Published in F. Kerschbaum and S. Paraboschi, Eds., *Data and Applications Security and Privacy XXXII (DBSec 2018)*, vol. 10980 of *Lecture Notes in Computer Science*, pp. 243–259, Springer, 2018.

Private yet Efficient Decision Tree Evaluation

Marc Joye[1] and Fariborz Salehi[2]

[1] NXP Semiconductors, San Jose, CA, USA
E-mail: marc.joye@nxp.com
[2] California Institute of Technology, Pasadena, CA, USA
E-mail: fsalehi@caltech.edu

Abstract. Decision trees are a popular method for a variety of machine learning tasks. A typical application scenario involves a client providing a vector of features and a service provider (server) running a trained decision-tree model on the client's vector. Both inputs need to be kept private. In this work, we present efficient protocols for privately evaluating decision trees. Our design reduces the complexity of existing solutions with a more interactive setting, which improves the total number of comparisons to evaluate the decision tree. It crucially uses oblivious transfer protocols and leverages their amortized overhead. Furthermore, and of independent interest, we improve by roughly a factor of two the DGK comparison protocol.

Keywords: Data mining; privacy; integer comparison; decision trees.

1 Introduction

Machine learning techniques are currently widely used for many real-world applications. These applications range from spam detection [1], face and pattern recognition [21,25], to the analysis of genome sequencing and financial markets [17,22]. Unfortunately, in many cases, data mining and privacy are perceived to be at odds since the data mining algorithm requires access to the user's information in the clear. Privacy is especially relevant to applications handling sensitive data. As an example, consider the case of a medical study to diagnose a certain disease. In this scenario, medical profiles of patients are considered as highly sensitive data and their usage has to be compliant with regulations [7] such as Health Insurance Portability and Accountability Act (HIPAA).

An important class of machine learning algorithms is known as *classification* where each datapoint belongs to a certain class. The goal is to generate a model that can predict the class of a new datapoint. These models are useful in applications that provide personalized services, such as recommender systems [29], credit scoring models [35], automatic medical assessments [4], etc.

In this paper, we address the problem of *privacy-preserving* classification. We focus on commonly used classifiers: *decision trees*. Decision trees are simple classifiers that consist of a collection of decision nodes in a tree structure. A classical example is the twenty-question game where one player has in mind some

object and another player tries to guess the object with no more than 20 yes-or-no questions. Decision trees are non-linear models for classification, yet they are easy to interpret since their evaluation simply corresponds to a tree traversal.

The *secure evaluation* of decision trees involves two parties. A server possesses a decision-tree model and a client wishes to evaluate the model. This is a typical setting in a cloud-based query system, where the service provider has a model which was trained by integrating the data of thousands of users and the client wants to learn the output of the model for her input data. An evaluation protocol is said to be secure, when at the conclusion of the protocol execution, the server cannot learn anything about the client's data and the client cannot learn additional information about the server's model.

The output of a decision-tree model is computed by traversing the tree, level by level. At each level, an entry of the client's input is compared against a fixed threshold and the result indicates how to traverse to the next level. The comparison at each visited node has to be performed in a secure way, otherwise there would be information leakage about the client's input and/or the server's model. At the heart of our privacy-preserving decision-tree evaluation lies an efficient protocol for the secure comparison of private values. It is worth mentioning that comparison is an essential building block for developing many other secure machine learning algorithms. These include clustering [8], support vector machines (SVM) [32], matrix factorization [26], regression [27], and neural networks [23]. Hence, the proposed comparison algorithms can improve the performance of a wide range of applications.

Related work. Privacy-preserving data mining was introduced in [2,20,12]. These works present different approaches to securely construct decision trees. Protocols for the private evaluation of decision trees were subsequently developed in [5] and more recently in [7,34,31]. In [7], Bost *et al.* express the decision tree as a polynomial whose output is the result of the classification. Their representation requires a small number of multiplications and is evaluated using a fully homomorphic encryption scheme. Wu *et al.* [34] reduce the problem of decision-tree evaluation to the oblivious transfer of a leaf node. Assuming a complete decision tree, they hide its structure to the client by applying a random permutation. They so gain an order of magnitude reduction in client computation and bandwidth. Tai *et al.* [31] replace the evaluation step in [34] via linear functions. This leads to better performance for sparse decision trees. Finally, we note that [34] and [31] also introduce extended protocols that are made secure against malicious adversaries.

In [9,10,11], Damgård, Geisler, and Krøigaard (DGK) present an elegant two-party protocol for comparing private values. It was later modified in [13] and [33], and adapted in [7,34]. It relies on additively homomorphic encryption. The DGK protocol and its variants are dominated by exponentiations in the group underlying the homomorphic encryption scheme. Those are costly operations. Another drawback in the DGK protocol is the communication cost. The former issue was addressed by Veugen in [33]. The author was able to divide the computational workload by approximately a factor of two, *on average*. Unfortunately, the resulting implementation is subject to timing attacks [16].

Our contributions. We devise privacy-preserving comparison protocols that reduce by roughly a factor of two *both* the computational complexity and the necessary bandwidth. Furthermore, unlike [33], provided proper implementation the proposed protocols are made resistant against timing attacks.

Another contribution of this work is a new protocol for evaluating a decision tree model. We borrow from [34] the astute idea of hiding the indexes of the comparison nodes using a random permutation at each level of the tree. However, we reduce the number of comparisons with a more interactive setting. Doing so, we also take advantage of the amortized complexity of efficient OT protocols. The works of [34] and [31] require a comparison for every internal node. In our setting, a single comparison per level is required. For a decision tree of depth d, this amounts to a total of d comparisons. This has to be compared with the m comparisons in [31,34], where $m \gg d$ is the number of internal nodes.

Paper outline. The rest of the paper is organized as follows. The next section introduces some cryptographic tools. Sections 3 and 4 are the core of the paper. They present a new design for evaluating decision trees in a privacy-preserving fashion, making use of an enhanced comparison protocol. The security and performance are discussed in Section 5. Finally, Section 6 concludes the paper.

2 Cryptographic Tools

2.1 Additively homomorphic encryption

An *additively homomorphic encryption scheme* [30] consists of a tuple of four algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{AddH})$. On input a security parameter κ, the key generation algorithm KeyGen returns a matching pair (pk, sk) of public key and secret key. Let $\mathcal{M}$ denote the message space. The encryption algorithm Enc is a randomized algorithm that takes as input pk and a plaintext $m \in \mathcal{M}$, and returns a ciphertext c. Given a valid ciphertext c, the decryption algorithm Dec, using sk, returns the corresponding plaintext m.

For homomorphic encryption, the message space $\mathcal{M}$ is modeled as a finite ring. Additional public-key algorithm AddH operates on ciphertexts. It takes as input the encryption of two messages $m, m' \in \mathcal{M}$ and returns an encryption of $m + m'$. When the public key is clear from the context, it is customary to write an encryption of m as $[\![m]\!]$ in lieu of $\mathsf{Enc}_{pk}(m)$. We then use the 'boxplus' operator ($\boxplus$) to denote the addition of two ciphertexts. Hence, an encryption of $m+m'$ is obtained as $[\![m+m']\!] = [\![m]\!] \boxplus [\![m']\!]$. Likewise, for a known constant d, the encryption of $d \cdot m$ can be obtained from the encryption of m as $[\![d \cdot m]\!] = d \cdot [\![m]\!]$; *i.e.*, as $\sum_{i=1}^{d} [\![m]\!] = [\![m]\!] \boxplus [\![m]\!] \cdots \boxplus [\![m]\!]$ (d times). Finally, we write $[\![m]\!] \boxminus [\![m']\!]$ for $[\![m]\!] \boxplus [\![-m']\!] = [\![m - m']\!]$.

