Privacy-Preserving Scoring of Tree Ensembles: A Novel Framework for AI in Healthcare

Kyle Fritchman*, Keerthanaa Saminathan*, Rafael Dowsley†, Tyler Hughes‡,
Martine De Cock*,§, Anderson Nascimento* and Ankur Teredesai*,‡
*Institute of Technology, University of Washington, Tacoma, Washington, USA
Email: kfritchman46@gmail.com, keergs@uw.edu, mdecock@uw.edu, andclay@uw.edu, ankurt@uw.edu
†Dept. of Computer Science, Aarhus University, Aarhus, Denmark
Email: rafael@cs.au.dk
‡KenSci, Seattle, Washington, USA
Email: tyler@kensci.com, ankur@kensci.com
§Dept. of Applied Math., Comp. Science and Statistics, Ghent University, Ghent, Belgium
Email: martine.decock@ugent.be

***Abstract*—Machine Learning (ML) techniques now impact a wide variety of domains. Highly regulated industries such as healthcare and finance have stringent compliance and data governance policies around data sharing. Advances in secure multiparty computation (SMC) for privacy-preserving machine learning (PPML) can help transform these regulated industries by allowing ML computations over encrypted data with personally identifiable information (PII). Yet very little of SMC-based PPML has been put into practice so far. In this paper we present the very first framework for privacy-preserving classification of tree ensembles with application in healthcare. We first describe the underlying cryptographic protocols that enable a healthcare organization to send encrypted data securely to a ML scoring service and obtain encrypted class labels without the scoring service actually seeing that input in the clear. We then describe the deployment challenges we solved to integrate these protocols in a cloud based scalable risk-prediction platform with multiple ML models for healthcare AI. Included are system internals, and evaluations of our deployment for supporting physicians to drive better clinical outcomes in an accurate, scalable, and provably secure manner. To the best of our knowledge, this is the first such applied framework with SMC-based privacy-preserving machine learning for healthcare.**

***Keywords*-privacy-preserving machine learning; secure multiparty computation; encryption; healthcare; random forest; boosted decision trees**

I. Introduction

Data-driven applications are economic drivers of growth and are becoming essential in many domains, including healthcare, infrastructure, and public policy. The data involved is often very sensitive and personal in nature. The importance of protecting the privacy of individuals in a data science ecosystem where large-scale collection and processing are commonplace is central to achieving transformational impact. The need to protect personal privacy in the "era of big data", and the potential to do so successfully in the future, with techniques that allow computation over encrypted data, are widely acknowledged [1], [2].

Industries like healthcare and finance are highly regulated when it comes to data ownership, use and data sharing for analytics. These regulations are always evolving. Organizations that are tasked with custody of personal information, be it patient health or other data, are skeptical of their ability to engage in machine learning (ML), partly due to lack of clarity on policies governing use of such data, and partly due to the fear of unknowingly violating the privacy of individuals that may occur in the process of mining such information [3], [4]. The kind of privacy-preserving machine learning (PPML) solution we offer in this paper can help policy makers understand and explain the potential of ML over encrypted data, and how it may inform future evolution of such regulation. The use of PPML will enable organizations to decrease liability because they will be able to provide ML services over encrypted data, without requiring individuals to expose their personal data to anyone.

In many ML applications, one party – such as a health system or an insurer or a third party cloud platform – possesses a trained ML model, and the need or desire arises to make predictions with that ML model for an input that is held by a second party, which could be a customer or a patient. One such example is a physician using a prediction model to estimate the risk of 30-day hospital readmission of a patient [5], [6]. The problem that we address in this paper is how to classify the second party's input with the first party's classifier such that, at the end of the interaction, the second party is still the only one who knows what their input looks like, and the first party is still the only one who knows what the classifier looks like.

To this end, we use techniques from secure multiparty computation (SMC), an umbrella term for cryptographic approaches that allow two or more parties to jointly compute a specified output from their private information in a distributed fashion, without actually revealing the private information to each other [7]. In particular, we work with secret sharing based solutions in which the parties share their information using a *linear secret sharing scheme*. Next, operations are performed by the parties over the shares till the desired outcome is obtained. The final result can be recovered by combining the final shares, and disclosed as intended, i.e. to one of the parties or to both. While SMC has been hailed as a potential solution to enable PPML [2], [8], very little of it has been put into practice so far [9]. Part of this is due to the fact that while in theory any function can be evaluated on private information from different parties using a secure function evaluation protocol (SFE), the use of conventional SFE protocols typically results in an increase of computation time by up to several orders of magnitude. To enable PPML, domain specific customized SMC techniques need to be developed and implemented.

In this paper we describe (1) how we successfully designed such cryptographic protocols for privacy-preserving classification with tree ensembles [10] – random forests (RFs) and boosted decision trees (boosted DTs) – and (2) how we integrated them in the KenSci ML platform for predicting clinical outcomes of patients.

The large majority of the economic value created by AI today stems from supervised learning applications [11]. Within supervised learning, RFs and boosted DTs are among the most common algorithms of choice for data scientists across the world, because of their wide applicability and their state-of-the-art performance. Our focus on tree ensembles is therefore intentional: to create a widespread impact on applied ML use-cases that are prevalent today. Acknowledging that deployment of PPML solutions requires a non-trivial effort, we describe how the SMC based retooling of the KenSci ML platform required us to rework the KenSci data science workflow. The secure prediction environment has been developed in collaboration with expert physicians at KenSci who currently work with many of the large healthcare systems in the U.S., Europe, and Asia. Our system produces the same precision (hit rate) compared to non-encrypted in-the-clear models, while preserving the privacy of patient records and trained machine learning models. Widespread adoption of solutions such as one presented in this paper will have a significant impact on health outcomes and public policy, and enable global health systems to transition to cloud-based ML systems.

