\mathcal{S}$ can extract the respective shares of X and Y to give to the distributed matrix multiplication functionality $\mathcal{F}_{\mathsf{DMM}}$. And whenever an honest party sends its shares to the functionality, $\mathcal{S}$ simulates the announced messages for $\mathcal{A}$ by sending random messages, which from $\mathcal{A}$'s point of view are indistinguishable from the messages in the real protocol execution as the shares of U and V are uniformly random and unknown to $\mathcal{A}$. Given its knowledge about $[\![U]\!]_q, [\![V]\!]_q, [\![W]\!]_q, D$ and E by the end of the simulated execution, $\mathcal{S}$ knows, for each corrupted party, which value its share of the output is supposed to take, and therefore $\mathcal{S}$ can fix these values in $\mathcal{F}_{\mathsf{DMM}}$ so that the sum of the uncorrupted parties' shares is compatible with the simulated execution. □

Notation: We denote by π_{DM} the protocol for the special case of multiplication of single elements. The special case of inner-product computation will be denoted as π_{IP}.

III. Machine Learning Classifiers

In this section we briefly review the machine learning models for which we propose privacy-preserving scoring protocols in Section VIII. Our presentation and notation is similar to that of Bost et al. [10], [9].

A. Decision Trees

Decision trees are non-parametric, discriminative classifiers[1]. Alice holds an input vector $\mathbf{x} = (x_1, \ldots, x_t) \in \mathbb{R}^t$ consisting of t features. The classification algorithm consists of a mapping $C\colon \mathbb{R}^t \to \{c_1, \ldots, c_k\}$ on $\mathbf{x}$. The result of the classification $C(\mathbf{x})$ is one of the k possible classes $c_1, \ldots, c_k$. The model is a tree structure and is held by Bob. Each internal node of the tree structure tests the value of a particular feature against a corresponding threshold and branches according to the results. Each leaf node specifies one of the k classes. The result of the classification is the class associated with the leaf reached from traversing the tree.

In all our secure protocols a full tree is assumed. In the case where a decision tree is not full, one can always fill it with dummy nodes to obtain a full tree. It is assumed, without loss of generality, that the trees are binary.

Bob's model is $D = (d, G, H, \mathbf{w})$, where d is the depth of the tree, $G\colon \{1, \ldots, 2^d\} \to \{1, \ldots, k\}$ is a mapping from the indices of the leaves to the indices of the classes, $H\colon \{1, \ldots, 2^d - 1\} \to \{1, \ldots, t\}$ is a mapping from the indices of the internal nodes (always considered in level-order) to the indices of Alice's input features and $\mathbf{w} = (w_1, \ldots, w_{2^d-1})$ with $w_i \in \mathbb{R}$ contains the thresholds corresponding to each internal node. For each internal node v_i with $1 \leq i \leq 2^d - 1$, let z_i be the Boolean variable denoting the result of comparing $x_{H(i)}$ with w_i, which is one if $x_{H(i)} \geq w_i$ and zero otherwise. The classification process goes as follows:

[1]Being non-parametric means that the structure of the model is not completely fixed, the model can grow in size to accommodate the complexity of the training data. Being discriminative means that the model learns boundaries between the classes.

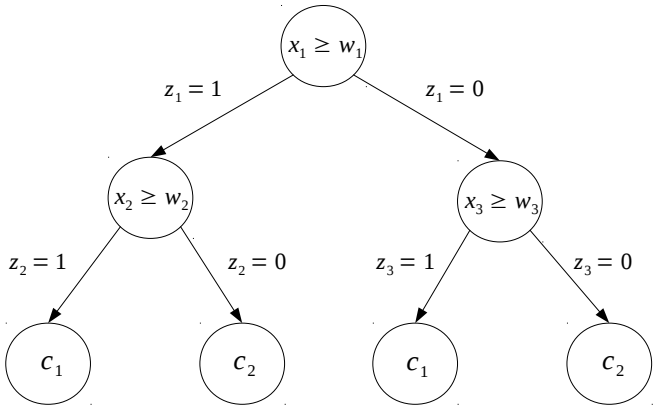

Fig. 4. Example of decision tree with 7 nodes and 2 classes.

- Starting from the root node, for the current internal node v_i, evaluate z_i. If $z_i = 1$, take the left branch; otherwise, the right branch.
- The algorithm terminates when a leaf is reached. If the j-th leaf is reached, then the output is $c_{G(j)}$.

Similar to Bost et al. [9], we are able to express D as a polynomial which has an output corresponding to the label of the resulting leaf node. The polynomial is a sum of terms such that each term corresponds to one possible path in the tree: the term corresponding to path taken by $\mathbf{x}$ in the tree evaluates to the classification result (i.e., the class associated to that leaf), while the remaining terms evaluate to zero. This polynomial is created with the knowledge of G and takes as input all z_i, which can be calculated via comparisons between the thresholds held by Bob and the features held by Alice. The classification then consists of evaluating the polynomial $P_G\colon \{0,1\}^{2^d-1} \to \{1, \ldots, k\}$ on input $\mathbf{z} = (z_1, \ldots, z_{2^d-1})$. For example, for the tree portrayed in Figure 4, the polynomial P_G that represents the tree is: $P_G(z_1, z_2, z_3) = z_1 z_2 c_1 + z_1 \bar{z_2} c_2 + \bar{z_1} z_3 c_1 + \bar{z_1} \bar{z_3} c_2$ where $\bar{x}$ denotes $1 - x$.

B. Hyperplane Based Classifiers and Support Vector Machines

Hyperplane-based classifiers are parametric, discriminative classifiers. For a setting with t features[2] and k classes, the model consists of k vectors $\mathbf{w} = (\mathbf{w}_1, \ldots, \mathbf{w}_k)$ with $\mathbf{w}_i \in \mathbb{R}^t$ and the classification result is obtained by determining, for Alice's feature vector $\mathbf{x} \in \mathbb{R}^t$, the index

$$k^* = \underset{i \in [k]}{\operatorname{argmax}} \langle \mathbf{w}_i, \mathbf{x} \rangle,$$

where $\langle \cdot, \cdot \rangle$ is the inner-product.

Hyperplane-based classifiers are very common in machine learning. They can be obtained, for example, through maximizing the margin (as in support vector machines, which are explained below), perceptron learning, Fisher linear discriminant analysis and least squares optimization. All these techniques

[2]We can have one of the features being 1 in order to account for constants.

result in hyperplane-based classifiers for which the privacy-preserving scoring protocols we propose in Section VIII are applicable.

Support vector machine (SVM) learning is a method for training classifiers based on different types of kernel functions – polynomial functions, radial basis functions, etc. An SVM is characterized by a linear separating hyperplane which maximizes the margins between the classes [21]. The decision boundary is maximized with respect to the data points from each class (known as support vectors) that are closest to the decision boundary. Support vector machines are a particular case of hyperplane-based classifiers. For the particular case of an SVM classifier with two classes c^+ and c^-, we can rephrase hyperplane-based classifiers as follows. Alice holds an input vector $\mathbf{x}$, Bob holds a model $(\mathbf{a}, b)$, where $\mathbf{a}$ is an t-dimensional vector (the weight vector) and b is a real number. The result of the classification is obtained by computing

$$\mathsf{sign}\left(\langle \mathbf{x}, \mathbf{a}\rangle + b\right),$$

where $\mathsf{sign}\,(y)$ is $+$ if $y > 0$ and $-$ otherwise.

Logistic regression is a classifier that models the posterior probability of the class given the input features by fitting a logistic curve to the relationship between them [33]. As such, logistic regression model outputs can be interpreted as probabilities of the occurrence of a class. When the response is a binary variable with class labels c^+ and c^-, then for a new input instance $\mathbf{x}$, a trained logistic regression model outputs the probabilities

$$P_{C|X}(c^-|\mathbf{x}) = \frac{1}{1+\exp(\langle \mathbf{x}, \mathbf{a}\rangle + b)}$$

and $P_{C|X}(c^+|\mathbf{x}) = 1 - P_{C|X}(c^-|\mathbf{x})$, where the weight vector $\mathbf{a}$ and the real number b are learned during the logistic regression model training process. The class decision for the given probability is then made based on a threshold value which is often set to 0.5: if $P_{C|X}(c^+|\mathbf{x}) \geq 0.5$, then we predict that the instance belongs to the positive class, and otherwise we predict the instance belongs to the negative class. In this case the classification can be done by computing

$$\mathsf{sign}\left(\langle \mathbf{x}, \mathbf{a}\rangle + b\right).$$

IV. Secure Distributed Comparison

In this section, we present a secure distributed comparison protocol which uses the multiplication protocol from Section II-C as a building block. Let the ℓ-bit integers to be compared be $x = x_\ell \ldots x_1$ and $y = y_\ell \ldots y_1$. The parties $P_1, \ldots, P_n$ have additively secret sharings $[\![x_i]\!]_2$ and $[\![y_i]\!]_2$ of each bit of x and y. The output of the distributed comparison is $[\![1]\!]_2$ if $x \geq y$ and $[\![0]\!]_2$ if $x < y$. The distributed comparison functionality $\mathcal{F}_{\mathsf{DC}}$ is described in Figure 5. It takes bitwise shares of the numbers to be compared as input and outputs shares of the comparison result. The protocol that implements it follows the lines of Damgård, Geisler and Krøigaard [15], which is one of the most efficient known solutions for the secure comparison problem. However, due to our usage of pre-distributed, correlated randomness, it is possible to eliminate the use of the computationally intensive steps. We also simplified the protocol to work over $\mathbb{Z}_2$ instead of a much larger ring $\mathbb{Z}_q$. The resulting comparison protocol π_{DC} is described in Figure 6.

