Enhanced Outsourced and Secure Inference for Tall Sparse Decision Trees

Andrew Quijano
Dept. of Computer Science and Engineering
New York University
New York, NY, USA
andrew.quijano@nyu.edu

Spyros T. Halkidis
Dept. of Applied Informatics
University of Macedonia
Thessaloniki, Greece
halkidis@uom.edu.gr

Kevin Gallagher
NOVA LINCS
NOVA School of Science and Technology
Lisbon, Portugal
k.gallagher@fct.unl.pt

Kemal Akkaya
Advanced Wireless and Security Lab
Florida International University
Miami, FL, USA
kakkaya@fiu.edu

Nikolaos Samaras
Dept. of Applied Informatics
University of Macedonia
Thessaloniki, Greece
samaras@uom.edu.gr

Abstract—A decision tree is an easy-to-understand tool that has been widely used for classification tasks. On the one hand, due to privacy concerns, there has been an urgent need to create privacy-preserving classifiers that conceal the user's input from the classifier. On the other hand, with the rise of cloud computing, data owners are keen to reduce risk by outsourcing their model, but want security guarantees that third parties cannot steal their decision tree model. To address these issues, Joye and Salehi introduced a theoretical protocol that efficiently evaluates decision trees while maintaining privacy by leveraging their comparison protocol that is resistant to timing attacks. However, their approach was not only inefficient but also prone to side-channel attacks. Therefore, in this paper, we propose a new decision tree inference protocol in which the model is shared and evaluated among multiple entities. We partition our decision tree model by each level to be stored in a new entity we refer to as a "level-site." Utilizing this approach, we were able to gain improved average run time for classifier evaluation for a non-complete tree, while also having strong mitigations against side-channel attacks.

Index Terms—Decision Trees, Encrypted Integer Comparison, Privacy, Data Mining

I. Introduction

Machine learning is widely used in various real-world applications, such as managing student data [30], analyzing health data [16][19], and fraud detection in energy consumption [5]. One type of classifier commonly used in machine learning is the Decision Tree (DT). A DT is a classifier that consists of decision nodes and leaves corresponding to an output class. DT evaluations have the advantage of being a fast and computationally inexpensive solution compared to other ML techniques. Also, a DT model is easy to audit as decisions are made through tree traversal.

However, as machine learning techniques begin to analyze sensitive data, such as medical records, privacy concerns arise.

Privacy is becoming an important issue as there are regulations such as the Health Insurance Portability and Accountability Act (HIPAA) in the United States [11] and the General Data Protection Regulation (GDPR) in the EU [22] that protect sensitive user data. One potential approach to balance both the desire to utilize machine learning and maintain a user's privacy is to use differential privacy. However, given that differential privacy relies on the injection of noise into the data set, the resulting classifier loses accuracy [18]. Instead, we propose using a *privacy-preserving decision tree (PPDT)* to achieve both accuracy and privacy preservation. A PPDT ensures that the server(s) storing the DT would not have information about the model leaked to the client, and the server is unaware of the client's input or output of the DT model.

Previous work by Joye and Salehi [13] designed a theoretical PPDT classification protocol that used an encrypted integer comparison protocol based on Veugen's [28][29] protocol that has additional protection against timing attacks. To this end, they converted a DT model into a complete binary tree, where all classes are in the bottom level of the tree. This ensures that every evaluation will have the same number of encrypted integer comparisons, so using a timing attack to extract information about the DT is still infeasible.

However, the PPDT model proposed in [13] has some issues. First, there is a performance loss since the full depth of the PPDT has to be traversed even if the classification could be computed earlier. In the case of a sparse DT [12][15], which is defined as a DT that has few internal nodes, this implies that most leaves would not be at the maximum depth of the tree; therefore, Joye and Salehi's PPDT evaluation protocol would be inefficient. Also, they used an asymmetric based comparison based on RSA, which could be susceptible to power-based side channel attacks [3]. Finally, since the classification occurs on one server, this server is a single point of failure. So, to address these concerns, we propose to split the PPDT model between multiple entities. We call these entities

level-sites, inspired by the fact that the protocol progresses level by level, ending as soon as the client reaches a leaf.