II. Related work

SMC based Machine Learning. A significant body of work in PPML with SMC has focused on the problem of privacy-preserving training of machine learning models (see, e.g., [12], [13], [14], [15], [16] and references therein). Privacy-preserving protocols for predicting with trained ML classifiers – hereafter also referred to as "scoring" – has received far less attention. Non-application specific protocols were designed just lately. Bost et al. [17] introduced privacy-preserving protocols for hyperplane-based, Naive Bayes and DT classifiers, Wu et al. [18] for DTs and RFs, David et al. [19] for hyperplane-based and Naive Bayes classifiers, and De Cock et al. [20] for DTs and hyperplane-based classifiers. De Hoogh et al. [14] had also previously presented a protocol for privacy-preserving scoring of DTs with categorical attributes. The protocol for privacy-preserving scoring of DTs of De Cock et al. [20] cannot be directly used as a building block to obtain random forests and boosted decision trees. We present in Section III a modified protocol for scoring decision trees that does the job.

Regarding tree ensembles, to the best of our knowledge, no protocols have been proposed in the literature for privacy-preserving scoring with boosted DTs. The protocol of Wu et al. [18] for privacy-preserving scoring of RFs relies on an original comparison protocol based on the Paillier encryption scheme and on an oblivious transfer scheme. Both of these building blocks involve expensive public-key cryptography operations. Our solution, on the other hand, only uses additions and multiplications over a finite ring in the online phase.

Differential Privacy. Most work and results concerning privacy-preserving data mining are based on differential privacy (DP), a field in cryptography that aims to maximize the accuracy of answers to queries from databases while minimizing the chances of identifying its records. Before releasing statistics of a dataset, noise is added to prevent an adversary from learning information about any particular individual in the dataset from the aggregate statistics [21]. While DP has been proven very useful in ensuring the privacy of information in data, the need to address privacy risks at the level of computations that manipulate data, and the potential of SMC as a suitable technique for this, have recently been acknowledged [1]. The aim of DP in an ML setting is protecting the privacy of information in the dataset used during training, whereas the focus in this paper is on the use of SMC to protect the trained model and the privacy of new user data that is classified with the model.

Deployed SMC systems for PPML. In contrast to DP, SMC has come to the attention of the data mining and knowledge discovery community only fairly recently, and deployed prototypes are still few and far between. Noteworthy are the endeavors of the ICT company Cybernetica who developed the SMC platform Sharemind, and deployed it in privacy-preserving applications for statistical analysis and fraud detection based on tax records in Estonia [22], [23]. As far as we are aware, there are no deployed SMC based systems yet for privacy-preserving training of ML models, nor for scoring, as we present in this paper.

Model Extraction Attacks. In the system that we present in this paper, we classify one party's input with another party's classifier. At the end of the interaction, the party with the classifier will not have learned anything about the other party's input, and the party with the input will not have learned anything about the other party's classifier (other than the depth of the decision trees, see Section III). At that point, the output of the classification (class label) is typically disclosed to the party who gave the input, and this class label in itself can leak information about the model. It has been shown that popular ML model classes like logistic regression, neural networks, and DTs are vulnerable to model inversion attacks [24] and model extraction attacks [25]. In a model inversion attack, an adversary who has black-box query access to the model uses the revealed labels and the associated confidence scores of the classifier to uncover information about individuals in the training data. In a model extraction attack, an adversary uses the same kind of information to reverse engineer the model. Model ensembles, such as the tree ensembles that we use in this paper, are suspected to be more resilient to such extraction attacks, because their output is an aggregate of a number of individual models [25]. Nevertheless, some countermeasures may be needed to prevent a model inversion or model extraction attack, which is beyond the scope of the current paper. Regardless, it should be clear that (1) the SMC solution presented in this paper computes the class label in a fully secure manner; (2) only the label that is revealed to the party with the input *after* the secure computation might leak some information about the classifier; (3) the party with the classifier does not learn anything about the input of the other party at any point. The privacy of the patient or user who is using the ML scoring service is thus fully protected, even if the trained ML model becomes subject to a model inversion or a model extraction attack.

III. Cryptographic Protocols

A. Problem Description

In many ML applications, one party – such as a hospital or an insurance company – possesses a trained ML model, and the need or desire arises to make predictions with that model for an input that is held by a second party, which could be a customer or a patient. Using terminology common in cryptography, we will henceforth refer to the first party as *Alice*, and the second party as *Bob*. A commonly adopted approach is for Bob (the user) to give his input to Alice (the company), so that Alice can classify the input and return the predicted class label to Bob and/or follow up with actions derived from the knowledge of the class label. Typical examples are a Netflix customer who discloses his preferences in the form of ratings, in return for personalized movie recommendations; a user uploading a photo on Microsoft's how-old.net to get an estimate of the age of the people in the photo; and a healthcare provider using a clinical outcome prediction tool to estimate the risk of 30-day hospital readmission of a patient. In all these cases, the information of the customer, user, or patient (Bob) is fully disclosed to the first party (Alice) holding the machine learning model.

The problem that we address in this paper is how to classify the input held by Bob with the classifier held by Alice in such a way that no one, including Alice, learns Bob's input. A straightforward solution that might come to mind is for Alice to give her classifier to Bob, so that Bob can classify his input locally on his own machine. While this approach would protect Bob's privacy, it is in many cases not a viable solution because it violates Alice's privacy: trained ML models are often proprietary and need to be kept secret in order for the company not loose an important competitive advantage [26]. The challenge faced is therefore how to classify Bob's input with Alice's classifier such that, at the end of the interaction, Bob is still the only one who knows what his input looks like, and Alice is still the only one who knows what her classifier looks like.

In Section III-D we present a cryptographic protocol for solving the problem above in the case when the classifier is a random forest (RF) or a boosted decision trees (boosted DTs) model. We consider *honest-but-curious, static adversaries*, like other privacy-preserving classification protocols so far. A static adversary chooses the set of corrupted parties before the protocol execution. An honest-but-curious adversary follows the instructions of the protocol, but tries to gather additional information. We use as a building block an adaptation of the protocol for privacy-preserving scoring of DTs of De Cock et al. [20], which we recall in Section III-C, after presenting preliminaries about the security model in Section III-B.

B. Cryptographic Preliminaries and Building Blocks

Security Model: We consider the security of our protocols in the Universal Composability (UC) framework [7], [27] due to the fact that this framework considers the security of the protocols under arbitrary composition (i.e., multiple copies of the protocol can be run concurrently to themselves and to other protocols), which covers environments like the Internet. It is the default model for considering the security of cryptographic protocols nowadays. The UC composition theorem ensures that a protocol that is proven UC-secure can be securely executed in such environments. Additionally, the UC framework allows the modular design of complex protocols. A version of the UC composition theorem for the setting with honest-but-curious, static adversaries, is given by Cramer et. al. [7, Theorem 4.20].