Functionality $\mathcal{F}_{\mathsf{DC}}$

$\mathcal{F}_{\mathsf{DC}}$ runs with parties $P_1, \ldots, P_n$ and is parametrized by the bit-length ℓ of the values being compared.

Input: Upon receiving a message from a party with its shares of $[\![x_i]\!]_2$ and $[\![y_i]\!]_2$ for all $i \in \{1, \ldots, \ell\}$, record the shares, ignore any subsequent messages from that party and inform the other parties about the receipt.

Output: Upon receipt of the inputs from all parties, reconstruct x and y from the bitwise shares. If $x \geq y$, then create and distribute to the parties the secret sharing $[\![1]\!]_2$; otherwise the secret sharing $[\![0]\!]_2$. Before the deliver of the output shares, the corrupt parties fix their shares of the output to any constant values. In both cases the shares of the uncorrupted parties are then created by picking uniformly random values subject to the correctness constraint.

Fig. 5. The distributed comparison functionality.

Secure Distributed Comparison Protocol π_{DC}

Let ℓ be the bit length of the integers to be compared. The trusted initializer pre-distributes the correlated randomness necessary for the execution of all instances of the distributed multiplication protocol. The parties have as inputs shares $[\![x_i]\!]_2$ and $[\![y_i]\!]_2$ of each bit of x and y. The protocol proceeds as follows:

1) For $i = 1, \ldots, \ell$, compute in parallel $[\![d_i]\!]_2 \leftarrow [\![y_i]\!]_2 \left(1 - [\![x_i]\!]_2\right)$ using the multiplication protocol π_{DM} and locally compute $[\![e_i]\!]_2 \leftarrow [\![x_i]\!]_2 + [\![y_i]\!]_2 + 1$.
2) For $i = 1, \ldots, \ell$, compute $[\![c_i]\!]_2 \leftarrow [\![d_i]\!]_2 \prod_{j=i+1}^{\ell} [\![e_j]\!]_2$ using the multiplication protocol π_{DM}.
3) Compute $[\![w]\!]_2 \leftarrow 1 + \sum_{i=1}^{\ell} [\![c_i]\!]_2$ locally.

Fig. 6. The protocol for secure distributed comparison.

Theorem IV.1. *The distributed comparison protocol π_{DC} is correct and securely implements the distributed comparison functionality $\mathcal{F}_{\mathsf{DC}}$ against honest-but-curious adversaries in the commodity-based model.*

Proof. **Correctness:** We have that $x < y$ if and only if there exists i such that all the bits $(x_\ell, \ldots, x_{i+1})$ are identical to the bits $(y_\ell, \ldots, y_{i+1})$ and $x_i < y_i$. In our protocol the binary values $e_i = x_i + y_i + 1 \mod 2$ indicate whether $x_i = y_i$

Functionality $\mathcal{F}_{\mathsf{argmax}}$

$\mathcal{F}_{\mathsf{argmax}}$ runs with parties $P_1, \ldots, P_n$ and is parametrized by the bit-length ℓ of the values being compared and the number k of values being compared.

Input: Upon receiving a message from a party with its bitwise shares of $[\![v_{j,i}]\!]_2$ for all $j \in \{1, \ldots, k\}$ and $i \in \{1, \ldots, \ell\}$, record the shares, ignore any subsequent messages from that party and inform the other parties about the receipt.

Output: Upon receipt of the inputs from all parties, reconstruct the values v_j from the bitwise shares $v_{j,i}$, compute $m = \text{argmax}_{j \in \{1,\ldots,k\}} v_j$, and send m to P_1.

Fig. 7. The argmax functionality.

Secure Argmax Protocol π_{argmax}

Let ℓ be the bit length of the k values to be compared. The trusted initializer pre-distributes all the correlated randomness necessary for the execution of the instances of the distributed multiplication and comparison protocols. The parties have as input bitwise shares $[\![v_{j,i}]\!]_q$ for all $j \in \{1, \ldots, k\}$, $i \in \{1, \ldots, \ell\}$ and proceed as follows:

1) For all $j = 1, \ldots, k$ and $n \in \{1, \ldots, k\} \setminus j$, the parties execute in parallel the distributed comparison protocol π_{DC} with inputs $[\![v_{j,i}]\!]_2$ and $[\![v_{n,i}]\!]_2$ ($i = 1, \ldots, \ell$). Let $[\![w_{j,n}]\!]_2$ denote the output obtained.
2) For all $j = 1, \ldots, k$, the parties computed in parallel $[\![w_j]\!]_2 = \prod_{n \in \{1,\ldots,k\} \setminus j} [\![w_{j,n}]\!]_2$ using the distributed multiplication protocol π_{DM}.
3) The parties open w_j for P_1. If $w_j = 1$, P_1 append j to the value to be output in the end.

Fig. 8. The secure argmax protocol.

($e_i = 1$) or $x_i \neq y_i$ ($e_i = 0$). Additionally, the binary values $d_i = y_i(1 - x_i)$ indicate whether $x_i < y_i$ ($d_i = 1$) or $x_i \geq y_i$ ($d_i = 0$). Therefore $x < y$ if and only if there exists i such that $e_\ell = \ldots = e_{i+1} = 1$ and $d_i = 1$. If these conditions are met for some i, then $c_i = d_i \prod_{j=i+1}^{\ell} e_j$ will be 1; otherwise $c_i = 0$. Note that at most one c_i can be equal to 1 since a c_i could only possibly be 1 if it corresponds to the most significant bit in which the strings differ. Finally $w = 1 + \sum_{i=1}^{\ell} c_i \mod 2$ is equal to the complement of the c_i's exclusive-or and thus equal to zero if and only if there exists i such that $c_i = 1$. Putting all facts together the correctness of the protocol follows.

Security: The only messages exchanged are for the executions of the distributed multiplication protocol π_{DM}, therefore the security trivially follows from the fact that π_{DM} securely realizes $\mathcal{F}_{\mathsf{DMM}}$. The simulation is very simple and proceeds as follows. The simulator $\mathcal{S}$ runs internally a copy of the adversary $\mathcal{A}$ and reproduces the protocol execution perfectly for $\mathcal{A}$. For that, it simulates the protocol execution with dummy inputs for the uncorrupted parties. The leverage of the simulator is the fact that it can simulate the distributed multiplication functionality $\mathcal{F}_{\mathsf{DMM}}$ for $\mathcal{A}$. Using such leverage, $\mathcal{S}$ can easily extract the shares of the inputs and outputs that correspond to each corrupt party in order to give them to $\mathcal{F}_{\mathsf{DC}}$, which picks the shares of the uncorrupted parties uniformly at random subject to the correctness constraint. The real and ideal worlds are then indistinguishable. □

Optimization: The computation of the products of the e_i's do not need to be repeated. The idea is to create a binary tree with the e_i's in the leaves and then proceed upwards: at each internal node compute the product of its two children (and record it). The nodes at the same level can be computed in parallel. For the computation of the c_i's, there is at most one relevant node at each level of the tree, and the multiplications are done in parallel as soon as the values are available. This optimized version has $2 + \log \ell$ rounds and uses at most $2\ell + \frac{\ell \lceil \log \ell \rceil}{2} - 1$ instances of π_{DM} over $\mathbb{Z}_2$.

V. Secure Argmax

Suppose that the parties $P_1, \ldots, P_n$ have bitwise shares of a tuple of values $(v_1, \ldots, v_k)$ and want one of them, let's say P_1, to learn all the arguments $m \in \{1, \ldots, k\}$ such that $v_m \geq v_j$ for all $j \in \{1, \ldots, k\}$, but no party should learn any v_j or the relative order between the elements. I.e., the parties just want P_1 to learn

$$m = \arg \max_{j \in \{1,\ldots,k\}} v_j.$$

The argmax functionality $\mathcal{F}_{\mathsf{argmax}}$ is described in Figure 7. Using our protocol for secure distributed comparison it is possible to give simple and practical solutions for securely computing this function. An idea, which optimizes the number of communication rounds, is having the parties comparing in parallel each ordered pair of vectors and then using the result of the comparisons to determine the argmax. Note than when considering all executions of the comparison protocol involving a specific value v_j as the first argument, they will all return one if and only if the value is a maximum. The protocol π_{argmax} is described in Figure 8.