In this paper, we extend the theoretical PPDT classification protocol in [13] to a scenario where the DT model is shared between the level-sites, which improves our PPDT average run-time performance (especially for sparse, tall trees) while also providing robust security protections against side-channel attacks. To classify client data with our PPDT, a client would connect to the level-site 0, and provide an encrypted feature vector to classify. Using [13] comparison protocol for numerical data or a timing attack resistant [20], the level-site 0 calculates the index in scope for the next level in the PPDT and sends the index to the level-site 1, etc. This process repeats until a level-site index is on a leaf and returns the classification to the client.

Assuming that the user's classification will not always result in traversing the whole tree, we observed a faster evaluation time in our experiments, as we have fewer encrypted comparisons to compute. Each level-site takes less than 1 second to complete the encrypted integer comparison and passing the index to the next level-site. We also show that our approach is resistant to side channel attacks and to single point of failures through our decentralized level-site approach.

II. RELATED WORK

There have been several methods proposed for implementing PPDTs in the literature [27], [4]. Although some of these methods focus on protecting the confidentiality of training data [17], [9], other related works, such as [4] and [26], focus on a secure inference protocol. A secure inference protocol enables users and model owners to interact so that the user obtains the prediction result while ensuring that neither party learns any other information about the user input or the model.

Liu et al. [18] designate Data Owners (DO) as the users who possess the DT training data, Cloud Service User (CSU) as a user with data to evaluate in the PPDT, and both Cloud Service Provider (CSP) and Evaluation Service Provider (ESP) as distinct cloud providers responsible for training and evaluating data, respectively. It is worth noting that both the CSP and the ESP must have knowledge of all the possible labels and attributes for the DT. They utilize the Distributed Two Trapdoors Public Key Cryptosystem (DT-PKC), which is similar to Paillier [21], with the distinction that it allows comparisons between ciphertexts encrypted by different public keys. Liu et al. [18] created new protocols to securely count the frequency of attributes in the data set, and modified Veugen's algorithm to be used for DT-PKC [28][29]. Our paper has the advantage of using 2048-bit keys, which are more secure than Liu et al. [18] and our PPDT has faster PPDT evaluation times.

Yuan et al. [32] utilize gradient-boosting decision trees (GBDT) to train DTs. They introduce the privacy-preserving distributed GBDT (PPD-GBDT), which uses differential privacy, polynomial approximation, and fully homomorphic encryption to achieve comprehensive privacy protection for their DT model. The DOs add noise to their local GBDT, and then convert the model into a polynomial format that is sent to the CSP. For evaluation, the client encrypts their input and sends the ciphertext and public key to the CSP. The CSP would return encrypted results, and the client would decrypt the classification. Our PPDT has generally faster PPDT evaluation times and no noise is added, which could potentially cause classification inaccuracies.

III. PRELIMINARIES

A. Background on Homomorphic Encryption

An Additively Homomorphic Encryption scheme [25] such as Paillier [21] or DGK [6][7][8] allows a user to combine two ciphertexts to receive a ciphertext output that has the sum of both input ciphertexts. For example, assume that there are two messages α, α' in the plaintext space $\mathcal{M}$. We write an encryption of α as $[\![\alpha]\!]$ and use the 'boxplus' operator ($\boxplus$) to denote the addition of two ciphertexts. Then we have that, after decryption, $[\![(\alpha + \alpha')]\!] = [\![\alpha]\!] \boxplus [\![\alpha']\!]$.

We use encryption to securely compare two encrypted integers. Assume that there are two integers held by A and B. Suppose that A has the t-bit integer $x = \sum_{i=0}^{t-1} x_i 2^i$ in binary form, while party B has another t-bit integer $y = \sum_{i=0}^{t-1} y_i 2^i$. The goal is for A and B, respectively, to obtain the protocol bits δ_A and δ_B, such that $\delta_A \oplus \delta_B = \mathbb{1}\{x \leq y\}$.

The secure integer comparison problem was first stated in [31], where two millionaires would like to find out who is richer without revealing the amount of their wealth. Various more advanced solutions followed in [6][7][8]. The high communication cost of these protocols was addressed by Veugen [28] and Joye and Salehi [13] improved Veugen's protocol to be resistant to timing attacks.