Commodity-based Model: The commodity-based model [28] is a setup assumption about the existence of a trusted initializer that pre-distributes correlated randomness during an initialization phase (which can happen far before the protocol execution, even before knowing the inputs) to the parties participating in the protocol. The trusted initializer is not involved in any other part of the execution and does not learn any input from the parties. The main advantage of this model is that it enables very efficient solutions with unconditional security. The commodity-based model allows the realization of non-trivial functionality in the UC framework and has already been used to get very efficient secure computation protocols for tasks such as computing inner-products [29], [30] and other linear algebra operations [31], string equality [30], set intersection [30], oblivious polynomial evaluation [32] and verifiable secret sharing [33]. It was used in protocols for PPML [15], [19], [20]. In practice, this correlated randomness can be distributed by: (1) a single trusted server, (2) many not completely trusted servers (only a majority of honest servers is necessary [28]), or (3) pre-computed by the parties in an offline phase using an SMC protocol to emulate the trusted initializer (in this case the advantage is in offloading the heavy computation to be run at any idle time).

Secret Sharing: We perform SMC using additively secret sharings to do computations modulo q. A value x is secret shared over $\mathbb{Z}_q = \{0, 1, \ldots, q-1\}$ between two parties Alice and Bob by picking $x_A, x_B \in \mathbb{Z}_q$ uniformly at random subject to the constraint that $x = x_A + x_B \mod q$, and then revealing x_A to Alice and x_B to Bob. This secret sharing will be denoted by $[\![x]\!]_q$, which can be thought of as a shorthand for (x_A, x_B). Notice that from the point of view of Alice (resp., of Bob), no information about x is revealed by x_A (resp., by x_B). A secret shared value x can be revealed to one party by sending him the share of the other party.

Building Blocks: We use as a building block Beaver's protocol [34] for multiplication of numbers, denoted by π_{DM}. If two parties Alice and Bob each have shares x_A, y_A, x_B, and y_B of numbers x and y, then they can follow protocol π_{DM} to compute shares z_A and z_B of $z = x \cdot y$.

We also use the protocol π_{DC} for performing secure distributed bitwise comparison due to Garay et al. [35] with secret sharings in the field $\mathbb{Z}_2$. In this case, Alice and Bob have bitwise shares of two numbers x and y, which are expressed as bit strings of length l, and they follow the protocol to determine whether $x \geq y$. π_{DC} outputs a secret sharing of 1 if $x \geq y$ and of 0 otherwise.

Finally we use the protocols for oblivious input selection π_{OIS} and for secure argmax π_{argmax} of De Cock et al. [20]. In the case of π_{OIS}, Alice has as input a vector of values, $\mathbf{x} = (x_1, \ldots, x_n)$, in which each value is expressed as a bit string of length l. Bob has as input k, the index of the desired input value x_k. At the end of the protocol, Alice and Bob have a secret sharing of z_i over $\mathbb{Z}_2$, for $i = 1, \ldots, l$, which are the bits that make up x_k. Bob has not learned any of the values of Alice's vector $\mathbf{x}$, and Alice has not learned which index k Bob was interested in. In the case of π_{argmax}, Alice and Bob have as input bitwise shares of m values that need to be compared. At the end of the protocol, they have a secret sharing of the index of the largest value.

For the proof that these protocols are correct and UC-secure against honest-but-curious adversaries in the commodity-based model, and for the description of optimizations of these protocols (and also for more details), we refer to [20], [36]. We use the same fixed-point representation of Catrina and Saxena [37] to deal with real numbers (i.e., fixed-point precision real numbers are mapped into integers).

C. Decision Trees

For privacy-preserving scoring with decision trees, we propose a novel protocol that improves the method proposed in [20]. In [20], Bob has an input feature vector $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ and Alice has a single decision tree used to classify Bob's feature vector. At the end of the privacy-preserving scoring protocol in [20], the inferred class label is opened to Bob. The case that we consider in this paper is more general: instead of a single decision tree, Alice has an ensemble of decision trees, that have each been trained to output probabilities associated with the class labels (as opposed to only the class labels themselves, as in [20]). Bob should not learn the class label inferred by each individual tree from the ensemble for his input, or their probabilities; only the aggregated result should be disclosed to Bob (see Section III-D). The privacy-preserving scoring protocol π_{DT} for DTs from [20] cannot be directly used in that scenario. We propose a new protocol for scoring decision trees that is different from the one in [20] in two ways: (1) at the end of π_{DT} the output is not opened towards Bob, but instead it is kept as a secret sharing to perform further computations; and (2) instead of having a category associated with each leaf node of the decision tree, we have a probability vector over the class labels, i.e. a class distribution, associated with each leaf node. By using the novel protocol as a building block, we can them obtain privacy preserving scoring of random forests and boosted decision trees.

As stated above, Bob has as input his feature vector $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$. Alice has an ensemble of decision trees, in which each tree is a model $D = (d, G, H, \mathbf{w})$, where d is the depth of the tree, G maps the leaves to the class output $c_i, i = 1, \cdots, k$. Each class output is associated with a class distribution. H maps branch nodes (always considered in level-order) to input features and $\mathbf{w}$ is a vector of thresholds. An example of a DT of depth $d = 3$ is depicted in Figure 1. It is assumed without loss of generality that the binary tree is full, i.e. that it contains all $2^d - 1$ internal nodes (branch nodes) and 2^d leaf nodes. A decision tree can always be filled with dummy nodes to make it full. To this end, a leaf that occurs at level $s < d$ is expanded into a subtree of depth $d - s$ in which all the leaves are copies of the original leaf, and the branch nodes contain a new dummy feature with a randomly chosen threshold. Note that this can be done offline by Alice, before she engages in any privacy-preserving scoring protocol.