Theorem V.1. *The argmax protocol π_{argmax} is correct and securely implements the argmax functionality $\mathcal{F}_{\mathsf{argmax}}$ against honest-but-curious adversaries in the commodity-based model.*

Proof. **Correctness:** The correctness follows trivially as for a maximum value, all comparison involving it as the first argument will return one, and so the product of the comparison results will also be one and the index will be added to the output. For all values which are not a maximum, at least one comparison will return zero, and so the product will be zero and the index will not be added.

Security: The first two steps only involve invocations of the distributed comparison π_{DC} and multiplication π_{DM} protocols, while the last step only opens one bit of information per index,

Functionality $\mathcal{F}_{\mathsf{decomp}}$

$\mathcal{F}_{\mathsf{decomp}}$ runs with parties $P_1, \ldots, P_n$ and is parametrized by the bit-length ℓ of the value x being converted from additive sharings $[\![x]\!]_q$ in $\mathbb{Z}_q$ to additive bitwise sharings $[\![x_i]\!]_2$ in $\mathbb{Z}_2$ such that $x = x_\ell \cdots x_1$.

Input: Upon receiving a message from a party with its share of $[\![x]\!]_q$, record the share, ignore any subsequent messages from that party and inform the other parties about the receipt.

Output: Upon receipt of the inputs from all parties, reconstruct the value $x = x_\ell \cdots x_1$ from the shares, and for $i \in \{1, \ldots, \ell\}$ distribute new sharings $[\![x_i]\!]_2$ of the bit x_i. Before the output deliver, the corrupt parties fix their shares of the outputs to any constant values. The shares of the uncorrupted parties are then created by picking uniformly random values subject to the correctness constraints.

Fig. 9. The bit-decomposition functionality.

Secure Two-Party Bit-Decomposition Protocol π_{decomp}

Let ℓ be the bit length of the value x to be reshared. All distributed multiplications using protocol π_{DM} will be over $\mathbb{Z}_2$ and the required correlated randomness is predistributed by the trusted initializer. The parties, Alice and Bob, have as input $[\![x]\!]_q$ for $q = 2^\ell$ and proceed as follows:

1) Let a denote Alice's share of x, which corresponds to the bit string $a_\ell \ldots a_1$. Similarly, let b denote Bob's share of x, which corresponds to the bit string $b_\ell \ldots b_1$. Define the secret sharings $[\![y_i]\!]_2$ as the pair of shares (a_i, b_i) for $y_i = a_i + b_i \mod 2$, $[\![a_i]\!]_2$ as $(a_i, 0)$ and $[\![b_i]\!]_2$ as $(0, b_i)$.
2) Compute $[\![c_1]\!]_2 \leftarrow [\![a_1]\!]_2 [\![b_1]\!]_2$ using π_{DM} and locally set $[\![x_1]\!]_2 \leftarrow [\![y_1]\!]_2$.
3) For $i = 2, \ldots, \ell$:
   a) Compute $[\![d_i]\!]_2 \leftarrow [\![a_i]\!]_2 [\![b_i]\!]_2 + 1$
   b) $[\![e_i]\!]_2 \leftarrow [\![y_i]\!]_2 [\![c_{i-1}]\!]_2 + 1$
   c) $[\![c_i]\!]_2 \leftarrow [\![e_i]\!]_2 [\![d_i]\!]_2 + 1$
   d) $[\![x_i]\!]_2 \leftarrow [\![y_i]\!]_2 + [\![c_{i-1}]\!]_2$
4) Output $[\![x_i]\!]_2$ for $i \in \{1, \ldots, \ell\}$.

Fig. 10. The secure two-party bit-decomposition protocol.

indicating whether it corresponds to a maximum value or not; but this information is exactly the information contained in the output of the functionality $\mathcal{F}_{\mathsf{argmax}}$; hence the security of the protocol follows easily. Using the fact that π_{DC} securely realizes $\mathcal{F}_{\mathsf{DC}}$ and π_{DM} securely realizes $\mathcal{F}_{\mathsf{DMM}}$, the simulator $\mathcal{S}$ runs internally a protocol execution for the adversary $\mathcal{A}$ in which he simulates the ideal functionalities and uses dummy inputs for the uncorrupted parties. Using this leverage, it is trivial for $\mathcal{S}$ to extract the inputs of the corrupted parties in order to give to $\mathcal{F}_{\mathsf{argmax}}$. If P_1 is corrupted, $\mathcal{S}$ can then use the output it gets from $\mathcal{F}_{\mathsf{argmax}}$ to adjust the output of the simulated protocol by picking an uncorrupted party and changing its share of each w_j appropriately before the opening. The real and ideal worlds are then indistinguishable. □

Optimization: Similarly to the comparison protocol, the multiplications in the second step can be done using a binary tree approach, thus taking $\log\lceil k - 2 \rceil$ rounds.

VI. Secure Bit-Decomposition

In this section we deal with the problem of converting from shares $[\![x]\!]_q$ of a value x in a large field $\mathbb{Z}_q$ to shares of $[\![x_i]\!]_2$ in the field $\mathbb{Z}_2$, where $x_\ell \cdots x_1$ is the binary representation of x. The bit-decomposition functionality $\mathcal{F}_{\mathsf{decomp}}$ is described in Figure 9. The usefulness of such functionality comes from the fact that it allows to convert from a representation that allows the efficient execution of algebraic operations to a representation that allows the efficient execution of Boolean operations, such as a comparison. We present in Figure 10 a bit-decomposition protocol π_{decomp} that is specialized for the two-party case with $q = 2^\ell$. Alice and Bob know shares a and b, respectively, such that $x = a + b \mod 2^\ell$. Note that Alice also knows the bit string representation of a, i.e., $a_\ell \ldots a_1$, and Bob similarly knows $b_\ell \ldots b_1$. The main observation is that the difference between the sum of $a = a_\ell \ldots a_1$ and $b = b_\ell \ldots b_1$ modulo 2^ℓ and two bit strings that xor to the bit string $x_\ell \cdots x_1$ is exactly equal to the carry bits.[3] Therefore we use a carry computation to obtain the bitwise secret sharings $[\![x_i]\!]_2$ starting from $a_\ell \ldots a_1$ and $b_\ell \ldots b_1$.

Theorem VI.1. *Over any ring $\mathbb{Z}_{2^\ell}$, the bit-decomposition protocol π_{decomp} is correct and securely implements the bit-decomposition functionality $\mathcal{F}_{\mathsf{decomp}}$ for the special case of two players against honest-but-curious adversaries in the commodity-based model.*

Proof. **Correctness:** The protocol implements a full adder logic $c_i = (a_i \wedge b_i) \vee ((a_i \oplus b_i) \wedge c_{i-1})$, which can be similarly expressed as $c_i = \neg(\neg(a_i \wedge b_i) \wedge \neg((a_i \oplus b_i) \wedge c_{i-1}))$ to obtain the carry bit string. By adding c_{i-1} into y_i, we convert from bit strings that sum to x modulo 2^ℓ to bit strings that xor to x, thus obtaining the shares of x_i modulo 2.

Security: The only non-local operations are the invocations of the distributed multiplication protocol π_{DM}, which securely realizes $\mathcal{F}_{\mathsf{DMM}}$. Therefore the security follows essentially from the security of that protocol. $\mathcal{S}$ runs a copy of $\mathcal{A}$ and simulates an execution of the protocol using dummy inputs for the uncorrupted party. Since $\mathcal{S}$ is the one simulating the distributed multiplication functionality $\mathcal{F}_{\mathsf{DMM}}$, it can easily extract the corrupted party's share of the input in order to give it to $\mathcal{F}_{\mathsf{decomp}}$ and also derive the corrupted party's shares of the outputs in order to fix then in $\mathcal{F}_{\mathsf{decomp}}$. Consequently the real and ideal worlds are indistinguishable. □

[3]The protocol is similar to the one of Laud and Randmets [28], see the related works in Section XI for more details.

Functionality $\mathcal{F}_{\mathsf{OIS}}$

$\mathcal{F}_{\mathsf{OIS}}$ runs with Alice and Bob and is parametrized by the size n of the input vector $\mathbf{x} = (x_1, \ldots, x_n)$ and the bit-length ℓ of each input x_j.

Input: Upon receiving a message with the input vector $\mathbf{x} = (x_1, \ldots, x_n)$ from Alice, store them, ignore any subsequent message from her and inform Bob that the inputs were received.