2.2 Oblivious transfer

Oblivious transfer (OT) [28,14] is a two-party protocol between a chooser and a sender. On a 1-out-of-N OT, the sender has a set of N t-bit strings $\{\sigma_0, \sigma_1, \ldots,$

$\sigma_{N-1}\}$. The chooser selects an index $j \in \{0, 1, \ldots, N-1\}$ and exactly obtains from the sender the string σ_j in an oblivious way (*i.e.*, the sender does not know the value of j). Oblivious transfer protocols can be constructed from many cryptographic assumptions. Efficient implementations are provided in [24,3]; see also Appendix A.

3 Private Comparison of Integers

In this section, we introduce our enhanced design for the secure comparison of t-bit values based on additively homomorphic encryption. To make the presentation easier to follow, we describe it in stages. We start with a basic protocol which is not secure when some prior information is known. We then extend it to get full security regardless of the inputs.

3.1 Basic protocol

The setting is as follows. Each party possesses a private t-bit value: Party A (Alice) has $x = \sum_{i=0}^{t-1} x_i\, 2^i$ while party B (Bob) has $y = \sum_{i=0}^{t-1} y_i\, 2^i$. The goal for parties A and B is to respectively obtain at the conclusion of the protocol bits δ_A and δ_B such that $\delta_A \oplus \delta_B = \mathbb{1}\{x \leqslant y\}$. Neither party can learn anything more about the other party's input.

We depict in Fig. 1 the protocol by describing the different steps performed by the two parties. Party B is equipped with an additively homomorphic public-key encryption scheme. We let $[\![m]\!]$ denote the encryption of a message $m \in \mathcal{M}$ under B's public key; see Sect. 2.1. The message space $\mathcal{M}$ is assumed to be a finite integral domain and to satisfy $\#\mathcal{M} \geqslant t+1$.

Remark 1. In Step 3 (Fig. 1), note that given $[\![y_i]\!]$, A can obtain $[\![x_i \oplus y_i]\!]$ as $[\![y_i]\!]$ if $x_i = 0$, and as $[\![1]\!] \boxminus [\![y_i]\!]$ if $x_i = 1$.

To show the correctness of the protocol, it is useful to introduce some notation. For a t-bit integer $a = \sum_{i=0}^{t-1} a_i\, 2^i$ with $a_i \in \{0,1\}$, we let $\overline{a}$ denote the complement of a; *i.e.*, $\overline{a} = 2^t - a - 1$. In particular, for $t = 1$, $a = a_0$ and $\overline{a} = \overline{a_0} = 1 - a_0$. With this notation, we can reformulate an observation made in [9, Section 3].

Proposition 1. *Let $x = \sum_{i=0}^{t-1} x_i\, 2^i$ and $y = \sum_{i=0}^{t-1} y_i\, 2^i$, with $x_i, y_i \in \{0,1\}$, be two t-bit integers. Define*

$$\begin{cases} c_{-1} = \sum_{j=0}^{t-1} (x_j \oplus y_j)\,, \\ c_i = x_i + \overline{y_i} + \sum_{j=i+1}^{t-1} (x_j \oplus y_j) \quad \textit{for } 0 \leqslant i \leqslant t-1\,. \end{cases}$$

Then $x < y$ if and only if there exists some unique index i with $0 \leqslant i \leqslant t-1$ such that $c_i = 0$. Moreover, $x = y$ if and only if $c_{-1} = 0$ and $c_i = 1$ for $0 \leqslant i \leqslant t-1$.

Proof. As defined, c_i is the sum of nonnegative terms. Therefore, for $0 \leqslant i \leqslant t-1$, $c_i = 0$ is equivalent to (i) $x_i = \overline{y_i} = 0$ and (ii) for $i+1 \leqslant j \leqslant t-1$, $x_j \oplus y_j = 0$.

Fig. 1. Basic comparison protocol.

1. Party B encrypts the bits of $y = \sum_{i=0}^{t-1} y_i\, 2^i$ under his public key and sends $[\![y_i]\!]$, $0 \leqslant i \leqslant t-1$, to A.
2. Party A computes the Hamming weight of x (*i.e.*, the number of nonzero bits of x). Let h denote the Hamming weight of x. There are three cases to consider:
   (a) if $h > \lfloor t/2 \rfloor$, A sets $\delta_A = 0$;
   (b) if $h < \lceil t/2 \rceil$, A sets $\delta_A = 1$;
   (c) if $h = t/2$ (this can only occur when t is even), A chooses a random value in $\{0,1\}$ for δ_A.
3. Next, party A forms a set $\mathscr{L}$ of indexes i such that
   (a) $\mathscr{L} \supseteq \mathscr{L}'$ where $\mathscr{L}' = \{0 \leqslant i \leqslant t-1 \mid x_i = \delta_A\}$; and
   (b) $\#\mathscr{L} = \lfloor t/2 \rfloor$.
   For each $i \in \mathscr{L}$, A draws at random a non-zero element $r_i \in \mathcal{M}$ and computes
   $$[\![c_i^*]\!] = r_i \cdot \Big([\![1 + (1-2\delta_A)x_i]\!] \boxplus \big((2\delta_A - 1) \cdot [\![y_i]\!]\big) \boxplus \big(\textstyle\sum_{j=i+1}^{t-1} [\![x_j \oplus y_j]\!]\big) \Big) \ .$$
   Finally, she computes
   $$[\![c_{-1}^*]\!] = r_{-1} \cdot \Big([\![\delta_A]\!] \boxplus \textstyle\sum_{j=0}^{t-1} [\![x_j \oplus y_j]\!] \Big)$$
   for a random non-zero element $r_{-1} \in \mathcal{M}$. Party A sends the $\lfloor t/2 \rfloor + 1$ ciphertexts $[\![c_i^*]\!]$ *in a random order* to B.
4. Using his private key, party B decrypts the received $[\![c_i^*]\!]$'s. If one is decrypted to zero, B sets $\delta_B = 1$. Otherwise, he sets $\delta_B = 0$.

This in turn is equivalent to (i) $x_i < y_i$ and (ii) for $i+1 \leqslant j \leqslant t-1$, $x_j = y_j$; that is, $x < y$. To see that index i such that $c_i = 0$ is unique, suppose that $c_{i'} = 0$ for some $i' \neq i$. Without loss of generality, assume that $i' < i$. This leads to $c_{i'} = x_{i'} + \overline{y_{i'}} + \sum_{j=i'+1}^{t-1}(x_j \oplus y_j) \geqslant x_i \oplus y_i = 1$, a contradiction.