Each branch node of a decision tree tests the value of

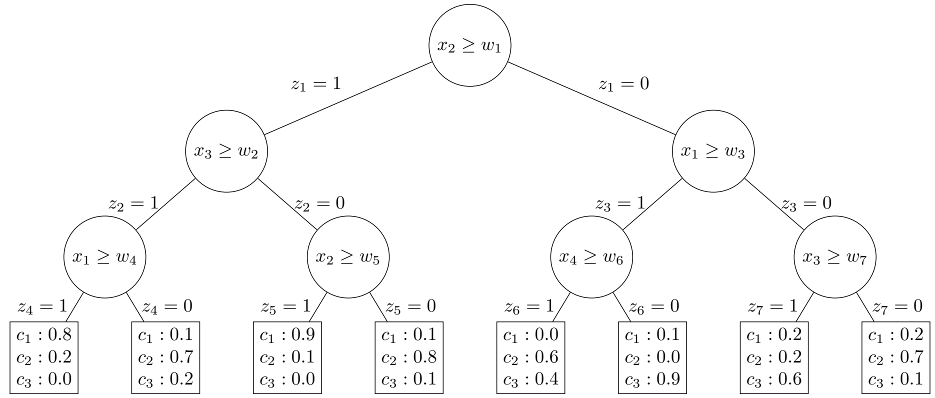
$x_2 \geq w_1$
$z_1 = 1$
$z_1 = 0$
$x_3 \geq w_2$
$x_1 \geq w_3$
$z_2 = 1$
$z_2 = 0$
$z_3 = 1$
$z_3 = 0$
$x_1 \geq w_4$
$x_2 \geq w_5$
$x_4 \geq w_6$
$x_3 \geq w_7$
$z_4 = 1$
$z_4 = 0$
$z_5 = 1$
$z_5 = 0$
$z_6 = 1$
$z_6 = 0$
$z_7 = 1$
$z_7 = 0$
$c_1 : 0.8$ $c_2 : 0.2$ $c_3 : 0.0$
$c_1 : 0.1$ $c_2 : 0.7$ $c_3 : 0.2$
$c_1 : 0.9$ $c_2 : 0.1$ $c_3 : 0.0$
$c_1 : 0.1$ $c_2 : 0.8$ $c_3 : 0.1$
$c_1 : 0.0$ $c_2 : 0.6$ $c_3 : 0.4$
$c_1 : 0.1$ $c_2 : 0.0$ $c_3 : 0.9$
$c_1 : 0.2$ $c_2 : 0.2$ $c_3 : 0.6$
$c_1 : 0.2$ $c_2 : 0.7$ $c_3 : 0.1$

Figure 1. Example of a decision tree of depth $d = 3$ for a classification problem with three classes c_1, c_2 and c_3.

Secure Decision Tree Protocol π_{DT}

Alice has as input a decision tree model $D = (d, G, H, \mathbf{w})$ and Bob has a feature vector $\mathbf{x}$. 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 using π_{OIS} with inputs $x_1, \cdots, x_n$ from Bob and input $H(i)$ from Alice.
2) Let $[p_D(c_1), p_D(c_2), \ldots, p_D(c_k)]$ be a class distribution vector in which $p_D(c_i)$ is the probability that $\mathbf{x}$ belongs to class c_i according to decision tree D. There will be one class distribution vector per possible output of the tree.
3) Alice multiplies the probabilities in the leaf nodes of each decision tree D by a confidence factor α *offline*. Alice them maps these real numbers as integers according to the procedure in [20] and bit-wise shares the resulting weighted probability vectors $\mathbf{p}^i = [\alpha \cdot p_D(c_1), \alpha \cdot p_D(c_2), \ldots, \alpha \cdot p_D(c_k)]$ with Bob, for $i = 1, \ldots, 2^d - 1$.
4) For $i = 1, \ldots, 2^d - 1$, Alice and Bob securely compare $x_{H(i)}$ and w_i. For the input w_i, Alice inputs its bit representation and Bob inputs zeros. Let $\llbracket z_i \rrbracket_2$ denote the result.
5) For $j = 0, \ldots, 2^d - 1$, let $j_d \ldots j_1$ be the binary representation of j with d bits and let $b_k \ldots b_1$ be the one-hot encoding of $G(j + 1) - 1$. For $r = 1, \ldots, k$, initialize $\llbracket y_{j,r} \rrbracket_2$ with the shares $(0, b_r)$. Initialize $u = 1$ and $s = d$. While $s > 0$ do:
   a) For $r = 1, \ldots, k$, $\llbracket y_{j,r} \rrbracket_2 \leftarrow \llbracket y_{j,r} \rrbracket_2 (\llbracket z_u \rrbracket_2 + j_s)$.
   b) Update $u \leftarrow 2u + j_s$ and $s \leftarrow s - 1$.
6) For all $r = 1, \ldots, k$ compute $\llbracket \sigma_r \rrbracket_2 \leftarrow \sum_{j=0}^{2^d-1} \llbracket y_{j,r} \rrbracket_2$
7) Alice and Bob secure compute $\mathbf{q}^i = [\sigma_1 \cdot \alpha \cdot p_D(c_1), \sigma_2 \cdot \alpha \cdot p_D(c_2), \ldots, \sigma_k \cdot \alpha \cdot p_D(c_k)]$ with Bob, for $i = 1, \ldots, 2^d - 1$ by using π_{DM}. Alice and Bob securely add all the resulting vectors $\mathbf{q}^i$ component-wise producing the secret sharing $\llbracket \mathbf{p} \rrbracket_2$ of the weighted probability vector $\mathbf{p}$, the desired output.

Figure 2. The protocol for secure scoring of a decision tree

a particular feature $x_{H(i)}$ against a specified threshold w_i and branches according to the results. Let z_i be the Boolean variable denoting the result of the comparison, i.e. $z_i = 1$ if $x_{H(i)} \geq w_i$, and $z_i = 0$ otherwise. Each leaf node specifies a probability distribution over the k possible classes $c_1, \ldots, c_k$. The classification algorithm for a stand-alone decision tree proceeds as follows:

- Starting from the root node, for the current branch node v_i, evaluate z_i. If $z_i = 1$, take the left branch; otherwise, the right branch.
- When a leaf node is reached, output $G(j)$, where j is the index of the leaf, and $G(j)$ the class distribution corresponding to that leaf, and terminate.