Output: Upon receipt of the selected index $k \in [t]$ from Bob, distribute bitwise sharings $[\![x_{k,i}]\!]_2$ for $i \in \{1, \ldots, \ell\}$ and ignore any subsequent messages. Before the output deliver, the corrupt party fix its shares of the outputs to any constant values. The shares of the uncorrupted parties are then created by picking uniformly random values subject to the correctness constraints.

Fig. 11. The oblivious input selection functionality.

Optimization: The idea to optimize the number of rounds to logarithmic is to compute speculatively. In the first round the bit strings are divided in blocks of size 1 and the values of x_i and c_i are computed speculatively using both $c_{i-1} = 1$ and $c_{i-1} = 0$ for all but $i = 1$, for which we know that there is no carry in and so only one computation is needed. The second round divides the bit strings in blocks of size 2 and uses the information from the previous round to compute $x_{i+1}x_i$ and $c_{i+1}c_i$ speculatively using both $c_{i-1} = 1$ and $c_{i-1} = 0$ (except for the least significant block that only needs one computation). The third round proceeds analogously with blocks of size 4 by joining the blocks of size 2, and so on. After $\lceil \log \ell \rceil$ rounds one gets the desired bit strings $x_\ell \ldots x_1$ and $c_\ell \ldots c_1$. The first iteration uses 3ℓ instances of the multiplication protocol and needs two rounds of communication as there are pairs of sequential multiplications, all other iterations only need one round of communication and use 2ℓ multiplications each. Therefore in total the optimized protocol has $2 + \lceil \log \ell \rceil$ rounds and uses $2\ell \lceil \log \ell \rceil + 3\ell$ instances of the multiplication protocol.

VII. Oblivious Input Selection

In our applications there are also circumstances in which Alice holds a vector of inputs $\mathbf{x} = (x_1, \ldots, x_n)$ and Bob holds an index k, and they want to obtain bitwise secret sharings of x_k for further uses in the protocol, but without revealing any information about the inputs or k. The oblivious input selection functionality $\mathcal{F}_{\mathsf{OIS}}$, which captures this task, is described in Figure 11. In Figure 12 a protocol π_{OIS} realizing this functionality is presented.

Theorem VII.1. *The oblivious input selection protocol π_{OIS} is correct and securely implements the oblivious input selection functionality $\mathcal{F}_{\mathsf{OIS}}$ against honest-but-curious adversaries in the commodity-based model.*

Proof. **Correctness:** Straightforward to verify.

Security: Similarly to the previous proofs, $\mathcal{S}$ uses the fact that the only messages exchanged are for performing the distributed multiplications and the leverage of being able to simulate $\mathcal{F}_{\mathsf{DMM}}$ in order to simulate an execution of the protocol to $\mathcal{A}$ and at the same time being able to extract the inputs and the output shares of a corrupted party in order to forward to $\mathcal{F}_{\mathsf{OIS}}$. By doing so, the real and ideal worlds are indistinguishable. □

VIII. Assembling the Building Blocks

We now present our privacy-preserving classifiers using the building blocks from the previous sections.

A. Secure Decision Trees

Here, Alice inputs $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ and the classification algorithm will result in one of the k possible classes $c_1, \ldots, c_k$. Bob holds the model $D = (d, G, H, \mathbf{w})$, where d is the depth of the tree, G maps the leaves to classes, H maps internal nodes (always considered in level-order) to input features and $\mathbf{w}$ is a vector of thresholds. Each internal node of the tree structure tests the value of a particular feature against a corresponding threshold and branches according to the results. Each leaf node specifies a class. In all our secure protocols, we assume without loss of generality that we have a full binary tree. In case a decision tree is not full, one can always fill it with dummy nodes and obtain a full one. Let z_i be the Boolean variable denoting the result of comparing $x_{H(i)}$ with w_i. We recall the classification algorithm:

- Starting from the root node, for the current internal node v_i, evaluate z_i. If $z_i = 1$, take the left branch; otherwise, the right branch.
- The algorithm terminates when a leaf is reached. If the j-th leaf is reached, then the output is $c_{G(j)}$.

The classification can be expressed as a polynomial $P_G\colon \{0,1\}^{2^d-1} \to \{1, \ldots, k\}$ that depends on the mapping G from the leaves to the classes. On input $\mathbf{z} = (z_1, \ldots, z_{2^d-1})$, P_G gives the classification result. This polynomial is a sum of terms such that each term corresponds to one possible path in the tree: the term corresponding to path taken by $\mathbf{x}$ in the tree evaluates to the classification result (i.e., the class associated to that leaf), while the remaining terms evaluate to zero.

The idea of our secure protocol is that, for each internal node, Alice and Bob use the oblivious input selection protocol π_{OIS} to obtain bitwise secret sharings of the value $x_{H(i)}$ that will be compared against the threshold w_i of this node. Note that, as Alice does not learn any information from the execution of π_{OIS}, she does know which feature will be used in the comparison at each internal node. Then the comparisons are performed using the secure distributed comparison protocol π_{DC} in order to obtain $\mathbf{z}$, which is then used to evaluate the polynomial P_G using the secure multiplication protocol π_{DM} and local addition of secret sharings. The only information leaked about the tree structure to Alice is its depth d. The

Oblivious Input Selection Protocol π_{OIS}

Let ℓ be the bit length of the inputs to be shared and n the dimension of the input vector. The trusted initializer pre-distributes all the correlated randomness necessary for the execution of π_{DM} over $\mathbb{Z}_2$. Alice has as input a vector of values, $\mathbf{x} = (x_1, \ldots, x_n)$, and Bob has as input k, the index of the desired input value. They proceed as follows:

1) Define $y_k = 1$ and, for $j \in \{1, \ldots, n\} \setminus \{k\}$, $y_j = 0$. For $j \in \{1, \ldots, n\}$ and $i \in \{1, \ldots, \ell\}$, let $x_{j,i}$ denote the i-th bit of x_j. Define $[\![y_j]\!]_2$ as the pair of shares $(0, y_j)$ and $[\![x_{j,i}]\!]_2$ as $(x_{j,i}, 0)$
2) For $i = 1, \ldots, \ell$, compute $[\![z_i]\!]_2 \leftarrow \sum_{j=1}^{n} [\![y_j]\!]_2 [\![x_{j,i}]\!]_2$ using the distributed multiplication π_{DM} over $\mathbb{Z}_2$.
3) Output $[\![z_i]\!]_2$ for $i \in \{1, \ldots, \ell\}$.

Fig. 12. The oblivious input selection protocol.

Functionality $\mathcal{F}_{\mathsf{DT}}$

$\mathcal{F}_{\mathsf{DT}}$ is parametrized by the tree depth d, which is revealed to Alice.

Input: Upon receiving the feature vector $\mathbf{x}$ from Alice or the decision tree model $D = (d, G, H, \mathbf{w})$ from Bob, store it, ignore any subsequent message from that party, and inform the other party about the receipt.

Output: Upon receipt of the inputs from both parties, evaluate the decision tree D with the input $\mathbf{x}$. Let j be the reached leaf. Output $G(j)$ to Alice.

Fig. 13. The decision tree functionality.

decision tree functionality $\mathcal{F}_{\mathsf{DT}}$ is described in Figure 13 and a more detailed description of the protocol π_{DT} realizing $\mathcal{F}_{\mathsf{DT}}$ is in Figure 14.

Theorem VIII.1. *The decision tree protocol π_{DT} is correct and securely implements the decision tree functionality $\mathcal{F}_{\mathsf{DT}}$ against honest-but-curious adversaries in the commodity-based model.*

Proof. **Correctness:** For each leaf $j \in \{1, \ldots, 2^d\}$, the secret sharings $[\![y_{j-1,r}]\!]_2$ with $r = 1, \ldots, \lceil \log k \rceil$ obtained in step 3 correspond to a binary representation of the index of its associated class (offset by 1) if j is the leaf that would be reached by using the model D on input $\mathbf{x}$; otherwise they correspond to zeros as at least one of the terms $[\![z_u]\!]_2 + j_s$ in the multiplication would be zero. Thus in step 4, by summing all $[\![y_{j-1,r}]\!]_2$ for $j \in \{1, \ldots, 2^d\}$, opening the results and adding 1, Alice obtains the result of the classification k^*.