The second part of the proposition is clear. If $\sum_{j=0}^{t-1} x_j\, 2^j = \sum_{j=0}^{t-1} y_j\, 2^j$ then $c_{-1} = 0$ and $c_i = 1$ for $i \geqslant 0$. □

By reversing the roles of x and y in Proposition 1, we get as an immediate corollary the following proposition.

Proposition 2. *Let* $x = \sum_{i=0}^{t-1} x_i\, 2^i$ *and* $y = \sum_{i=0}^{t-1} y_i\, 2^i$, *with* $x_i, y_i \in \{0,1\}$, *be two* t*-bit integers. For* $0 \leqslant i \leqslant t-1$, *define*

$$c_i = y_i + \overline{x_i} + \textstyle\sum_{j=i+1}^{t-1}(y_j \oplus x_j) \ .$$

Then $x \leqslant y$ *if and only if there exists no index* i *with* $0 \leqslant i \leqslant t-1$ *such that* $c_i = 0$.

Proof. If there were such an index i, this would imply $y < x$ by Proposition 1. The absence of such an index therefore implies $y \geqslant x$. □

We are now ready to show that the protocol must terminate with the correct result. Following [33], depending on the value of h (see Step 2 in Fig. 1), we distinguish three cases.

1. Suppose first that the Hamming weight of x is greater than $\lfloor t/2 \rfloor$ (and thus $\delta_A = 0$). This means that x has more ones than zeros in its binary representation. Specifically, among the t bits of x, at most $\lfloor t/2 \rfloor$ bits are equal to 0. Furthermore, for $0 \leqslant i \leqslant t-1$, Proposition 1 shows that c_i needs only to be evaluated when $x_i = 0$ since when $x_i = 1$ we already know that the corresponding c_i cannot be zero. The case $x = y$ is taken into account using c_{-1}.
2. Now suppose that the Hamming weight of x is less than $\lceil t/2 \rceil$ (and thus $\delta_A = 1$). In this case, among the t bits of x, at most $\lfloor t/2 \rfloor$ bits are equal to 1. We can then make use of Proposition 2. With at most $\lfloor t/2 \rfloor$ tests for $c_i = 0$ (*i.e.*, when $x_i = 1$), we can decide whether $x \leqslant y$.
3. The last case is when the Hamming weight of x is $t/2$ (and thus δ_A is equiprobably equal to 0 or 1). This supposes t even. In this case, among the t bits of x, $t/2$ bits are equal to 0 and $t/2$ bits are equal to 1. Proposition 1 or Proposition 2 can be used indifferently to decide after at most $t/2 = \lfloor t/2 \rfloor$ tests for $c_i = 0$ whether $x \leqslant y$.

The above analysis shows that (i) only the indexes $i \in \mathscr{L}'$ need to be tested, and (ii) $\#\mathscr{L}' \leqslant \lfloor t/2 \rfloor$. If $\#\mathscr{L}' < \lfloor t/2 \rfloor$ then additional indexes are added to $\mathscr{L}'$ to form $\mathscr{L}$. This ensures that $\#\mathscr{L}$ is always equal to $\lfloor t/2 \rfloor$ and is aimed at preventing timing attacks. Now the correctness follows by noting that the $[\![c_i^*]\!]$'s include the encryptions of $r_i \cdot c_i$ for all $i \in \mathscr{L}'$. It is also important to see that $[\![c_{-1}^*]\!]$ is the encryption of a non-zero value when $\delta_B = 1$.

By construction, $\delta_B = 1$ if one of the $[\![c_i^*]\!]$'s decrypts to 0.

- When $\delta_A = 0$, Proposition 1 is used. A decryption to 0 means $x \leqslant y$. We therefore have $\mathbb{1}\{x \leqslant y\} = 1 = \delta_A \oplus \delta_B$, as desired.
- When $\delta_A = 1$, Proposition 2 is used and a decryption to 0 means $x \not\leqslant y$. Then, $\mathbb{1}\{x \leqslant y\} = 0 = \delta_A \oplus \delta_B$, as desired.

If none of the $[\![c_i^*]\!]$'s decrypts to 0 then $\delta_B = 0$. When $\delta_A = 0$, this means $x \not\leqslant y$; when $\delta_A = 1$, this means $x \leqslant y$. In both cases, we have $\mathbb{1}\{x \leqslant y\} = \delta_A \oplus \delta_B$, as desired.

3.2 Full protocol

The basic protocol needs special care. In particular, it requires that the Hamming weight of x *a priori* has the same probability to be greater than $\lfloor t/2 \rfloor$ or less than $\lceil t/2 \rceil$. This guarantees that δ_A is uniformly distributed over $\{0, 1\}$. Indeed, if party B knows for example that the Hamming weight of x is more likely less than $\lceil t/2 \rceil$ (and thus δ_A is more likely equal to 1), a value $\delta_B = 0$ tells party B that x is more likely less than or equal to y since $\delta_A \oplus \delta_B = \mathbb{1}\{x \leqslant y\}$.

We modify our basic protocol so that it remains secure when party B has some prior knowledge on the Hamming weight of x. The resulting distribution of

δ_A will always be uniform over $\{0,1\}$, independently of the value of x. The full protocol is detailed in Fig. 2.

Fig. 2. Full comparison protocol.

1. (a) Party B generates a random mask $\eta \in \mathbb{Z}_{2^t}$, forms $y^* = y + \eta \bmod 2^t$ and $Y^* = \lfloor \frac{y+\eta}{2^t} \rfloor$, and sends y^* to A.
   (b) Likewise, party A generates a random t-bit integer $x' \in \mathbb{Z}_{2^t}$, computes $z^* = y^* + x' - x \bmod 2^t$ and $Z^* = \lfloor \frac{y^*+x'-x}{2^t} \rfloor$, and sends z^* to B.
   (c) Party B removes the mask and defines the t-bit integer $y' = z^* - \eta \bmod 2^t$. B also defines $Y' = \lfloor \frac{z^*-\eta}{2^t} \rfloor$.
2. Parties A and B apply the basic comparison protocol (Fig. 1) on t-bit integers x' and y'.
   Let δ'_A and δ'_B denote the respective outputs for A and B of the protocol, with $\delta'_A \oplus \delta'_B = \mathbb{1}\{x' \leqslant y'\}$.
3. Party A sets $\delta_A = \delta'_A$ if Z^* is even, and $\delta_A = \delta'_A \oplus 1$ otherwise.
4. Party B sets $\delta_B = \delta'_B$ if $(Y^* + Y')$ is even, and $\delta_B = \delta'_B \oplus 1$ otherwise.

It is worth remarking that x' as defined in Step 1b (Fig. 2) is a *random* t-bit integer. There is therefore no way for party B to gain more information on its Hamming weight.

The correctness of the protocol is easily verified. By definition, we have $y^* = y + \eta - 2^t Y^*$, $z^* = y^* + x' - x - 2^t Z^*$, and $y' = z^* - \eta - Y' 2^t$. This leads to

$$\begin{aligned} \delta_A \oplus \delta_B = \mathbb{1}\{x \leqslant y\} &= \left\lfloor \tfrac{y+2^t-x}{2^t} \right\rfloor = \left\lfloor \tfrac{y'+2^t-x'}{2^t} \right\rfloor + Y^* + Y' + Z^* \\ &= (\delta'_A \oplus \delta'_B) + Y^* + Y' + Z^* \ . \end{aligned}$$

Reducing the above relation modulo 2 yields $\delta_A + \delta_B \equiv \delta'_A + \delta'_B + Y^* + Y' + Z^* \pmod 2$, a solution of which is $\delta_A = \delta'_A + Z^* \bmod 2$ and $\delta_B = \delta'_B + Y^* + Y' \bmod 2$.