B. System Model

We assume the following entities. A **Client** has a feature vector x and wants to classify it using the DT. The client creates public and private Paillier [21] and DGK [6][7][8] keys. The client will give the server the public keys. *In a real-world scenario, a client could be a doctor who needs to know if a patient has thyroid disease*. A **Server** has access to the training data and creates the DT. The server creates the PPDT to send to the level-sites. To save network bandwidth, the server will give the level-sites the public keys. *In a real-world scenario, this could be a laboratory that can inform a doctor if there is a probable sign of thyroid disease*. A **Level-site** only has a level of the PPDT and the public keys used to compare integers with the client. *In a real-world scenario, a level-site would be a cloud provider hosting the PPDT*.

C. Threat Model

We assume the server is trusted, but the client and level-sites are "honest-but-curious" (HBC). The client may attempt to learn the decision nodes of a DT, which would contain the feature and a corresponding threshold. We assume that the level-sites will attempt to learn the client's feature vector data and classification output.

We expect the level-sites to strictly follow the protocol of our designed PPDT. However, we will assume that the level-sites would be curious to analyze both power consumption and computation times for potential side-channel attacks [3][2]. We also consider a passive adversary, $\mathcal{A}$, outside the system in our model. $\mathcal{A}$'s goal is to learn the client input, the classification from the PPDT model, and the PPDT model's thresholds.

IV. PPDT Approach

A. Goals and Overview

We are motivated by the fact that in a tall and sparse decision tree, most of the leaves in a DT are not at the maximum depth of the tree. Indeed, we computed the level of each classification from the training dataset [1] as shown in Table I. We want to minimize the encrypted comparisons used to improve our performance for our PPDT in this circumstance. In addition, if a level-site l fails, we would like the traffic re-directed to a backup, solving the single point of failure in Joye and Salehi's approach [13].

TABLE I
Analysis of level of classifications of Training data

Dataset	Average	Median	3rd Quartile	Max	Size
Hypothyroid	2.78	2	2	9	3372
Spambase	9.60	9	11	22	4601
Nursery	5.99	7	8	14	12960

Based on these goals, we set up our PPDT as follows: **(i)** The server builds the DT with the training data; **(ii)** The client sends to the server the DGK [6][7][8] and Paillier [21] public key. The server provides the client with both the classes of the DT; and **(iii)** For each level $0, 1, \cdots, d-1$, where d is the depth of the DT, the server sends the **encrypted** thresholds and classifications to the respective level-site.

B. PPDT Inference

In our setting, there is one client and d level-sites, where d is equivalent to the depth of the DT. The client has a private feature vector, $\boldsymbol{x} = (x_1, x_2, \cdots, x_n) \in \mathbb{Z}^n$, which is encrypted using homomorphic encryption to generate $[\![x_i]\!]$, where $i \in 1, \cdots, n$. We also assume that all thresholds are t-bits long. Each level site uses the Joye and Salehi encrypted integer comparison protocol [13] or a timing attack resistant encrypted equality protocol [20] for categorical data. We note that in our PPDT, *we improve our PPDT performance by not using oblivious transfer or converting to a complete binary tree as in the original work* [13], as shown in Figure 1.

C. Level-site Evaluation

We evaluated the PPDT by traversing down level-by-level, where level-site l sends to level-site $l+1$ the encrypted index i of the previous level-site. The client encrypts $\boldsymbol{x}$ and sends it to level-site 0.

Consider l to be the level of the tree we are currently considering and k to be the index of the current node at that level. We say M_k^l is the blinded threshold comparison value

[1] https://github.com/renatopp/arff-datasets

at level l at position k and $[\![M_k^l]\!]$ to be the encrypted blinded threshold comparison value.

We have to compute at each level-site l the quantity $[\![M_k^l]\!] = [\![x_i - T_k^l + \mu_l]\!]$. All these calculations are performed without decryption by using additively homomorphic encryption since: $[\![M_k^l]\!] = [\![x_i - T_k^l + \mu_l]\!] = [\![x_i]\!] \boxplus [\![-T_k^l]\!] \boxplus [\![\mu_l]\!]$. We note that μ_l is a random $(t+\kappa)$ bit mask computed at the level-site l, where κ is a security parameter [13]. In addition, $[\![x_i]\!]$ is the value of an attribute of $\boldsymbol{x}$ matched with the attribute of T_k^l.