Secure Decision Tree Protocol π_{DT}

Alice has as input a feature vector $\mathbf{x}$ and Bob has a decision tree model $D = (d, G, H, \mathbf{w})$. Alice and Bob proceed as follows:

1) For $i = 1, \ldots, 2^d - 1$, Alice and Bob obtain bitwise secret sharings of $x_{H(i)}$ by executing the protocol π_{OIS} with inputs $x_1, \cdots, x_n$ from Alice and input $H(i)$ from Bob.
2) For $i = 1, \ldots, 2^d - 1$, Alice and Bob securely compare $x_{H(i)}$ and w_i using the protocol π_{DC}. For the input w_i, Bob inputs its bit representation to π_{DC} and Alice inputs zeros. Let $[\![z_i]\!]_2$ denote the result.
3) For $j = 0, \ldots, 2^d - 1$, let $j_d \ldots j_1$ be the binary representation of j with d bits and let $b_\alpha \ldots b_1$ for $\alpha = \lceil \log k \rceil$ be the binary representation of $G(j + 1) - 1$. For $r = 1, \ldots, \alpha$, initialize $[\![y_{j,r}]\!]_2$ with the shares $(0, b_r)$. Initialize $u = 1$ and $s = d$. While $s > 0$ do:
   a) For all $r = 1, \ldots, \alpha$ update $[\![y_{j,r}]\!]_2 \leftarrow [\![y_{j,r}]\!]_2 ([\![z_u]\!]_2 + j_s)$ using π_{DM} over $\mathbb{Z}_2$.
   b) Update $u \leftarrow 2u + j_s$ and $s \leftarrow s - 1$.
4) For all $r = 1, \ldots, \alpha$ compute $[\![\sigma_r]\!]_2 \leftarrow \sum_{j=0}^{2^d - 1} [\![y_{j,r}]\!]_2$ and open σ_r to Alice. Alice reconstructs σ from the bit string $\sigma_\alpha \ldots \sigma_1$ and outputs $k^* = \sigma + 1$.

Fig. 14. The protocol for secure evaluation of a decision tree.

Security: Alice learns the depth d of the tree in order to allow the execution, but this is leaked by $\mathcal{F}_{\mathsf{DT}}$ as well. In the first three steps messages are only exchanged in order to execute the sub-protocols π_{OIS}, π_{DC} and π_{DM} respectively, which securely realize the functionalities $\mathcal{F}_{\mathsf{OIS}}$, $\mathcal{F}_{\mathsf{DC}}$ and $\mathcal{F}_{\mathsf{DMM}}$ respectively. Then the last step simply reveals the bit string encoding the class that was the result of the classification to Alice. The simulation strategy is similar to the one in the previous sections. The simulator $\mathcal{S}$ internally runs a protocol execution for the adversary $\mathcal{A}$ in which $\mathcal{S}$ simulates $\mathcal{F}_{\mathsf{OIS}}$, $\mathcal{F}_{\mathsf{DC}}$ and $\mathcal{F}_{\mathsf{DMM}}$ and uses dummy inputs for the uncorrupted parties. Using this leverage $\mathcal{S}$ can easily extract the inputs of the corrupted party, $\mathbf{x}$ in case Alice is corrupted or $D = (d, G, H, \mathbf{w})$ in case Bob is corrupted, in order to forward to $\mathcal{F}_{\mathsf{DT}}$. In case Alice is corrupted, upon learning the correct output from $\mathcal{F}_{\mathsf{DT}}$, $\mathcal{S}$ can adjust appropriately Bob's shares of σ_r in the simulated protocol in order to match the right result. The real and ideal worlds are then indistinguishable. □

Optimization: All independent operations are run in parallel and the round complexity of step 3(a) can be reduced using techniques similar to the previous sections.

B. Secure Hyperplane-Based Classifiers

A privacy-preserving hyperplane-based classifier is easily achievable using our building blocks. The classification result of hyperplane-based classifiers is given by the index

$$k^* = \underset{i \in [k]}{\operatorname{argmax}} \langle \mathbf{w}_i, \mathbf{x} \rangle.$$

Thus, one just needs to represent the model and features in $\mathbb{Z}_q$, compute each inner product between $\mathbf{w}_i$ and $\mathbf{x}$ by using π_{IP}, input the results into the bit-decomposition protocol π_{decomp} and then into the argmax protocol π_{argmax}.

In the specific case of SVM, Alice holds an input vector $\mathbf{x}$, Bob holds a model $(\mathbf{a}, b)$, where $\mathbf{a}$ is an t-dimensional vector (the weight vector) and b is a real number. The result of the classification is obtained by computing

$$\mathsf{sign}\left(\langle \mathbf{x}, \mathbf{a} \rangle + b\right),$$

where $\mathsf{sign}\,(y)$ is $+$ if $y > 0$ and $-$ otherwise. The overall idea for obtaining privacy-preserving SVM classifiers is as follows: Alice inputs her personal vector $\mathbf{x}$ and Bob inputs his model vector $\mathbf{a}$ to the secure distributed inner product protocol π_{IP}. After that, the result is run through the bit-decomposition protocol π_{decomp}. The resultant bitwise shares are used in the comparison protocol π_{DC} to check whether it is greater than b or not, and then the final result is opened to Alice as her prediction.

To score a logistic regression classifier with threshold 0.5 one needs to check whether the expression

$$\log\left(\frac{P_{C|X}(c^+|\mathbf{x})}{P_{C|X}(c^-|\mathbf{x})}\right)$$

is positive or not, where

$$P_{C|X}(c^-|\mathbf{x}) = \frac{1}{1 + \exp(\langle \mathbf{x}, \mathbf{a} \rangle + b)}.$$

This boils down to computing $\mathsf{sign}\left(\langle \mathbf{x}, \mathbf{a} \rangle + b\right)$, where $\mathbf{x}$ is the input feature vector, and the $\mathbf{a}$ and b are vectors defining the logistic regression classification model (held by Bob). Therefore, the protocol used to privately evaluate a logistic regression model is exactly the same as the one described in the support vector machines section.

The security of these compositions follows from the security of the sub-protocols and the fact that no values are ever opened before the final result; each party only sees shares, which appear completely random.

IX. Experiments

For decision trees, SVM and logistic regression models we report accuracy (calculated using 10-fold cross validation) for 7 different datasets within the UCI Repository[4]. We also report average classification time for an instance in each dataset when following our privacy-preserving protocol as well as average time required when the classification is done in the clear. Note that the bit-length used to express the values should be large enough as not to compromise the accuracy of the algorithms. It is no real gain for applications if the performance is improved at the cost of drastically decreasing the accuracy, therefore the accuracy is also reported.

Support Vector Machine: For this study, we tested SVM with a linear kernel, and we report the results for accuracy for 7 different datasets from the UCI repository. We leveraged the e1071 package within R [29], setting type to 'C-classification', indicating our problems were classification tasks.

Decision Trees: We used an implementation of the classification and regression tree algorithm (CART) [11] in R [37]. The minimum deviance (mean squared error) is used as the test parameter for proceeding with a new split. That is, adding a node should reduce the error by at least a certain amount. For our models, we set the complexity parameter to 0.01 and report the corresponding accuracy.

Logistic Regression: For our experimentation, we used R's base glm function[35], setting the family parameter to binomial(link="logit") to obtain a logistic regression model.

The following datasets were chosen for our experimentation:

1) **Breast Cancer Wisconsin (Diagnostic):** The goal with this dataset is to classify 568 different tumors as malignant or benign. Each tumor is characterized by 30 different continuous features derived from an image of the tumor (i.e. perimeter, area, symmetry, etc.).
2) **Pima Indians Diabetes:** This dataset includes 767 females of at least 21 years of age, all with Pima Indian decent, and we wish to identify those with diabetes. We leverage 8 different continuous features which describe each woman's health (examples: body mass index, diastolic blood pressure).
3) **Parkinsons:** Here, the task is to differentiate between patients with and without Parkinsons. To this end, the dataset includes 22 features, all of which are measures derived from voice recordings of 195 different patients (example: average vocal fundamental frequency).
4) **Connectionist Bench (Sonar, Mines vs. Rocks):** The goal with this dataset is to differentiate whether 207 sonar signals were bounced off of a metal cylinder vs. a roughly cylindrical rock. Each of the 60 features is within the range of 0.0 to 1.0 and represents energy within a particular frequency band over a certain period of time.
5) **Hill-Valley:** The task for this dataset is to identify hills vs. valleys in terrain. Each of the 100 continuous features is a point on a 2-D graph. We chose the dataset which did not contain any noise.
6) **LSVT Voice Rehabilitation:** This dataset includes 126 patients who have undergone voice rehabilitation treatment and we wish to determine the success of their treatment, i.e. whether their phonations are considered acceptable or unacceptable. To do this, we leverage 312 features, each of which is the results of a different speech signal algorithm.
7) **Spambase:** Here, the goal is to identify 4,600 emails as either spam or not spam. This dataset includes 57 features which describe the contents of each email (examples: word frequencies, number of capital letters).

[4]UC Irvine Machine Learning Repository https://archive.ics.uci.edu/ml/datasets.html

A. Results

1) Implementation Specifics: To generate preliminary results, the privacy-preserving algorithms were implemented in Java, and compared against a simple implementation without any privacy preservation. For our experiments with the privacy-preserving classifiers, a general bit length, ℓ, of 64 bits was used for representing all the inputs and throughout all calculations, as this allowed for a good trade off between complexity and space for precision. For some trials, a smaller bit length might have served with sufficient precision.