Inspired by the ideas of Bost et al. [17], we perform inference with a decision tree D by evaluating a polynomial P_D: $\{0,1\}^{2^d-1} \rightarrow \{1, \ldots, 2^d\}$. On input $\mathbf{z} = (z_1, \ldots, z_{2^d-1})$, P_D gives the index of the selected leaf level. This polynomial is a sum of terms such that each term corresponds to one possible path in the tree. Given $\mathbf{z}$, the term corresponding to the path taken by $\mathbf{x}$ in the tree evaluates to the inference result (i.e., the index of the leaf), while the remaining terms evaluate to zero.

Similar as in [20], the idea of the secure protocol π_{DT} for scoring decision trees (see Figure Figure 2) is that, for each branch 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. π_{OIS} guarantees that Bob does not learn which feature will be used in the comparison at each branch node, and also that Alice does not learn the values of the features. 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_D using the secure multiplication protocol π_{DM} and local addition of secret sharings. The only information leaked about the tree structure to Bob is its depth d. The full description of protocol π_{DT} is given in Figure 2. The proof that the decision tree protocol π_{DT} is correct and UC-secure against honest-but-curious adversaries in the commodity-based is simmilar to the one in [20], [36].

D. Tree Ensembles

We assume that Alice has an ensemble of decision trees $D_1, D_2, \ldots, D_m$, each with an associated confidence factor or weight $\alpha_1, \alpha_2, \ldots, \alpha_m$, and Bob wants to classify his input $\mathbf{x} = (x_1, \ldots, x_n)$ with this ensemble. For each tree D_j individually, the inference algorithm produces a class distribution vector $[p_{D_j}(c_1), p_{D_j}(c_2), \ldots, p_{D_j}(c_k)]$ in which $p_{D_j}(c_i)$ is the probability that $\mathbf{x}$ belongs to class c_i according to decision tree D_j. To obtain a final classification result, these intermediate results are aggregated as follows:

$$c = \operatorname*{argmax}_{i=1}^{k} \sum_{j=1}^{m} \alpha_j \cdot p_{D_j}(c_i) \tag{1}$$

i.e. the predicted label is that of the class with the highest weighted average of probabilities among the individual decision trees. In the case of random forests, all α_j's are usually 1, while in boosted decision tree models the α_j's can vary, reflecting the confidence in each tree as a correct classifier. Without loss of generality, we can assume that Alice multiplies the probabilities in the leaf nodes of each decision tree D_j with α_j *offline*, before engaging in any computation with Bob. As a result, the outcome of the protocol π_{DT} will be secret sharings of the *weighted* probability vectors $\mathbf{p}_j = [\alpha_j \cdot p_{D_j}(c_1), \alpha_j \cdot p_{D_j}(c_2), \ldots, \alpha_j \cdot p_{D_j}(c_k)]$ thereby stripping away the need for computationally more expensive secure multiplications of the tree weights with the class probabilities.

Our solution for obtaining this final class label in a privacy-preserving manner then works as follows. First, for each tree D_j in the ensemble we use the protocol π_{DT} for scoring $\mathbf{x}$, obtaining as a result a secret sharing of the weighted probability vector, i.e. the probability vector multiplied by the weight α_j of the tree D_j. After that, a secure bitwise addition protocol [38] is used to add these weighted vectors and obtain one accumulator for each of the possible categories (note that these accumulators are still kept as secret sharings). Finally, the secure argmax protocol is executed with these accumulators as input to obtain the most likely category. The full description of protocol π_{TE} is in Figure 3.

The security of protocol π_{TE} follows from the UC-security of the building blocks using the UC composition theorem.

IV. System Design

A. The KenSci Platform

KenSci has developed a novel healthcare-specific AI platform where standardized ML models are used by customers as templates to enable end-to-end solutions for various problems across care management, cost predictions, and operational efficiency in health systems. The design of this platform required KenSci to solve non-traditional ML challenges such as feature standardization for healthcare, data integration across multiple data sources, and setting up cloud-based scalable data pipelines, subject to the constraint that all services need to be compliant with various regulatory standards. Over the years, KenSci has developed many ML models for various use cases, as well as proprietary model orchestration components to be able to score these models at scale on datasets that may contain PHI (protected health information). KenSci also developed innovative ways to expose ML model end-points to backward integrate into health IT systems that are prevalent in the healthcare systems.

Figure 4 and 5 present a high level system overview of a KenSci deployment service for model scoring. Both diagrams depict a *server* (Alice) with a model bank of classifiers, and a *client* (Bob/Healthcare System) that wishes to risk-stratify and score patient data. The KenSci healthcare AI platform acts as an intermediary. Figure 4 depicts the traditional process where the client's data is encrypted at-rest, say in a database, at the client side, as well as in-transit to the server's side. At server side, the data is then decrypted before classification, and a score or class label is generated. This class label or score is then encrypted and returned to the client side, where it is again subsequently decrypted. The next Figure 5 depicts our approach using the cryptographic protocols presented in Section III where the data is kept encrypted during computation as well, in addition to the traditional encryption at-rest (in storage) and in-transit. The fundamental difference between the two approaches in Figure 4 and 5, is that with the approach in Figure 5, the client's data is never exposed to the ML model server.

In-the-clear scoring (Figure 4) can operate without the use of sessions initiated by KenSci between the client and the server; the client can provide all of the required inputs to a traditional model, and receive a score in response. Invocation of the privacy-preserving execution environment as described in Figure 5 requires iterative back-and-forth communication between the client and the server during the scoring operation. To accomplish this, we must guarantee

Secure Tree Ensemble Protocol π_{TE}

Alice has as input decision tree models $D_1, \ldots, D_m$, with $D_j = (d_j, G_j, H_j, \mathbf{w}_j)$, and Bob has a feature vector $\mathbf{x}$. Alice and Bob proceed as follows:

1) For each of the trees $D_1, \ldots, D_m$ in the ensemble, use the protocol π_{DT} to obtain a secret sharing $[\![\mathbf{p}_j]\!]_2$ of the weighted probability vector $\mathbf{p}_j$.
2) Compute $[\![\mathbf{a}]\!]_2 \leftarrow \sum_j [\![\mathbf{p}_j]\!]_2$ using a protocol for secure bitwise addition [38].
3) The secure argmax protocol π_{argmax} is run with the k elements of $\mathbf{a}$ as input and Bob obtains the most likely category as output.