Both the client and the level site l utilize [20] to compare categorical data or [13] to compare numerical data. We used a label encoder so that we can use an encrypted equality check to traverse a decision tree. The encrypted comparison protocols require M_k^l and μ_l as inputs. At the end of the protocol, the level-site l has δ_l and the client has δ_l', such that:

$$\delta_l \oplus \delta_l' = \begin{cases} Numerical & \mathbb{1}\{T_k^l \le x_i\} \\ Categorical & \mathbb{1}\{T_k^l == x_i\} \end{cases}$$

Without the other corresponding bit, neither the level-site l nor the client knows the value of the inequality. To get around this, the level-site l can additively blind δ_l with r' and send it to the client to compute $(\delta_l + r') \oplus \delta_l'$. This output is returned to level-site l, as level-site l knows r', it can determine the inequality output without revealing δ_l to the client. The output of the inequality informs how to update the index k for the level-site $l+1$.

If the node at index k at the level-site l is a leaf, level-site l returns the encrypted classification to the client. We summarize our approach in Algorithm 1.

Algorithm 1 PPDT Secure Inference Protocol with level-sites

1: next-index=0
2: Client sends **x** and next-index to level-site 0
3: **for** *int l = 0; l < d; l++* **do**
4:     level-site l receives indexes and **x**
5:     **for** *int k = 0; k < l.NodeCount(); k++* **do**
6:         **if** *Not in-scope based on next-index* **then**
7:             **continue**
8:         **else if** in-scope and node i is a leaf **then**
9:             **return encrypted-classification** to client
10:         **else**
11:             secure comparison w/ client and level-site l
12:             next-index $\leftarrow$ set index for next level-site
13:             level-site l sends index and **x** to level-site $l+1$
14:             **break**
15:         **end if**
16:     **end for**
17: **end for**

V. Security Analysis

We construct security proofs via simulation, reducing our protocol to multi-party integer comparisons and homomorphic encryption, and citing original proofs. Assuming an honest-but-curious (HBC) adversary, we find that it can gather information on the PPDT through numerous queries. We then
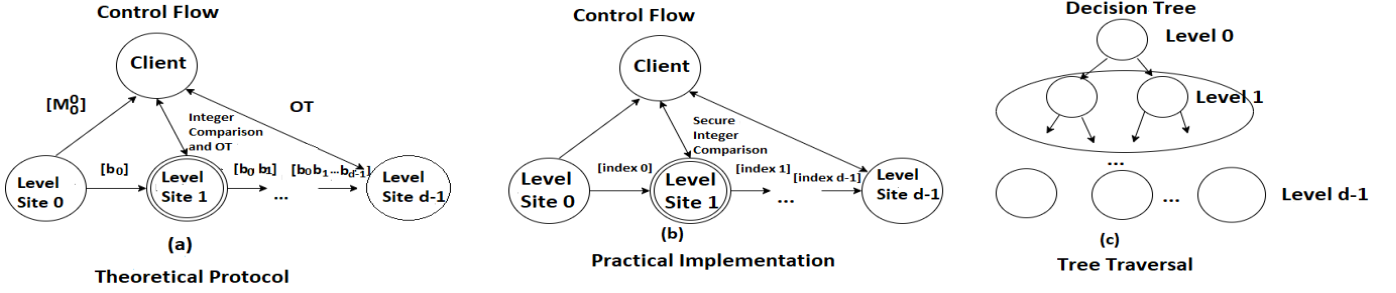
Fig. 1. (a) Direct conversion of the Joye and Salehi PPDT [13] to the level-site approach. (b) Control flow in our PPDT implementation, and (c) Tree Traversal. We emphasize Level 1 of the DT evaluation.

discuss countermeasures. We conclude by demonstrating how our PPDT is safeguarded against side-channel attacks and can be upgraded for post-quantum security.

Our proofs use simulation. We consider a PPDT classification protocol, $f(x, y)$, which we break into a two-party protocol such that $f(x, y) = f_1(x, y), f_2(x, y)$, and consider our implementation, π, of the functionality. During the execution of π, each of the two parties will generate a $view_i^\pi, \forall i \in \{1, 2\}$. We claim our system is secure if we can show that there exist two simulators, $\mathcal{S}_1(1^n, x, f_1(x, y))$ and $\mathcal{S}_2(1^n, y, f_2(x.y))$ that are computationally equivalent to $view_1^\pi(x, y, n)$ and $view_2^\pi(x, y, n)$ where n is a security parameter.