3.3 Further settings

Encrypted comparison bit. Let δ denote the comparison bit; *i.e.*, $\delta = \mathbb{1}\{x \leqslant y\}$. In certain settings, a party wishes to produce an encryption of δ at the end of the protocol, rather than a share δ_A of δ (the other share, δ_B, being held by the other party). In this case, we can add the following step to our comparison protocols:

5. Party B encrypts δ_B using his public key and sends $[\![\delta_B]\!]$ to A. Upon receiving $[\![\delta_B]\!]$, party A computes the encryption of δ as $[\![\delta]\!] = [\![\delta_B]\!]$ if $\delta_A = 0$, and $[\![\delta]\!] = [\![1]\!] \boxminus [\![\delta_B]\!]$ otherwise.

Encrypted inputs. There exists another practical setting for the comparison of private inputs. Suppose that one party possesses $[\![x]\!]$ and $[\![y]\!]$, the encryption of two t-bit values $x = \sum_{i=0}^{t-1} x_i\, 2^i$ and $y = \sum_{i=0}^{t-1} y_i\, 2^i$. The other party possesses the corresponding decryption key. Our protocols easily generalize to cover this setting as well. An example is given in Sect. 4.2.

Other frameworks. The technique we employed is fairly generic and can be adapted to increase the efficiency of other bit-wise comparison protocols, including the protocol in [18].

4 Application: Private Evaluation of Decision Trees

Secure comparison protocols find numerous practical applications. We apply the results of the previous section to the *private* evaluation of decision trees. As the values being compared will be random, our basic protocol (Fig. 1) suffices.

4.1 Problem setup

There are two parties involved: a client and a server. The client has a private feature vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{Z}^n$ and the server possesses a decision tree model $\mathfrak{T}\colon \mathbb{Z}^n \to \mathbb{Z}$. At the end of protocol, the client obtains the value $z_r := \mathfrak{T}(\boldsymbol{x})$ and learns nothing else; the server learns nothing.

In a binary tree, each internal node $\nu_k^{(\ell)}$ (with $0 \leqslant k \leqslant \ell$) at level ℓ in the tree is associated with a Boolean function

$$f_k^{(\ell)}(\boldsymbol{x}) = \mathbb{1}\big\{x_{i_k^{(\ell)}} \leqslant T_k^{(\ell)}\big\}\,, \tag{1}$$

where $i_k^{(\ell)}$ is an index in the feature vector $\boldsymbol{x} \in \mathbb{Z}^n$, and $T_k^{(\ell)}$ is a threshold.

The depth of the tree (*i.e.*, the longest path from the root to a leaf) is denoted by d. The number of internal nodes is denoted by m. Without loss of generality, we assume that $\mathfrak{T}$ is a *complete* binary decision tree; that is, a binary decision tree with exactly 2^ℓ nodes at each level ℓ. We note that it is easy to derive a complete binary decision tree by introducing *dummy* internal nodes and assigning an arbitrary value in $\{0, 1\}$ for the corresponding Boolean function $f_k^{(\ell)}(\boldsymbol{x})$. This is illustrated in the figure below on a decision tree of depth $d = 2$.

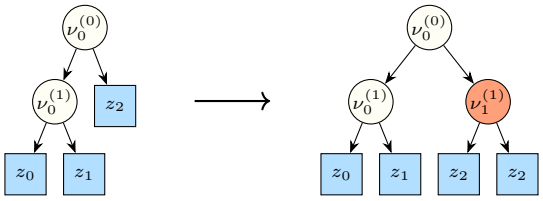

Fig. 3. Transforming a binary decision tree into a complete binary decision tree.

4.2 From public to private evaluation

On input $\boldsymbol{x}$, the evaluation of a decision tree starts at the root node. At each level ℓ, depending on the result of $f_k^{(\ell)}(\boldsymbol{x})$, either the left branch (for 0) or the right branch (for 1) is taken. The process is repeated until a leaf node is reached. The output of $\mathfrak{T}(\boldsymbol{x})$ is z_r, the value of the so-obtained leaf node.

Public evaluation. When the feature vector $\boldsymbol{x}$ and the decision tree $\mathfrak{T}$ are available in the clear, the decision tree can be evaluated by performing d comparisons. Let $\beta_\ell \in \{0,1\}$ denote the result of the decision (0 or 1) at level ℓ, for $\ell = 0, 1, \ldots, d-1$. It turns out that

$$\begin{cases} \beta_0 = f_0^{(0)}(\boldsymbol{x}) \\ \beta_\ell = f^{(\ell)}_{(\beta_0,\ldots,\beta_{\ell-1})_2}(\boldsymbol{x}) \quad \text{for } \ell = 1, \ldots, d-1 \end{cases} . \tag{2}$$

Consequently, the index r of the corresponding leaf node can be expressed as $r = (\beta_0, \beta_1, \ldots, \beta_{d-1})_2 = \sum_{\ell=0}^{d-1} \beta_\ell \, 2^{d-1-\ell}$, where $(\beta_0, \beta_1, \ldots, \beta_{d-1})_2$ represents the binary expansion of r.

Example 1. Figure 4 depicts an example of a binary decision tree with 4 levels. In this example, the index r of the output, z_r, is given by $r = (\beta_0, \beta_1, \beta_2, \beta_3)_2 = (0, 1, 0, 1)_2 = 5$.

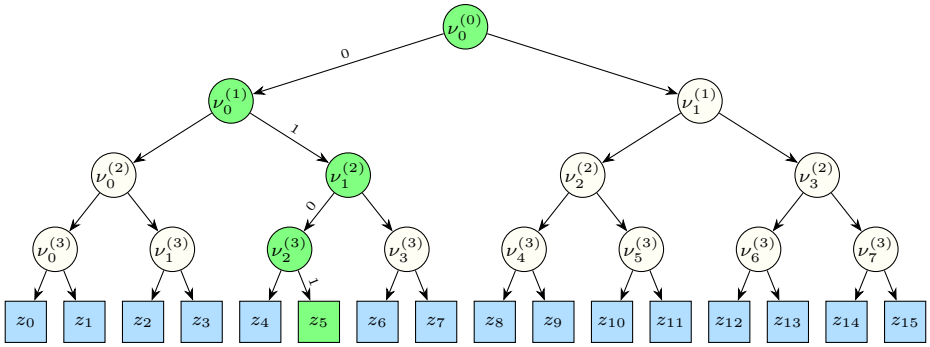

Fig. 4. Evaluation of a decision tree.

Private evaluation. In the private setting, the server knows the model $\mathfrak{T}$ (including $i_k^{(\ell)}$ and $T_k^{(\ell)}$, for $0 \leqslant \ell \leqslant d-1$) while the client knows $\boldsymbol{x} = (x_1, \ldots, x_n)$.