Figure 3. The protocol for secure scoring of a tree ensemble

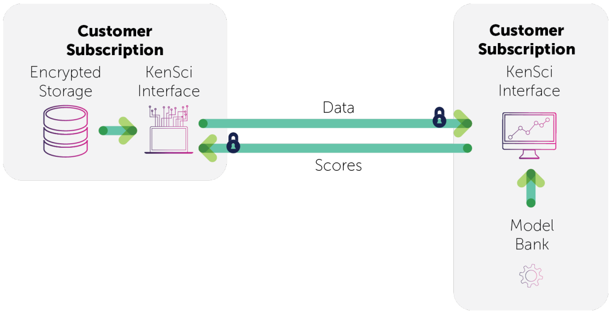

Figure 4. KenSci ML Platform deployment with "in-the-clear" scoring. Data is encrypted during storage and transit, but decrypted before model scoring. Data and models both reside in the customer's cloud subscription.

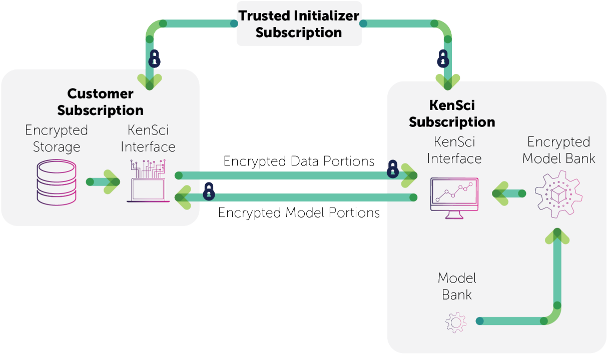

Figure 5. KenSci ML Platform deployment with privacy-preserving scoring. Data is encrypted at-rest (in storage), in-transit, and for scoring computations.

that the intermediate state about the scoring operation is held on both sides of the connection, and the model bank is built in a way to allow this run-time communication. Our solution uses a session-style approach, pairing a client with a model service in favor of other implementations. We have also designed a solution that maintains the tally of calculations done on the model side in a database store on the model cluster, such that any model server can respond to a client's request.

B. From Standard to Encrypted Model Format

As part of our deployment framework we created a capability that allows us to accept any tree-based classifier into the encrypted model bank of the privacy-preserving model execution environment at KenSci. Different adapters were used to convert from native R, Python, or the Predictive Model Markup Language PMML, to the format required by the privacy-preserving model execution environment. These adapters work by iteratively traversing the model's branches (or multiple trees) to produce a standardized representation. Ultimately, the nature of the model's tree (nodes and their weights) must be communicated. Once this format has been produced, it can be associated with the appropriate metadata about a model (model identifier, performance characteristics, and so on) and made consumable to clients.

Decision trees, random forests and boosted decision tree models are all stored as a TreeModel data structure, represented as a dictionary, and saved as a JSON string. This TreeModel contains a list of the class labels, the input features that the model requires, the weights/confidence of the trees, and a list of the trees themselves. The 'weights' field only exists if the model is a boosted decision tree model. The 'classes' field is a list of the resulting labels the model can produce. For each tree in the ensemble, TreeModel contains the following properties:

- **Features:** A list of the feature names for each node in the tree, in level-order. For the tree in Figure 1, this list is $[x_2, x_3, x_1, x_1, x_2, x_4, x_3]$. The same feature can occur more than once. The function $H(i)$ returns the index of the input feature for node i, e.g. $H(5) = 2$. The i^{th} feature corresponds to the i^{th} threshold. There can be duplicate features, but each feature/threshold pair is a unique tuple and generally represents a unique node. Sometimes you may have nodes in different parts of the tree with identical feature/threshold values. We only need to store the tuple once. Let f_i represent the feature name of the i^{th} node. Let the function $H(i)$ return the index of the input feature for node i.
- **Thresholds:** A list of the thresholds for each node of the tree. For the tree in Figure 1, this list is $[w_1, w_2, w_3, w_4, w_5, w_6, w_7]$. The i^{th} threshold, denoted by

w_i, corresponds to the i^{th} feature. There can be duplicate thresholds, but each feature/threshold pair is a unique tuple. Let t_i represent the threshold value of the i^{th} node.

- **Depth:** It is an integer that represents the depth of the tree.
- **Classifiers:** An array of arrays where each internal array maps to the leaf of the tree whose nodal path is represented by the corresponding internal array of the 'polynomial' field. So the i^{th} array of the 'classifier' field is the votes of the leaf attained by the threshold comparison of every node listed in the i^{th} array of the 'polynomial' field evaluating to 1. The length of each internal array is the same as the length of the 'classes' field in the TreeModel, since each element of the classifier's internal arrays is that leaf's vote to the corresponding class. In other words, each internal array corresponds to the probability distribution over the class labels in the leaf of a tree. Let $classifier[i][c]$ be the vote for class c from leaf i. For the tree in Figure 1 this array is $[[0.8, 0.2, 0.0]$, $[0.1, 0.7, 0.2]$, $[0.9, 0.1, 0.0]$, $[0.1, 0.8, 0.1]$, $[0.0, 0.6, 0.4]$, $[0.1, 0.0, 0.9]$, $[0.2, 0.2, 0.6]$, $[0.2, 0.7, 0.1]]$.

C. Implementation

We now present detailed information about the implementation of our protocol described in Section III-D.