A. Security against an HBC client

We assume a HBC client who follows the protocol faithfully but wishes to learn information about the tree node. To demonstrate that this client does not learn anything more than its view, we generate a simulator $\mathcal{S}_1$ that takes the input of the client and simulates the output of a level-site.

If the level-site l is equal to zero, the simulator must sample a value $M_0^0 \xleftarrow{R} \mathbb{Z}_p^*$, and encrypt it to achieve $[\![M_0^0]\!]$. This number is computationally indistinguishable from any real input that could be sent by a level-site given the properties of additive homomorphic encryption as demonstrated in [25]. The client will then decrypt this number, resulting in M_0^0, and define $M_0' = M_0^0$ and $m_0' = M_0^0 \mod 2^t$. To simulate the integer comparison protocol, it will then sample $b_r \xleftarrow{R} \{0, 1\}$. This value will take the place of b_0^2. As a result of the integer comparison protocol, each output in this case is equally likely to meet the standard of computational equivalence.

In the case that the level-site is greater than 0, we need to perform more steps. First, the simulator sets $j \leftarrow (b_0', \cdots, b_{l-1}')_2$, according to the protocol. As before, the simulator samples a value $M_k^l \xleftarrow{R} \mathbb{Z}_p^*$ and follows the protocol by encrypting a randomly sampled value to receive $[\![M_k^l]\!]$, simulating the response from the level-site. The next level-site is then passed the encrypted client feature vector and the node

[2] In our practical implementation, we use integers or indexes instead of bit strings that denote a path down the tree. However, this is functionally equivalent.

index with which to compare. This process repeats until the simulator comes to a classification.

This simulator simply performs the same steps as the client, and randomly draws values to simulate the secret parameters of the level-site. Given that the comparison protocol and the additive homomorphic encryption scheme are statistically indistinguishable from the random output [28], [29], [20], [25], nothing about the internal state of the level-site is revealed through these operations.

B. Security against an HBC level-site l

We assume the existence of HBC level-sites that wish to learn the client's private input vector x. We construct a simulator that produces output computationally indistinguishable from a legitimate run of the system.

The simulation begins when the simulator sets $x_i = 0, \forall x_i \in x$, generates a key pair (pk, sk) and encrypts the x_i's. The simulator then runs the protocol from the level-site by generating μ_l, σ_l, k, s, etc., and generates $[\![M_k^l]\!]$. Then it generates j by randomly selecting bits $(b_0', \cdots, b_{l-1}')_2$. The simulator then goes through the comparison portion of the protocol. The level-site 1 passes on $[\![b_0, \cdots, b_l]\!]$ and the encrypted feature vector to level-site $l + 1$.

Simply, the simulator runs the level-site's protocol and encrypts the value 0 for $x_i, \forall x_i \in$ feature vector x. It then steps through the protocol and, due to the encrypted integer comparison protocol, does not learn any information about the value of x_i. This makes it computationally indistinguishable from a real run of the system. Level-site l uses the previous results of the tree-traversal for the previous level-sites, $0, \cdots, l - 1$ the required x_i's for the current level l the thresholds for the current level l. Only the client knows the feature vector x and its classification. Thus, the level-site does not learn any relevant information from the client.

C. Client Timing Attack and Performance Tradeoffs

Over numerous runs, the client can begin to estimate the thresholds stored in different nodes by observing different feature vectors and classification results. An easy remediation is to throttle the client, as is done by other PPDT models [18]. However, the client can determine the level on which the class is on using the amount of time that a classification takes. This

attack can be mitigated by utilizing a proxy between the client and level-sites that adds random waiting times to the query result. The minimum extra time needed would be between 0.5 and 1 seconds to make it indistinguishable between two adjacent level-sites, as shown in Table II. Throttling would not severely affect the average run-time compared to the original Joye and Salehi PPDT protocol [13].