All values had to be converted to integers to properly work in the proposed algorithms. This was accomplished by choosing a multiplier value and applying it to the features and the weights for SVM and logistic regression or the thresholds for decision trees and rounding any remaining decimals. Furthermore, since calculations were done over a ring, any negative values had to be expressed as their additive inverses. This means in addition to precision considerations, the bit length must be selected in such a way that the positive values and negative values will remain distinctly separate in the lower half and upper half of the values, respectively. This allows us to differentiate between positive and negative values by comparing against $2^{\ell-1}$ instead of 0.

Table I presents the results for the case of decision tree classifiers and Table II for SVM and logistic regression classifiers. These results were generated using our implementations as run on a nearly off the shelf personal computer with 4 GB RAM at 1333 MHz, an Intel Core i7 at 2 GHz, and a Windows 7 OS, with most nonessential background tasks stopped for the duration of the tests.

Classifications were run for each dataset at least 50 times and the average duration of the online portion of the protocol execution was recorded. For secure classification experiments with the Spambase dataset, a subset of the full UCI dataset was used to limit the amount of pre-distributed data necessary.

B. Analysis and Comparisons to Previous Results

Decision Trees: the computing time for running our protocol for the privacy-preserving evaluation of decision trees is at most 26 milliseconds for trees of depth up to 9. In [10], [9], for evaluating a tree of depth 4, the computing time is in the order of a few seconds. Our protocol has 12 rounds of communication or less for trees with depth up to 9, while the number of interactions in [10], [9] is always over 30, even for trees of depth 4. In the case of the protocols for computing decision trees of [41], the computing time for a tree with depth 4 is around 100 ms (about 50 time slower than ours). The communication complexity of our protocol for a decision tree of depth 4 and 8 features is around 2KB, while the results in [41] are around 100KB and in [10], [9] are around 3MB for trees of the same dimension. As stated in [10], [9] and in [41], solutions based on general purpose multiparty computation frameworks have a much poorer performance then the specific protocols presented here as well as the protocols presented in [10], [9] and in [41].

It is noteworthy that while our implementation is in Java, the implementations in [10], [9] and in [41] are in C++. Thus, we could probably decrease our running time significantly by implementing them in C++.

Support Vector Machines: We run the protocols proposed in [18] with our optimized bit decomposition and comparison protocols. While there are no implementation times given in [18], it is clear that our implementations have a significant impact in the performance. The number of rounds is usually the most important factor in determining the latency of these protocols and we reduce the round complexity from linear, as proposed in [18], to logarithmic in the input length. Compared to the implementations described in [10], [9] the computation times are about 50ms for 30 and 47 features. In our case for 30 features, the computing time is less than 6 ms. Our number of rounds is larger than in [10], [9]. Our solution takes 17 rounds. The solution in [10], [9] takes 7 rounds. If the roundtrip time is the major factor in the total time the solution proposed in [10], [9] is preferable to ours. The main reason for the elevated round complexity in our solution is the bit decomposition protocol, which is not needed in [10], [9].

Logistic Regression: The efficiency of the logistic regression protocol is the same as the support vector machine one.

X. Removing the Trusted Initializer

Our protocols assume that pre-distributed data is made available to the players by a trusted initializer: random binary multiplication triples (binary Beaver triples) in the case of decision trees and random binary multiplication triples and random inner product evaluations for the support vector machines and logistic regression classifiers.

In case a trusted initializer is not available or desirable, Alice and Bob can run pre-computations during a setup phase. In the case of the protocol evaluating decision trees, to obtain the binary random multiplication triples, Alice and Bob can run oblivious transfer protocols on random inputs. The outcome of these evaluations can be easily transformed in the random binary multiplication triples. The nice point of this solution is that oblivious transfer can be extended efficiently by using symmetric cryptographic primitives [24], [27], [2]. The online phase of our protocols would remain the same - using solely modular additions and multiplications. Therefore, even considering the offline phase, our protocol would still be substantially more efficient than the protocols proposed in [10], [9] and in [41]. We also remark that the protocol for evaluating decision trees in [10], [9] does not allow its computationally heavy steps (Paillier encryptions and uses of a somewhat homomorphic encryption scheme) to be pre-computed. We also note that while the oblivious transfer executions in [41] could also be pre-computed, the Paillier encryption scheme would still be needed in the online phase.

XI. Related Works

Privacy-preserving Scoring of Machine Learning Classifiers: There is a huge literature in *training* privacy-preserving machine learning models (see [1] for a survey). However, general (non-application specific) privacy-preserving protocols for privately scoring machine learning classifiers were proposed just recently in [10], [9] for the case of hyperplane-based classifiers, Naive Bayes and decision trees and in [41]

Dataset	Depth of Tree	Number of Features	Accuracy	Classification Time in the Clear (ms)	Classification Time Secure Protocol (ms)	Communication Complexity Uplink+Downlink (kB)
Breast Cancer	4	30	95.95%	0.07 + 1 RTT/2	6.5 + 11 RTT/2	8.21
Diabetes	9	8	77.18%	0.02 + 1 RTT/2	23.0 + 12 RTT/2	104.36
Parkinson's	4	22	88.72%	0.40 + 1 RTT/2	7.6 + 11 RTT/2	6.34
Connectionist Bench	4	60	73.91%	0.10 + 1 RTT/2	20.8 + 11 RTT/2	15.24
Hill-Valley	3	100	49.83%	0.14 + 1 RTT/2	9.2 + 10 RTT/2	11.49
LSVT rehabilitation	3	310	79.37%	0.75 + 1 RTT/2	25.7 + 10 RTT/2	34.46
Spambase	6	57	88.89%	0.10 + 1 RTT/2	9.2 + 12 RTT/2	61.08

TABLE I

RESULTS OF THE EXPERIMENTS FOR THE DECISION TREE CLASSIFIERS. THE CLASSIFICATION TIME IS GIVEN AS THE COMPUTING TIME PLUS THE NUMBER OF HALF ROUNDTRIP TIMES (RTT/2).

Dataset	Number of Features	Accuracy	Classification Time in the Clear (ms)	Classification Time Secure Protocol (ms)	Communication Complexity Uplink+Downlink (kB)
SVM					
Breast Cancer	30	97.71%	0.06 + 1 RTT/2	5.1 + 17 RTT/2	0.94
Diabetes	8	77.05%	0.02 + 1 RTT/2	2.8 + 17 RTT/2	0.59
Parkinson's	22	87.18%	0.04 + 1 RTT/2	3.1 + 17 RTT/2	0.81
Connectionist Bench	60	74.70%	0.10 + 1 RTT/2	6.3 + 17 RTT/2	1.40
Hill-Valley	100	57.59%	0.17 + 1 RTT/2	6 + 17 RTT/2	2.03
LSVT rehabilitation	310	80.16%	0.51 + 1 RTT/2	13.3 + 17 RTT/2	5.31
Spambase	57	92.72%	0.10 + 1 RTT/2	7.3 + 17 RTT/2	1.36
Logistic Regression					
Breast Cancer	30	95.95%	0.07 + 1 RTT/2	5.4 + 17 RTT/2	0.94
Diabetes	8	77.31%	0.02 + 1 RTT/2	4.7 + 17 RTT/2	0.59
Parkinson's	22	85.13%	0.04 + 1 RTT/2	5.1 + 17 RTT/2	0.81
Connectionist Bench	60	74.40%	0.11 + 1 RTT/2	6.2 + 17 RTT/2	1.40
Hill-Valley	100	60.07%	0.16 + 1 RTT/2	7.2 + 17 RTT/2	2.03
LSVT rehabilitation	310	53.17%	0.49 + 1 RTT/2	12.8 + 17 RTT/2	5.31
Spambase	57	92.70%	0.10 + 1 RTT/2	6.0 + 17 RTT/2	1.36

TABLE II

RESULTS OF THE EXPERIMENTS FOR THE SVM AND LOGISTIC REGRESSION CLASSIFIERS. THE CLASSIFICATION TIME IS GIVEN AS THE COMPUTING TIME PLUS THE NUMBER OF HALF ROUNDTRIP TIMES (RTT/2). ALL DATASETS ONLY HAVE TWO CLASSES.

for decision trees and random forests. In [18] protocols for hyperplane-based and Naive Bayes classifiers were proposed.

In [10], [9], hyperplane-based classifiers were implemented by using a secure protocol for computing the inner product based on the Paillier encryption scheme and a comparison protocol that also relies heavily on the Paillier encryption scheme.