1) Requested model and data are piped into the ClientSide secure multiparty computation DT/RF/ADA evaluator as a json string in the following format: $\{$"cmd" : "$score$", "$modelName$" : "$name$", "$modelID$" : "id", "$data$" : $\{$"$feature_1$" : $value_1, \ldots,$ "$feature_n$" : $value_n\}\}$. The model name and id are model identifiers used to identify exactly which type of model the client is requesting.
2) The ClientSide evaluator sends the ServerSide evaluator the list of variable names in their given order of evaluation: $[$"$feature_1$"$, \ldots,$ "$feature_n$"$]$ v
3) Score model. If the model is an Adaboost model and has tree confidence values, we multiply these weights into the classifiers of each tree before starting the multiparty computation scoring on data. Each tree can be scored in parallel, but in our current deployments the tree scoring is serial.
   a) For each tree, the ServerSide evaluator sends the dimensions of the 'polynomial' field to the ClientSide evaluator. This leaks the depth of the tree - the only information that the client ever learns about the model.
   b) For $i = 1, \ldots, 2^d - 1$, the client and the server obtain bitwise secret sharings of $x_{H(i)}$ by executing the protocol π_{OIS} with inputs $x_1, \ldots ; x_n$ from the client and input $H(i)$ from the server.
   c) For $i = 1, ..., 2^d - 1$, securely compare $x_H(i)$ and w_i. For the input w_i, the server inputs its bit representation and the client inputs zeros. Let $[[z_i]]_2$ denote the result.
   d) Create a two dimensional array y such that y[i] contains the bitwise shares of the one hot encoding of $G(i + 1) - 1$ for $i = 0, \ldots, 2^d - 1$. For $i = 0, \ldots, 2^d - 1$, iteratively multiply every bit in y[i] d times where d is the depth of the tree according to Step 5 in Secure Decision Tree protocol in Figure 2.
   e) For $j = 0, \ldots, 2^d - 1$, add the bit shares of y[j] as $TreeOutput[r] = \sum_{j=0}^{2^d-1} y[j][r]$ according to Step 6 in Figure 2. Both the evaluators now hold the shares of the one hot encoding of the output leaf.
   f) For $i = 0, \ldots, 2^d - 1$, do secure bitwise multiplication of the $i'th$ bit in the tree output with the bit representation of the weighted probability vectors of $(i + 1)'th$ leaf. Do a bitwise addition of the weighted probability vectors of 2^d leaf nodes. The evaluators now hold the bitwise shares of the weighted probability vector corresponding to the tree output.
   g) When all the trees have been scored and the shares of each final classifier have been calculated, the corresponding 'votes' from each tree are added together via bitwise addition.
   h) Perform the secure argmax function to select the index k of the highest vote.
   i) Open k to the client. The client can then get $class[k]$ from the known list of classes.

V. Performance

The overall runtime of the protocol is $O(2^d \cdot l \cdot \log(l))$, where l is the bit length used for the feature values, the thresholds in the branch nodes, and the probabilities, and d is the depth of the decision trees. The round complexity, i.e. the number of sequential steps in the protocol (discounting on operations that can be done in parallel) is $O(\log l)$.

To illustrate the practical runtime performance, we apply the protocol to predict the risk that patients run to get an infection after surgery using tree ensemble models as described in [39]. Each patient is characterized by a vector with 94 features, namely their gender, age, and 92 features derived from blood tests done prior to surgery. The task is to predict whether the patient will develop an infection after surgery or not.

Table I shows the time it takes to classify a patient in a privacy-preserving manner with AdaBoost models (ADA) and random forests (RF), with a varying number of trees (10, 50, 100). The predictive accuracy is the same as when classifying without any encryption (i.e. an AUC of around 85%, depending on the model), since the cryptographic protocols perform the same operations as the traditional, unencrypted classification algorithms, thereby obtaining the same class labels. The experiments were run on a 64 bit Linux virtual machine with 72 vCPUs and 144 GB RAM.

Table I
Time required to classify a patient in a privacy-preserving manner; average runtime over 3 runs

Model	Number of Trees	Depth of Trees	Runtime (sec)
ADA	10	1	3
ADA	50	1	10
ADA	100	1	19
RF	10	4	18
RF	50	4	84
RF	100	4	160

Each experiment was repeated 3 times, and the average runtime is recorded in Table I.

In the ADA models, by default, each of the trees in the trained models is a decision stump, i.e. a tree of depth 1. In RF models, each tree is of depth $d = 4$. In all models, the number of bits used for the representation of numbers is $l = 30$.

As the experiments show, the runtime grows with the number of trees as well as with the depth of the trees. With an AdaBoost model of 50 trees, which was the best model in [39], it takes approximately 10 sec to make a prediction for a patient in a fully privacy-preserving manner, allowing secure predictions for 360 patients per hour on a single machine.

VI. Conclusion

In this paper we have presented the first secure multiparty computation (SMC) enabled cryptographic protocols for private classification with tree ensembles – random forests and boosted decision trees – and the deployment of our protocols in the KenSci healthcare analytics platform. To the best of our knowledge this is the very first time a SMC-based privacy-preserving machine learning protocol goes live in a real world scenario. Techniques such as the ones here presented are making it increasingly clear that sacrificing privacy for the benefits of Big Data and AI shouldn't be necessary. From a technological perspective, SMC makes privacy-preserving machine learning possible. The field is still in its infancy, and deployed solutions are few and far between. Our results help bringing this exciting field a step closer to practical use.

Acknowledgment

Kyle Fritchman was employed at KenSci, Inc. while conducting this work. Rafael Dowsley has received funding from the European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreement No 669255 (MPCPRO) and from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731583 (SODA).

References

[1] C. Dwork and G. J. Pappas, "Privacy in information-rich intelligent infrastructure," *arXiv preprint arXiv:1706.01985*, 2017.

[2] Commission of Evidence-Based Policymaking, "The promise of evidence-based policymaking," https://www.cep.gov/content/dam/cep/report/cep-final-report.pdf, 2017.

[3] K. Fiveash, "Google AI given access to health records of 1.6 million english patients," ArsTechnica, https://arstechnica.com/information-technology/2016/05/google-deepmind-ai-nhs-data-sharing-controversy/, 2016.

[4] Royal Free London NHS, "Google Deepmind data agreement with NHS UK," https://drive.google.com/file/d/0BwQ4esYYFC04NFVTRW12TTFFRFE/view, 2016.

[5] S. Basu Roy, A. Teredesai, K. Zolfaghar, R. Liu, D. Hazel, S. Newman, and A. Marinez, "Dynamic hierarchical classification for patient risk-of-readmission," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015, pp. 1691–1700.

[6] R. Caruana, Y. Lou, J. Gehrke, P. Koch, M. Sturm, and N. Elhadad, "Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015, pp. 1721–1730.

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

[8] R. Wyden, "Wyden pushes for stronger security in collection of personal information," https://www.wyden.senate.gov/download/20170515-wyden-mpc-letter-to-cep, 2017.