– For $\ell = 0$, we have $\beta_0 = \mathbb{1}\{x_{i_0^{(0)}} \leqslant T_0^{(0)}\}$. However, the private comparison protocols of Sect. 3 do not directly apply because the value of $i_0^{(0)}$ is unknown to the client. This issue is resolved by providing the server with $[\![x_i]\!]$, for $1 \leqslant i \leqslant n$. Here, $[\![x_i]\!]$ denotes an encryption[3] of x_i under the public-key of the

[3] We use triple brackets rather than double brackets to indicate that the encryption scheme may be different from the one used for the comparisons.

client. The encryption scheme is supposed being additively homomorphic with message space $\mathcal{M}$ such that $\#\mathcal{M} \geqslant 2^{t+\kappa}$ for a certain security parameter κ. Using the techniques developed in the previous section, the client and server can now engage in a two-party protocol to secret-share the decision bit $\beta_0 = b_0 \oplus b'_0$ —where the server holds b_0 and the client holds b'_0. Details are provided in Step 2 of Fig. 6.

– For $\ell = 1, \ldots, d-1$, Equations (1) and (2) become $\beta_\ell = \mathbb{1}\{x_{i^{(\ell)}_{k^\star}} \leqslant T^{(\ell)}_{k^\star}\}$ with $k^\star := k^\star(\ell) = (\beta_0, \ldots, \beta_{\ell-1})_2$. In particular, for $\ell = 1$, we obtain

$$\beta_1 = \mathbb{1}\{x_{i^{(1)}_{k^\star}} \leqslant T^{(1)}_{k^\star}\} \quad \text{with } k^\star = \beta_0 = \begin{cases} b'_0 & \text{if } b_0 = 0 \\ b'_0 \oplus 1 & \text{otherwise} \end{cases} .$$

Specifically, if $b_0 = 0$, the server knows that the client possesses the correct result of the comparison; *i.e.*, $b'_0 = \beta_0$. If $b_0 = 1$, the server knows that the client possesses the flipped result. To maintain the consistency, the server uses a copied version $\mathfrak{T}^*$ of the initial tree. If $b_0 = 1$, the server updates $\mathfrak{T}^*$ by switching the left subtree and the right subtree at level $\ell = 1$. What is important to observe here is that $k^\star$ coincides with b'_0 in $\mathfrak{T}^*$. Hence, the client can obtain $[\![x_{i^{(1)}_{k^\star}} - T^{(1)}_{k^\star} + \mu_1]\!]$ —and in turn $x_{i^{(1)}_{k^\star}} - T^{(1)}_{k^\star} + \mu_1$ after decryption— from the server, where μ_1 is a mask chosen by the server to hide the value of $x_{i^{(1)}_{k^\star}} - T^{(1)}_{k^\star}$. Next, the client and server engage in a two-party protocol to secret-share the decision bit $\beta_1 = b_1 \oplus b'_1$. To prevent the server to learn the index $k^\star$, $[\![x_{i^{(1)}_{k^\star}} - T^{(1)}_{k^\star} + \mu_1]\!]$ is obtained via oblivious transfer. Again, refer to Step 2 in Fig. 6 for details.
The same process is iterated for $\ell = 2, \ldots, d-1$. Each time $b_\ell = 1$, the server switches *all* subtrees of $\mathfrak{T}^*$ at level ℓ and calls $\mathfrak{T}^*$ the so-obtained tree.

– At this stage the client knows $(b'_0, \ldots, b'_{d-1})_2$, which is the index of the leaf node containing the result in the permuted tree $\mathfrak{T}^*$. The client engages in a 1-out-of-2^d OT with the server and thereby learns z_r.

Example 2 (Example 1 cont'd). Suppose that the server successively obtains $b_0 = 1$, $b_1 = 0$, $b_2 = 1$, and $b_3 = 1$. For $\ell = 1, \ldots, 3$: if $b_\ell = 1$ the subtrees at level ℓ are switched. This is illustrated in Fig. 5. The bottom picture is the final permuted tree $\mathfrak{T}^*$.

Our decision-tree evaluation protocol is given in Figure 6. The permuted tree $\mathfrak{T}^*$ is represented at level $\ell \geqslant 1$ by the string $\sigma^{(\ell)} = (b_0, \ldots, b_{\ell-1})_2$; $\mathfrak{T}^* = \mathfrak{T}$ for $\ell = 0$. Step 2d in Fig. 6 outputs shares of the decision at level ℓ. It is worth noting that a *single* execution of the comparison protocol is run per level.

Proposition 3. *With the notation of Fig. 6, for $0 \leqslant \ell \leqslant d-1$, the server and the client secret-share the decision bit at each level; i.e.,*

$$b_\ell \oplus b'_\ell = \beta_\ell = \mathbb{1}\left\{x_{i^{(\ell)}_{k^\star}} \leqslant T^{(\ell)}_{k^\star}\right\} \quad \textit{where } k^\star = \sigma^{(\ell)} \oplus j \ .$$

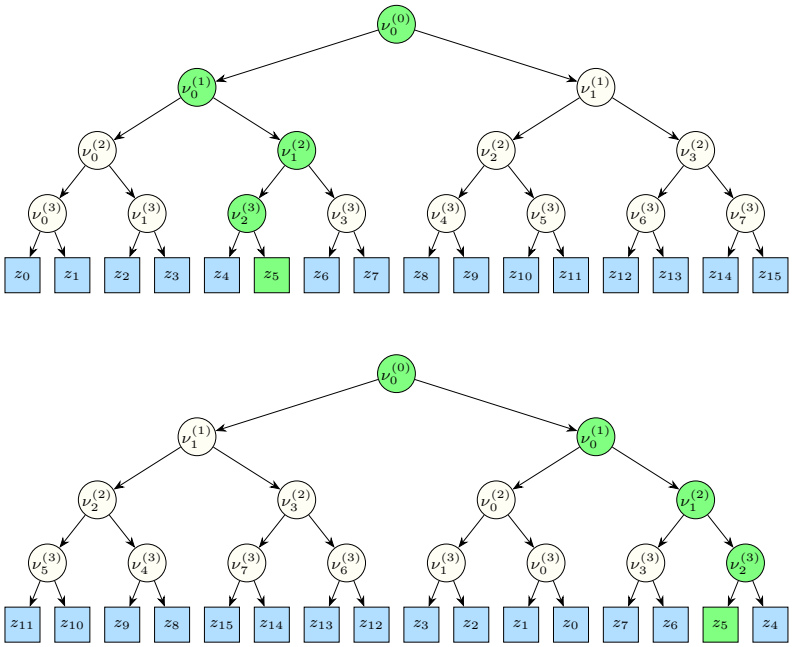

Fig. 5. Public vs. private evaluation of a decision tree.