The decision tree protocol of Bost et al. [10], [9] is divided in two phases. In a first stage Paillier-based comparison protocols are run with Alice inputting a vector containing her features and Bob inputting the threshold values of the decision tree. On a second stage, fully homomorphic encryption is used to process the outcomes of the comparison protocols run in the first stage. It is claimed that the protocol leaks nothing about the tree (we will show that in a more realistic attack scenario this is not true) and the second stage is round-optimal. However, the computations to be performed are heavy and the first stage involves many rounds (in total their protocol typically has more rounds than ours). In our solution, we allow the depth of the tree to be leaked, but avoid altogether using Paillier and fully homomorphic encryption. In our solution, the online phase for evaluating decision trees uses solely modular additions and multiplications.

In [41] protocols for decision trees and random forests were proposed. The protocols are based on an original comparison protocol also based on the Paillier encryption scheme and on oblivious transfer. The Paillier encryption scheme uses modular exponentiation and oblivious transfer protocols that are usually as expensive as public-key cryptographic primitives. As pointed out in the introduction, our solutions use, in the online phase, solely additions and multiplications over a finite field or ring.

In [18], one can find protocols for hyperplane-based and Naive Bayes classifiers in the commodity-based model. By directly replacing some of the building blocks used in [18] (the comparison and bit decomposition protocols) by the ones we propose in this paper, the communication and computing complexities can be decreased.

All the published results for privacy-preserving machine learning classification are secure in the honest-but-curious model.

How much information is leaked about the decision trees in [10], [9] and in [41]: In the protocol in [10], [9], theoretically nothing is ever leaked about the tree. However, if an adversary can measure the time it takes for Bob to do the evaluation of the decision tree protocol, clearly the deeper the tree the longer the computation becomes. Therefore, some information about the depth of the tree is leaked if this side channel attack is considered. Therefore, in our solution we do not loose much by giving away the depth of the tree to an adversary. In [41], the depth of the tree is also leaked.

Privacy-preserving Comparison Protocols: Secure comparison cannot be performed without some kind of assumption. In the setting of computational security, it can be obtained using general building blocks such as homomophic encryption [15], [16], [23], [26] and Yao's garbled circuits [30],

or also using specific assumptions, such as encryption of bits as quadratic and non-quadratic residues modulo an RSA modulus [22]. All these protocols involve costly computational operations.

In this work the focus is on protocols with unconditional security. For obtaining unconditionally secure comparison one standard assumption is the existence of a secure multiplication protocol [14], [34], [38], [8], which itself can be achieved using pre-distributed correlated randomness in the commodity-based model. We consider the problem of comparison separately from that of bit-decomposition. Therefore, to have a fair comparison with previous solutions, we consider their internal comparison protocols, which already receive bit-decomposed values; instead of their full-fledged comparison protocols, which also perform the bit-decomposition. For inputs with ℓ bits, it is possible to use only $O(\ell)$ invocations of the multiplication protocol and even get constant round comparison protocols [14], [34], [38]. The protocol in [18] uses exactly ℓ instances of the underlying multiplication protocol and has $\lceil \log(\ell+1) \rceil$ rounds; however its output is not represented as 0 or 1 in $\mathbb{Z}_q$ as usual, but instead as 0 or uniformly random in $\mathbb{Z}_q^*$, and this restricts its usability as a sub-protocol to create more complex protocols.

Our comparison protocol uses operations over $\mathbb{Z}_2$ instead of $\mathbb{Z}_q$ with large q. In its optimized version it uses $2\ell + \frac{\ell\lceil \log \ell \rceil}{2} - 1$ instances of a *binary* multiplication protocol (i.e., an AND gate) and has $2 + \log \ell$ rounds. We have a bigger complexity $O(\ell \log \ell)$ in terms of invocations of the multiplication protocol, but the operations are changed from $\mathbb{Z}_q$ (with a large value q) to $\mathbb{Z}_2$, which implies smaller storage complexity and faster operations. We should also mention that our constants are better. The solution of Toft [38], for example, uses $13\ell + 6\sqrt{\ell}$ instances of the underlying multiplication protocol (in $\mathbb{Z}_q$), which for the typical values of ℓ used in machine learning classifiers is more than our solution. And for these typical values of ℓ, the total number of rounds of our protocol will be very close to the 6 rounds of Toft's protocol. In comparison to the protocol of David et al. [18], we trade-off ℓ multiplications in $\mathbb{Z}_q$ with $q > 2^{\ell+2}$ for $2\ell + \frac{\ell\lceil \log \ell \rceil}{2} - 1$ multiplications in $\mathbb{Z}_2$, and also eliminate the restrictions due to its non-standard output representation.

Finally, we should mention that if the parties are the ones generating the correlated data during a pre-computation phase, as our protocol uses multiplications in $\mathbb{Z}_2$, they can benefit from oblivious transfer extension techniques [24], [27], [2] for generating the random *binary* multiplication triples necessary to perform the online phase.

Bit Decomposition Protocols: The best solution for bit-decomposition, in terms of round complexity, is a constant-round solution by Toft [38], which has round complexity equal to 23. Veugen noted in [40] that for a certain range of practical parameters (number of input bits less than 20), a protocol with a linear number of rounds in the length of the input could outperform the solution presented by Toft [38]. Veugen proposed a protocol that has a linear number of rounds in ℓ, where ℓ is the length of the input in bits. Veugen also proposed a way to reduce the number of rounds of this protocol by a factor of β, obtaining a round complexity equal to ℓ/β at the cost of performing an exponential (in β) number of multiplications in a pre-processing phase.

The bit-decomposition protocol used in this work is over binary fields and runs in $2+\lceil \log \ell \rceil$ rounds. For practical values of ℓ (less than 100 typically), it is always better than Toft's and Veugen's solutions. The number of multiplications to be performed in our the online phase, $2\ell\lceil \log \ell \rceil + 3\ell$, is less than the $31\ell\lceil \log \ell \rceil + 71\ell + 30\lceil \sqrt{\ell} \rceil$ multiplications in the case of Toft's protocol. While Veugen's protocol can have a fast online phase, requiring only $3\ell - 2\beta$ multiplications for ℓ/β rounds, it requires an exponential (in β) number of multiplications in the offline phase.

A restriction of our protocol is that it only works for operations modulo a power of 2. As we need no modular inversions in our privacy-preserving machine learning protocols this imposes no problem at all. The bit-decomposition protocol of Laud and Randmets [28] for the case of three parties with at most one corruption is similar to one here. It first reduce the original problem to a new one between two-parties, and then uses the adder idea to obtain bitwise shares. Although the protocol is not fully specified in [28], we believe that the authors intended to use the same adder computation as in our work.

XII. Conclusion

In this paper we have proposed a novel protocol for privacy-preserving classification of decision trees, and improved the performance of previously proposed protocols for general hyperplane-based classifiers and for the two specific cases of support vector machines and logistic regression. Our protocols work in the commodity-based model. The pre-distributed data can be distributed during a setup phase by a trusted authority to Alice and Bob. In the case a trusted authority is not available or desirable, Alice and Bob can pre-compute this data by themselves, during a setup phase, with the help of well-known computationally secure schemes.

Our solutions are very efficient and use solely modular addition and multiplications. We present accuracy and runtime results for 7 classification benchmark datasets from the UCI repository.

References

[1] C. C. Aggarwal and S. Y. Philip. *A general survey of privacy-preserving data mining models and algorithms*. Springer, 2008.

[2] G. Asharov, Y. Lindell, T. Schneider, and M. Zohner. More efficient oblivious transfer and extensions for faster secure computation. In A.-R. Sadeghi, V. D. Gligor, and M. Yung, editors, *ACM CCS 13: 20th Conference on Computer and Communications Security*, pages 535–548, Berlin, Germany, Nov. 4–8, 2013. ACM Press.

[3] D. Beaver. Precomputing oblivious transfer. In D. Coppersmith, editor, *Advances in Cryptology – CRYPTO'95*, volume 963 of *Lecture Notes in Computer Science*, pages 97–109, Santa Barbara, CA, USA, Aug. 27–31, 1995. Springer, Heidelberg, Germany.

[4] D. Beaver. Commodity-based cryptography (extended abstract). In *29th Annual ACM Symposium on Theory of Computing*, pages 446–455, El Paso, Texas, USA, May 4–6, 1997. ACM Press.

[5] D. Beaver. One-time tables for two-party computation. In *Computing and Combinatorics*, pages 361–370. Springer, 1998.

[6] D. Beaver. Server-assisted cryptography. In *Proceedings of the 1998 workshop on New security paradigms*, NSPW '98, pages 92–106, New York, NY, USA, 1998. ACM.

[7] C. Blundo, B. Masucci, D. R. Stinson, and R. Wei. Constructions and bounds for unconditionally secure non-interactive commitment schemes. *Des. Codes Cryptography*, 26(1-3):97–110, June 2002.