[9] P. Bogetoft, D. L. Christensen, I. Damgård, M. Geisler, T. Jakobsen, M. Krøigaard, J. D. Nielsen, J. B. Nielsen, K. Nielsen, J. Pagter *et al.*, "Secure multiparty computation goes live," in *International Conference on Financial Cryptography and Data Security*. Springer, 2009, pp. 325–343.

[10] T. G. Dietterich, "Ensemble methods in machine learning," in *International Workshop on Multiple Classifier Systems*, ser. LNCS, vol. 1857. Springer, 2000, pp. 1–15.

[11] A. Ng, "The state of artificial intelligence," MIT Technical Review, https://www.youtube.com/watch?v=NKpuX_yzdYs, 2017.

[12] C. C. Aggarwal and S. Y. Philip, "A general survey of privacy-preserving data mining models and algorithms," in *Privacy-preserving data mining*. Springer, 2008, pp. 11–52.

[13] V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, and N. Taft, "Privacy-preserving ridge regression on hundreds of millions of records," in *2013 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2013, pp. 334–348.

[14] S. de Hoogh, B. Schoenmakers, P. Chen, and H. op den Akker, "Practical secure decision tree learning in a teletreatment application," in *FC 2014*, ser. LNCS, N. Christin and R. Safavi-Naini, Eds., vol. 8437. Springer, Heidelberg, Mar. 2014, pp. 179–194.

[15] M. De Cock, R. Dowsley, A. C. A. Nascimento, and S. C. Newman, "Fast, privacy preserving linear regression over distributed datasets based on pre-distributed data," in *AISec 2015*, 2015.

[16] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu, "Tools for privacy preserving distributed data mining," *ACM SIGKDD Explorations Newsletter*, vol. 4, no. 2, pp. 28–34, 2002.

[17] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data," in *NDSS 2015*. The Internet Society, Feb. 2015.

[18] D. J. Wu, T. Feng, M. Naehrig, and K. E. Lauter, "Privately evaluating decision trees and random forests," *PoPETs*, vol. 2016, no. 4, pp. 335–355, 2016.

[19] B. M. David, R. Dowsley, R. Katti, and A. C. A. Nascimento, "Efficient unconditionally secure comparison and privacy preserving machine learning classification protocols," in *ProvSec 2015*, ser. LNCS, M. H. Au and A. Miyaji, Eds., vol. 9451. Springer, Heidelberg, Nov. 2015, pp. 354–367.

[20] M. De Cock, R. Dowsley, C. Horst, R. Katti, A. Nascimento, W.-S. Poon, and S. Truex, "Efficient and private scoring of decision trees, support vector machines and logistic regression models based on pre-computation," *IEEE Transactions on Dependable and Secure Computing*, vol. PP, no. 99, 2017.

[21] C. Dwork, "Differential privacy: A survey of results," in *International Conference on Theory and Applications of Models of Computation*. Springer, 2008, pp. 1–19.

[22] D. Bogdanov, M. Jõemets, S. Siim, and M. Vaht, "Privacy-preserving tax fraud detection in the cloud with realistic data volumes," T-4-24, Cybernetica AS, https://cyber.ee/en/research/, Tech. Rep., 2016.

[23] D. Bogdanov, L. Kamm, B. Kubo, R. Rebane, V. Sokk, and R. Talviste, "Students and taxes: a privacy-preserving study using secure computation," *Proceedings on Privacy Enhancing Technologies*, vol. 2016, no. 3, pp. 117–135, 2016.

[24] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 1322–1333.

[25] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction APIs," in *USENIX Security Symposium*, 2016, pp. 601–618.

[26] G. Ateniese, L. V. Mancini, A. Spognardi, A. Villani, D. Vitali, and G. Felici, "Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers," *International Journal of Security and Networks*, vol. 10, no. 3, pp. 137–150, 2015.

[27] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," in *42nd FOCS*. IEEE Computer Society Press, Oct. 2001, pp. 136–145.

[28] D. Beaver, "Commodity-based cryptography (extended abstract)," in *29th ACM STOC*. ACM Press, May 1997, pp. 446–455.

[29] R. Dowsley, J. Graaf, D. Marques, and A. C. A. Nascimento, "A two-party protocol with trusted initializer for computing the inner product," in *WISA 10*, ser. LNCS, Y. Chung and M. Yung, Eds., vol. 6513. Springer, Heidelberg, Aug. 2011, pp. 337–350.

[30] Y. Ishai, E. Kushilevitz, S. Meldgaard, C. Orlandi, and A. Paskin-Cherniavsky, "On the power of correlated randomness in secure computation," in *TCC 2013*, ser. LNCS, A. Sahai, Ed., vol. 7785. Springer, Heidelberg, Mar. 2013, pp. 600–620.

[31] 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*, vol. 11, no. 1, pp. 59–73, 2016.

[32] 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*, vol. 14, no. 1, pp. 73–84, 2015.

[33] 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*, vol. 94-A, no. 2, pp. 725–734, 2011.

[34] D. Beaver, "Efficient multiparty protocols using circuit randomization," in *CRYPTO'91*, ser. LNCS, J. Feigenbaum, Ed., vol. 576. Springer, Heidelberg, Aug. 1992, pp. 420–432.

[35] J. A. Garay, B. Schoenmakers, and J. Villegas, "Practical and secure solutions for integer comparison," in *PKC 2007*, ser. LNCS, T. Okamoto and X. Wang, Eds., vol. 4450. Springer, Heidelberg, Apr. 2007, pp. 330–342.

[36] R. Dowsley, "Cryptography based on correlated data: Foundations and practice," Ph.D. dissertation, Karlsruhe Institute of Technology, Germany, 2016.

[37] O. Catrina and A. Saxena, "Secure computation with fixed-point numbers," in *FC 2010*, ser. LNCS, R. Sion, Ed., vol. 6052. Springer, Heidelberg, Jan. 2010, pp. 35–50.

[38] S. J. A. de Hoogh, "Design of large scale applications of secure multiparty computation: Secure linear programming," Ph.D. dissertation, Technische Universiteit Eindhoven, 2012.

[39] P. Mandagani, S. Coleman, A. Zahid, A. Pugel Ehlers, S. Basu Roy, and M. De Cock, "Machine learning models for surgical site infection prediction," in *AMIA KDDM-WG Symposium (American medical Informatics Association Knowledge Discovery and Data Mining Working Group)*, 2016.