Proof. From $m'_\ell = M'_\ell - \lfloor M'_\ell/2^t \rfloor 2^t$ and $m_\ell = \mu_\ell - \lfloor \mu_\ell/2^t \rfloor 2^t$, we can write

$$\delta_\ell \oplus \delta'_\ell = \mathbb{1}\{m'_\ell \leqslant m_\ell\} = \left\lfloor \tfrac{m_\ell + 2^t - m'_\ell}{2^t} \right\rfloor = \left\lfloor \tfrac{\mu_\ell + 2^t - M'_\ell}{2^t} \right\rfloor - \left\lfloor \tfrac{\mu_\ell}{2^t} \right\rfloor - \left\lfloor \tfrac{M'_\ell}{2^t} \right\rfloor \ .$$

Furthermore, defining $s = \sigma^{(\ell)} \oplus j$, we have $M'_\ell = M_j^{(\ell)} = x_{i_s^{(\ell)}} - T_s^{(\ell)} + \mu_\ell$. Hence, we get

$$\left\lfloor \tfrac{\mu_\ell + 2^t - M'_\ell}{2^t} \right\rfloor = \left\lfloor \frac{T_s^{(\ell)} - x_{i_s^{(\ell)}} + 2^t}{2^t} \right\rfloor = \mathbb{1}\left\{ x_{i_s^{(\ell)}} \leqslant T_s^{(\ell)} \right\} \ .$$

Putting the two relations together, modulo 2, yields

$$\mathbb{1}\left\{ x_{i_s^{(\ell)}} \leqslant T_s^{(\ell)} \right\} \equiv \delta_\ell + \delta'_\ell + \left\lfloor \tfrac{\mu_\ell}{2^t} \right\rfloor + \left\lfloor \tfrac{M'_\ell}{2^t} \right\rfloor \pmod 2 \ .$$

This concludes the proof by noting that $s = k^\star$, $b_\ell = \delta_\ell + \lfloor \mu_\ell/2^t \rfloor \pmod 2$, and $b'_\ell \equiv \delta'_\ell + \lfloor M'_\ell/2^t \rfloor \pmod 2$. □

As a result, the client learns the classification result $\mathfrak{T}(\boldsymbol{x})$ at the end of the protocol in Fig. 6.

5 Discussion

5.1 Security considerations

The decision tree evaluation protocol presented in Fig. 6 is secure in the *semi-honest model*, a.k.a. *honest-but-curious model*. It assumes two semantically secure

Fig. 6. Secure decision tree evaluation protocol.

1. The client encrypts the entries of the feature vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ and sends $[\![x_i]\!]$, for $1 \leqslant i \leqslant n$, to the server.
2. For $\ell = 0, \ldots, d-1$:
   (a) The server chooses a random $(t+\kappa)$-bit mask μ_ℓ. It also defines $m_\ell = \mu_\ell \bmod 2^t$ and

$$\sigma^{(\ell)} = \begin{cases} 0 & \text{if } \ell = 0 \\ (b_0, \ldots, b_{\ell-1})_2 & \text{otherwise} \end{cases} .$$

   For $k = 0, \ldots, 2^\ell - 1$:
      i. The server sets $s \leftarrow k \oplus \sigma^{(\ell)}$.
      ii. The server computes[a] $[\![M_k^{(\ell)}]\!] = [\![x_{i_s^{(\ell)}} - T_s^{(\ell)} + \mu_\ell]\!]$.

   (b) The client sets $j \leftarrow (b'_0, \ldots, b'_{\ell-1})_2$ and engages in a 1-out-of-2^ℓ OT with the server to obtain the value of $[\![M_j^{(\ell)}]\!]$.
   (c) The client decrypts $[\![M_j^{(\ell)}]\!]$, gets $M_j^{(\ell)}$, and defines $M'_\ell = M_j^{(\ell)}$ and $m'_\ell = M_j^{(\ell)} \bmod 2^t$.
   (d) Client and server run the basic comparison protocol (Fig. 1) on inputs m'_ℓ and m_ℓ. At the end of the protocol, they respectively obtain a bit δ'_ℓ and δ_ℓ such that $\delta_\ell \oplus \delta'_\ell = \mathbb{1}\{m'_\ell \leqslant m_\ell\}$. The server sets $b_\ell = \delta_\ell$ if $\lfloor \mu_\ell / 2^t \rfloor$ is even, and $b_\ell = \delta_\ell \oplus 1$ otherwise. The client sets $b'_\ell = \delta'_\ell$ if $\lfloor M'_\ell / 2^t \rfloor$ is even, and $b'_\ell = \delta'_\ell \oplus 1$ otherwise.
3. For $k = 0, \ldots, 2^d - 1$, the server sets $z^*_k \leftarrow z_{k \oplus \sigma^{(d)}}$ where $\sigma^{(d)} = (b_0, \ldots, b_{d-1})_2$. The client sets $j \leftarrow (b'_0, \ldots, b'_{d-1})_2$ and engages in a 1-out-of-2^d OT with the server to obtain the value of z^*_j.

[a] For dummy nodes, the server can instead draw a random ciphertext for $[\![M_k^{(\ell)}]\!]$.

additively homomorphic encryption schemes, $[\![\cdot]\!]$ and $[\![\cdot]\!]$, and a semi-honest secure 1-out-of-N OT protocol. Informally, if the parties interact according to the protocol specification, the semi-honest model guarantees that (i) the client only learns the classification result and a bound d on the depth of the decision tree, and (ii) the server learns nothing.

The security is defined via the ideal-world/real-world simulation paradigm; see *e.g.* the excellent tutorial provided in [19, Chapter 6]. The security proof of our main construction is standard. We refer the reader to [34, Sect. 2.3] for precise security definitions and to [34, Theorem 3.2] for the proof technique.

Selecting parameter κ. Step 2a in Fig. 6 requires a random mask μ_ℓ to blind the value of $x_{i^{(\ell)}_{k \oplus \sigma^{(\ell)}}} - T^{(\ell)}_{k \oplus \sigma^{(\ell)}}$ in

$$M_k^{(\ell)} = x_{i^{(\ell)}_{k \oplus \sigma^{(\ell)}}} - T^{(\ell)}_{k \oplus \sigma^{(\ell)}} + \mu_\ell \ ,$$

for $k = 0, \ldots, 2^\ell - 1$. In Step 2c, as the output of the 1-out-2^ℓ OT, the client obtains $M'_\ell = M_j^{(\ell)}$ for a single value $j \in [0, \ldots, 2^\ell - 1]$. This justifies that the same mask μ_ℓ can be re-used at level ℓ, for each successive value of k.

Moreover, when the message space $\mathcal{M}$ for $[\![\cdot]\!]$ is much larger than the set $\{0,1\}^t$, since $x_{i^{(\ell)}_{k\oplus\sigma^{(\ell)}}}$ and $T^{(\ell)}_{k\oplus\sigma^{(\ell)}}$ are t-bit values, there is no need to draw the mask μ_ℓ in the whole range of $\mathcal{M}$. Any $(t+\kappa)$-bit value for a relatively short security length κ will generate a mask that *statistically* hides $x_{i^{(\ell)}_{k\oplus\sigma^{(\ell)}}} - T^{(\ell)}_{k\oplus\sigma^{(\ell)}}$.

When the message space $\mathcal{M} = \mathbb{Z}_{2^t}$ (like in [6]), μ_ℓ and M'_ℓ are defined modulo 2^t as elements in $\mathbb{Z}_{2^t}$ (and thus $\lfloor \mu_\ell / 2^t \rfloor = \lfloor M'_\ell / 2^t \rfloor = 0$). Parameter κ will be in this case set to 0.