[8] P. Bogetoft, D. L. Christensen, I. Damgård, M. Geisler, T. Jakobsen, M. Krøigaard, J. D. Nielsen, J. B. Nielsen, K. Nielsen, J. Pagter, M. I. Schwartzbach, and T. Toft. Secure multiparty computation goes live. In R. Dingledine and P. Golle, editors, *FC 2009: 13th International Conference on Financial Cryptography and Data Security*, volume 5628 of *Lecture Notes in Computer Science*, pages 325–343, Accra Beach, Barbados, Feb. 23–26, 2009. Springer, Heidelberg, Germany.

[9] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser. Machine learning classification over encrypted data. Cryptology ePrint Archive, Report 2014/331, 2014. http://eprint.iacr.org/2014/331.

[10] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser. Machine learning classification over encrypted data. In *ISOC Network and Distributed System Security Symposium – NDSS 2015*, San Diego, California, USA, Feb. 8–11, 2015. The Internet Society.

[11] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth Publishing Company, 1984.

[12] M. d. Cock, R. Dowsley, A. C. Nascimento, and S. C. Newman. Fast, privacy preserving linear regression over distributed datasets based on pre-distributed data. In *Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security*, AISec '15, pages 3–14, New York, NY, USA, 2015. ACM.

[13] R. Cramer, I. Damgård, and J. B. Nielsen. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 2015.

[14] I. Damgård, M. Fitzi, E. Kiltz, J. B. Nielsen, and T. Toft. Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. In S. Halevi and T. Rabin, editors, *TCC 2006: 3rd Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 285–304, New York, NY, USA, Mar. 4–7, 2006. Springer, Heidelberg, Germany.

[15] I. Damgård, M. Geisler, and M. Krøigaard. Homomorphic encryption and secure comparison. *IJACT*, 1(1):22–31, 2008.

[16] I. Damgård, M. Geisler, and M. Krøigaard. A correction to 'efficient and secure comparison for on-line auctions'. *IJACT*, 1(4):323–324, 2009.

[17] B. David, R. Dowsley, J. van de Graaf, D. Marques, A. C. A. Nascimento, and A. C. B. Pinto. Unconditionally secure, universally composable privacy preserving linear algebra. *Information Forensics and Security, IEEE Transactions on*, 11(1):59–73, Jan 2016.

[18] B. M. David, R. Dowsley, R. Katti, and A. C. A. Nascimento. Efficient unconditionally secure comparison and privacy preserving machine learning classification protocols. In M. H. Au and A. Miyaji, editors, *ProvSec 2015: 9th International Conference on Provable Security*, volume 9451 of *Lecture Notes in Computer Science*, pages 354–367, Kanazawa, Japan, Nov. 24–26, 2015. Springer, Heidelberg, Germany.

[19] R. Dowsley, J. Müller-Quade, A. Otsuka, G. Hanaoka, H. Imai, and A. C. A. Nascimento. Universally composable and statistically secure verifiable secret sharing scheme based on pre-distributed data. *IEICE Transactions*, 94-A(2):725–734, 2011.

[20] R. Dowsley, J. van de Graaf, D. Marques, and A. C. A. Nascimento. A two-party protocol with trusted initializer for computing the inner product. In Y. Chung and M. Yung, editors, *WISA 10: 11th International Workshop on Information Security Applications*, volume 6513 of *Lecture Notes in Computer Science*, pages 337–350, Jeju Island, Korea, Aug. 24–26, 2010. Springer, Heidelberg, Germany.

[21] H. Drucker, C. J. C. Burges, L. Kaufman, A. J. Smola, and V. Vapnik. Support vector regression machines. In *Advances in Neural Information Processing Systems 9, Proceedings of the 1996 NIPS conference*, pages 155–161, 1996.

[22] M. Fischlin. A cost-effective pay-per-multiplication comparison method for millionaires. In D. Naccache, editor, *Topics in Cryptology – CT-RSA 2001*, volume 2020 of *Lecture Notes in Computer Science*, pages 457–472, San Francisco, CA, USA, Apr. 8–12, 2001. Springer, Heidelberg, Germany.

[23] J. A. Garay, B. Schoenmakers, and J. Villegas. Practical and secure solutions for integer comparison. In T. Okamoto and X. Wang, editors, *PKC 2007: 10th International Conference on Theory and Practice of Public Key Cryptography*, volume 4450 of *Lecture Notes in Computer Science*, pages 330–342, Beijing, China, Apr. 16–20, 2007. Springer, Heidelberg, Germany.

[24] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank. Extending oblivious transfers efficiently. In D. Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 145–161, Santa Barbara, CA, USA, Aug. 17–21, 2003. Springer, Heidelberg, Germany.

[25] Y. Ishai, E. Kushilevitz, S. Meldgaard, C. Orlandi, and A. Paskin-Cherniavsky. On the power of correlated randomness in secure computation. In A. Sahai, editor, *TCC 2013: 10th Theory of Cryptography Conference*, volume 7785 of *Lecture Notes in Computer Science*, pages 600–620, Tokyo, Japan, Mar. 3–6, 2013. Springer, Heidelberg, Germany.

[26] R. S. Katti and C. Ababei. Secure comparison without explicit XOR. *CoRR*, abs/1204.2854, 2012.

[27] V. Kolesnikov and R. Kumaresan. Improved OT extension for transferring short secrets. In R. Canetti and J. A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 54–70, Santa Barbara, CA, USA, Aug. 18–22, 2013. Springer, Heidelberg, Germany.

[28] P. Laud and J. Randmets. A domain-specific language for low-level secure multiparty computation protocols. In I. Ray, N. Li, and C. Kruegel:, editors, *ACM CCS 15: 22nd Conference on Computer and Communications Security*, pages 1492–1503, Denver, CO, USA, Oct. 12–16, 2015. ACM Press.

[29] D. Meyer, E. Dimitriadou, K. Hornik, A. Weingessel, and F. Leisch. *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*, 2015. R package version 1.6-7.

[30] M. Naor, B. Pinkas, and R. Sumner. Privacy preserving auctions and mechanism design. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, EC '99, pages 129–139, New York, NY, USA, 1999. ACM.

[31] A. C. A. Nascimento, J. Müller-Quade, A. Otsuka, G. Hanaoka, and H. Imai. Unconditionally secure homomorphic pre-distributed bit commitment and secure two-party computations. In C. Boyd and W. Mao, editors, *ISC 2003: 6th International Conference on Information Security*, volume 2851 of *Lecture Notes in Computer Science*, pages 151–164, Bristol, UK, Oct. 1–3, 2003. Springer, Heidelberg, Germany.

[32] A. C. A. Nascimento, J. Müller-Quade, A. Otsuka, G. Hanaoka, and H. Imai. Unconditionally non-interactive verifiable secret sharing secure against faulty majorities in the commodity based model. In M. Jakobsson, M. Yung, and J. Zhou, editors, *ACNS 04: 2nd International Conference on Applied Cryptography and Network Security*, volume 3089 of *Lecture Notes in Computer Science*, pages 355–368, Yellow Mountain, China, June 8–11, 2004. Springer, Heidelberg, Germany.

[33] A. Y. Ng and M. I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. In *Advances in Neural Information Processing Systems 14, Proceedings of the 2001 NIPS conference*, pages 841–848. MIT Press, 2001.

[34] T. Nishide and K. Ohta. Multiparty computation for interval, equality, and comparison without bit-decomposition protocol. In T. Okamoto and X. Wang, editors, *PKC 2007: 10th International Conference on Theory and Practice of Public Key Cryptography*, volume 4450 of *Lecture Notes in Computer Science*, pages 343–360, Beijing, China, Apr. 16–20, 2007. Springer, Heidelberg, Germany.

[35] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2015.

[36] R. L. Rivest. Unconditionally secure commitment and oblivious transfer schemes using private channels and a trusted initializer. Preprint available at http://people.csail.mit.edu/rivest/Rivest- commitment.pdf, 1999.

[37] T. Therneau, B. Atkinson, and B. Ripley. *rpart: Recursive Partitioning and Regression Trees*, 2015. R package version 4.1-10.

[38] T. Toft. Constant-rounds, almost-linear bit-decomposition of secret shared values. In M. Fischlin, editor, *Topics in Cryptology – CT-RSA 2009*, volume 5473 of *Lecture Notes in Computer Science*, pages 357–371, San Francisco, CA, USA, Apr. 20–24, 2009. Springer, Heidelberg, Germany.

[39] R. Tonicelli, A. C. A. Nascimento, R. Dowsley, J. Müller-Quade, H. Imai, G. Hanaoka, and A. Otsuka. Information-theoretically secure oblivious polynomial evaluation in the commodity-based model. *International Journal of Information Security*, 14(1):73–84, 2015.

[40] T. Veugen. Linear round bit-decomposition of secret-shared values. *Information Forensics and Security, IEEE Transactions on*, 10(3):498–506, March 2015.

[41] D. J. Wu, T. Feng, M. Naehrig, and K. E. Lauter. Privately evaluating decision trees and random forests. *IACR Cryptology ePrint Archive*, 2015:386, 2015.