5.2 Performance analysis

We compare the proposed evaluation protocol with the two most recent protocols, Wu *et al.* (PETS 2016) and Tai *et al.* (ESORICS 2017), *in the semi-honest setting.*

Let $\mathfrak{T}$ be a binary decision tree of depth d with m [non-dummy] internal nodes (*i.e.*, decision nodes). Let also n be the number of entries in the feature vector; each entry being represented as a t-bit integer. Both the computation and bandwidth are analyzed. For the computation complexity, we count the number of public-key operations performed by each of the parties. For the bandwidth, we count the number of ciphertexts sent by one party to the other.

Table 1. Comparison.

	Bandwidth		Computation	
	Client	Server	Client	Server
Wu *et al.* [34]	$nt + m + 1$	$mt + m + 2^d$	$O((n+m)t + d)$	$O(mt + 2^d)$
Tai *et al.* [31]	$nt + m$	$mt + 2(m+1)$	$O((n+m)t)$	$O(mt)$
Ours (Fig. 6)	$d(\lfloor t/2 \rfloor + 2) + n$	$dt + 2^{d+1} - 1$	$O(n + dt)$	$O(m + dt)$

In our protocol, the client first encrypts the feature vector. This requires $O(n)$ public-key operations and produces n ciphertexts. Next, in the main loop, at each level ℓ (for $0 \leqslant \ell \leqslant d-1$), the client mainly performs two steps with the server: (1) one 1-out-of-2^ℓ OT where the client is the chooser and (2) one comparison of two t-bit integers. We assume that the OT is implemented with the Naor-Pinkas protocol (see Appendix A) and that the comparison makes use of our comparison protocol (Fig. 1). So, on the client's side, the OT requires in total for the main loop $O(d)$ public-key operations and $d-1$ ciphertexts; the comparison requires in total $O(dt)$ public-key operations and $d(\lfloor t/2 \rfloor + 1)$ ciphertexts. The last step is a 1-out-of-2^d OT, which requires $O(1)$ public-key operation and one ciphertext. Summing up, the total complexity for the client amounts to $O(n + dt)$ with $n + d - 1 + d(\lfloor t/2 \rfloor + 1) + 1 = d(\lfloor t/2 \rfloor + 2) + n$ ciphertexts. On the server's side,

the server processes in addition m encryptions (Step 2a in Fig. 6) to form $[\![M_k^{(\ell)}]\!]$ for the non-dummy nodes. This corresponds to a complexity of $O(m)$. The OT in the main loop requires in total $O(d)$ public-key operations and $2^d - 1$ ciphertexts. The comparison requires $O(dt)$ public-key operations and dt ciphertexts (we suppose here that the server plays the role of Party B; *cf.* Fig. 1). The last step for the final OT incurs for the server $O(1)$ public-key operation and 2^d ciphertexts. Consequently, the total complexity for the server is of $O(m + dt)$ and the needed bandwidth is of $2^d - 1 + dt + 2^d = dt + 2^{d+1} - 1$ ciphertexts.

A typical value for the precision is $t = 64$. As shown in Table 1, since $d \ll m$, the proposed protocol greatly reduces the workload on both the client's and server's sides. The bandwidth usage is also improved on the client's side with our protocol. On the server's side, the savings depend on the tree sparsity. Denser decision trees give rise to more savings; for a complete binary tree (*i.e.*, $m = 2^d - 1$), our protocol saves $(2^d - d)t$ ciphertexts on the server's side.

5.3 Random forests

As in [34], our main construction extend to the evaluation of *random forests*. Introduced by Ho [15], the random forest improves the quality of the classification task by combining the results of a multitude of decision trees. A random forest $\mathfrak{F}$ can be defined as an ensemble of decision trees, $\mathfrak{F} = \{\mathfrak{T}_i\}_i$. Its output is computed by taking the majority vote; *i.e.*, $\mathfrak{F}(\boldsymbol{x}) = \mathrm{maj}\{\mathfrak{T}_i(\boldsymbol{x})\}_i$.

6 Conclusion and Future Work

In this work, we introduced an enhanced comparison protocol and several extensions thereof. As an application, combined with a novel design strategy and a number of optimizations, we developed an efficient protocol for the private evaluation of decision trees.

Future work. An interesting direction for future work is to design a privacy-preserving evaluation protocol in the multi-user setting, wherein the feature vector and/or the model are shared among multiple entities. Another interesting direction is to extend the protocol to make it secure against malicious adversaries.

References

1. Abu-Nimeh, S., Nappa, D., Wang, X., Nair, S.: A comparison of machine learning techniques for phishing detection. In: 2nd Annual eCrime Researchers Summit. pp. 60–69. ACM (2007), doi:10.1145/1299015.1299021
2. Agrawal, R., Shrikant, R.: Privacy-preserving data mining. ACM SIGMOD Record 29(2), 439–450 (Jun 2000), doi:10.1145/335191.335438
3. Asharov, G., Lindell, Y., Schneider, T., Zohner, M.: More efficient oblivious transfer extensions. J. Cryptol. 30(3), 805–858 (2017), doi:10.1007/s00145-016-9236-6

4. Azar, A.T., El-Metwally, S.M.: Decision tree classifiers for automated medical diagnosis. Neural Computing & Applications 23(7-8), 2387–2403 (2013), doi:10.1007/s00521-012-1196-7
5. Barni, M., Failla, P., Kolesnikov, V., Lazzeretti, R., Sadeghi, A.R., Schneider, T.: Secure evaluation of private linear branching programs with medical applications. In: Computer Security – ESORICS 2009. LNCS, vol. 5789, pp. 424–439. Springer (2009), doi:10.1007/978-3-642-04444-1_26
6. Benhamouda, F., Herranz, J., Joye, M., Libert, B.: Efficient cryptosystems from 2^k-th power residue symbols. J. Cryptol. 30(2), 519–549 (2017), doi:10.1007/s00145-016-9229-5
7. Bost, R., Popa, R.A., Tu, S., Goldwasser, S.: Machine learning classification over encrypted data. In: 22nd Annual Network and Distributed System Security Symposium (NDSS 2015). The Internet Society (2015), doi:10.14722/ndss.2015.23241
8. Bunn, P., Ostrovsky, R.: Secure two-party k-means clustering. In: 14th ACM Conference on Computer and Communications Security (CCS 2007). pp. 486–497. ACM (2007), doi:10.1145/1315245.1315306
9. Damgård, I., Geisler, M., Krøigaard, M.: Efficient and secure comparison for on-line auctions. In: Information Security and Privacy (ACISP 2007). LNCS, vol. 4586, pp. 416–430. Springer (2007), doi:10.1007/978-3-540-73458-1_30
10. Damgård, I., Geisler, M., Krøigaard, M.: Homomorphic encryption and secure comparison. Int. J. Appl. Cryptography 1(1), 22–31 (2008), doi:10.1504/IJACT.2008.017048
11. Damgård, I., Geisler, M., Krøigaard, M.: A correction to 'Efficient and secure comparison for on-line auctions'. Int. J. Appl. Cryptography 1(4), 323–324 (2009), doi:10.1504/IJACT.2009.028031
12. Du, W., Zhan, Z.: Building decision tree classifier on private data. In: IEEE Workshop on Privacy, Security, and Data Mining. Conferences in Research and Practice in Information Technology, vol. 14. Australian Computer Society (2002), http://crpit.com/confpapers/CRPITV14Du.pdf
13. Erkin, Z., Franz, M., Guajardo, J., Katzenbeisser, S., Lagendijk, I., Toft, T.: Privacy-preserving face recognition. In: Privacy Enhancing Technologies (PETS 2009). LNCS, vol. 5672, pp. 235–253. Springer (2009), doi:10.1007/978-3-642-03168-7_14
14. Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. Commun. ACM 28(6), 637–647 (1985), doi:10.1145/3812.3818
15. Ho, T.K.: The random subspace method for constructing decision forests. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(8), 832–844 (1998), doi:10.1109/34.709601
16. Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Advances in Cryptology – CRYPTO '96. LNCS, vol. 1109, pp. 104–113. Springer (1996), doi:10.1007/3-540-68697-5_9
17. Libbrecht, M.W., Noble, W.S.: Machine learning applications in genetics and genomics. Nature Reviews Genetics 16(6), 321–332 (2015), doi:10.1038/nrg3920
18. Lin, H.Y., Tzeng, W.G.: An efficient solution to the millionaires' problem based on homomorphic encryption. In: Applied Cryptography and Network Security (ACNS 2005). LNCS, vol. 3531, pp. 456–466. Springer (2005), doi:10.1007/11496137_31
19. Lindell, Y. (ed.): Tutorials on the Foundations of Cryptography. Information Security and Cryptography, Springer (2017), doi:10.1007/978-3-319-57048-8
20. Lindell, Y., Pinkas, B.: Privacy preserving data mining. J. Cryptol. 15(3), 177–206 (2002), doi:10.1007/s00145-001-0019-2

21. Liu, C., Wechsler, H.: Gabor feature based classification using the enhanced Fisher linear discriminant model for face recognition. IEEE Trans. Image Processing 11(4), 467–476 (2002), doi:10.1109/TIP.2002.999679
22. Min, J.H., Lee, Y.C.: Bankruptcy prediction using support vector machine with optimal choice of kernel function parameters. Expert Systems with Applications 28(4), 603–614 (2005), doi:10.1016/j.eswa.2004.12.008
23. Mohassel, P., Zhang, Y.: SecureML: A system for scalable privacy-preserving machine learning. In: 2017 IEEE Symposium on Security and Privacy. pp. 19–38. IEEE (2017), doi:10.1109/SP.2017.12
24. Naor, M., Pinkas, B.: Efficient oblivious transfer protocols. In: 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2001). pp. 448–457. ACM/SIAM (2001), https://dl.acm.org/citation.cfm?id=365411.365502
25. Nasrabadi, N.M.: Pattern recognition and machine learning. J. Electronic Imaging 16(4), 049901 (2007), doi:10.1117/1.2819119
26. Nikolaenko, V., Ioannidis, S., Weinsberg, U., Joye, M., Taft, N., Boneh, D.: Privacy-preserving matrix factorization. In: 20th ACM Conference on Computer and Communications Security (CCS 2013). pp. 801–812. ACM (2013), doi:10.1145/2508859.2516751
27. Nikolaenko, V., Weinsberg, U., Ioannidis, S., Joye, M., Boneh, D., Taft, N.: Privacy-preserving ridge regression on hundreds of millions of records. In: 2013 IEEE Symposium on Security and Privacy. pp. 334–348. IEEE (2013), doi:10.1109/SP.2013.30
28. Rabin, M.O.: How to exchange secrets by oblivious transfer. Tech. Rep. TR-81, Harvard University (1981), https://ia.cr/2005/187
29. Resnick, P., Varian, H.R.: Recommender systems. Commun. ACM 40(3), 56–58 (1997), doi:10.1145/245108.245121
30. Rivest, R.L., Adleman, L., Dertouzous, M.L.: On data banks and privacy homomorphisms. In: Foundations of Secure Computation, pp. 169–179. Academic Press (1978), https://people.csail.mit.edu/rivest/RivestAdlemanDertouzos-OnDataBanksAndPrivacyHomomorphisms.pdf
31. Tai, R.K.H., Ma, J.P.K., Zhao, Y., Chow, S.S.M.: Privacy-preserving decision trees evaluation via linear functions. In: Computer Security – ESORICS 2017, Part II. LNCS, vol. 10493, pp. 494–512. Springer (2017), doi:10.1007/978-3-319-66399-9
32. Vaidya, J., Yu, H., Jiang, X.: Privacy-preserving SVM classification. Knowledge and Information Systems 14(2), 161–178 (2008), doi:10.1007/s10115-007-0073-7
33. Veugen, T.: Improving the DGK comparison protocol. In: 2012 IEEE International Workshop on Information Forensics and Security (WIFS 2012). pp. 49–54. IEEE (2012), doi:10.1109/WIFS.2012.6412624
34. Wu, D.J., Feng, T., Naehrig, M., Lauter, K.: Privately evaluating decision trees and random forests. Proceedings on Privacy Enhancing Technologies 2016(4), 335–355 (2016), doi:10.1515/popets-2016-0043
35. Yap, B.W., Ong, S.H., Husain, N.H.M.: Using data mining to improve assessment of credit worthiness via credit scoring models. Expert Systems with Applications 38(10), 13274–13283 (2011), doi:10.1016/j.eswa.2011.04.147

A Naor-Pinkas OT Protocol

Let $\mathbb{G} = \langle g \rangle$ be a group of order q, in which the Diffie-Hellman assumption holds. Let also a cryptographic hash function H mapping to $\{0,1\}^t$, modeled as a random

oracle. The sender selects at random $K_1, \ldots, K_{N-1} \in \mathbb{G}$ and computes $y = g^x$ for some random integer $x \in \mathbb{Z}_q$. The sender's public key is $(g, y, K_1, \ldots, K_{N-1})$ and the secret key is x. The sender pre-computes $S_i = (K_i)^x$ for $1 \leqslant i \leqslant N-1$.

The sender's input is a set of N bit-strings $\sigma_0, \ldots, \sigma_{N-1} \in \{0,1\}^t$. Suppose a chooser (Carol) wishes to get string σ_j for some $j \in \{0, \ldots, N-1\}$. The amortized 1-out-of-N Naor-Pinkas OT protocol [24, § 3.1] proceeds as follows.

1. The chooser draws a random integer $r \in \mathbb{Z}_q$ and computes $pk_j = g^r$. If $j \neq 0$, she sets $pk_0 = K_j / pk_j$. She sends pk_0 to the sender.
2. The sender computes $(pk_0)^x$ and then, for $1 \leqslant i \leqslant N-1$, sets $(pk_i)^x$ as $(pk_i)^x = S_i / (pk_0)^x$. Next, he chooses a nonce R and encrypts each string σ_i as $c_i = H\big((pk_i)^x, R, i\big) \oplus \sigma_i$, for $0 \leqslant i \leqslant N-1$. He sends $(c_0, \ldots, c_{N-1}, R)$ to the chooser.
3. The chooser recovers σ_j as $c_j \oplus H(y^r, R, j)$.

Interestingly, the protocol can be re-used multiple times. After the one-time initialization phase, each transfer only costs a single exponentiation (in $\mathbb{G}$) plus $N-1$ multiplications for the sender. The chooser essentially computes one exponentiation per transfer.