D. Security against Side Channel Attacks

Side channel attacks are possible on multiple cryptosystems such as RSA [3]. We use Joye and Salehi's [13] comparison protocol for comparing numerical attributes, which is timing attack resistant on each level-site. For categorical data comparison, we used a modified *Protocol 1 EQT-1* [20] that always runs the same computations, which also makes it resistant to timing attacks. As both comparison protocols also have a similar run-time, the overall system has timing attack protection. Thus, the computation time between all the level-sites will be indistinguishable. If a classification finishes early, we can send a bogus value downstream to ensure that all level-sites complete a computation.

We use containerization to ensure stronger mitigations against power-based side-channel attacks. Containers are immutable after being created [1], making it difficult to install software to analyze the runtime environment. Since containers are usually the end result of an automated deployment pipeline, consistent replacement of keys at level-sites [14] would also be a mitigation. More, a data owner could split the level-sites to be managed by multiple non-colluding cloud providers, which would make obtaining power readings to derive the private key much more difficult.

VI. Experimental Results

A. Experiment Setup

We implemented and tested the proposed approach using Amazon Elastic Kubernetes Service (EKS). We used t2.medium EC2 instances, which have 2 vCPUs and 4 GB of RAM. Our open source code implementation[3] used the homomorphic encryption library implemented in Java [24] that implements Joye and Salehi's encrypted integer protocol [13] and Veugen's encrypted equality comparison [20].

We used Docker containers and Kubernetes as it allows the creation of replicas [10] and supports horizontal scaling [23], which fits our availability requirements in Section IV-A. We considered two EC2 VMs in the same region but in a different availability zone to simulate using two non-colluding cloud providers.

B. Performance Results

For our experiments, we ran our PPDT with each container storing a level-site. When obtaining our results, we did not implement throttling based on the discussion in Section V-C, as we wanted to have a consistent view on how much time was needed for the PPDT evaluation at each level-site. We ran the

[3]https://github.com/adwise-fiu/Level-Site-PPDT

evaluation 10 times each and provided the average execution time. The client experienced on average a 25-millisecond transmission delay for each connection, which is considered in the results. We also report the estimated time that Joye and Salehi PPDT would take by computing an evaluation that would traverse the d levels of the PPDT.

Our solution agrees 100% with a standard DT classification. Setting up the level-sites was completed in less than a second for all datasets. The performance results of our approach compared to Joye and Salehi are reported in Table II. Our superior performance is due to not requiring a complete binary tree [13], reducing the evaluation depth and the number of encrypted comparisons.

TABLE II
Comparing classification time of our approach with Joye and Salehi Approach under hypothyroid and nursery datasets.

Levels	Hypothyroid level-sites	Hypothyroid Joye & Salehi	Nursery level-sites	Nursery Joye & Salehi
2	**0.807 s**	3.279 s	**0.656 s**	4.140 s
4	**1.772 s**	3.279 s	**2.041 s**	4.140 s
9	4.148 s	3.279 s	**2.950 s**	4.140 s
12	N/A	N/A	5.648 s	4.140 s

Based on Table I, the average depth for the Hypothyroid set is 2.78 which corresponds to a value between 0.807 and 1.772 secs in Table II. This is significantly faster than the Joye and Salehi approach. Similarly, the average depth for Nursery data is 5.99 which corresponds to a value between 2.041 and 2.950 seconds, again much less than 4.140 of Joye and Salehi.

C. Comparison with other Related Work

We compared our approach with Li et al. [18] and when repeating the same client queries it takes our PPDT 2.041, 6.198 and 15.878 seconds for the nursery, breast cancer and spambase dataset, respectively, which is at least 33% faster. Compared to Yuan et al. [32] using the *spambase* dataset, which they reported for evaluations. As shown in Table III, our method is faster at almost all levels due to Yuan et al.'s need to encrypt data for each evaluation and the computational overhead of traversing the PPD-GBDT.

TABLE III
Comparing the classification time of [32] with our approach using Spambase dataset.

Levels	Our Approach	Yuan et al.[32]
2	1.927 s	0.49 s
3	2.736 s	7.58 s
4	3.937 s	8.35 s
5	4.638 s	17.33 s
6	5.783 s	19.37 s

VII. Conclusion

In this paper, we present a new secure decision tree inference protocol by splitting the DT model into level-site constructs. Our PPDT exhibits optimal performance in sparse trees [12][15] due to fewer encrypted integer comparisons, resulting in faster performance compared to other PPDTs [18][32][13]. In addition, it offers resistance to timing attacks through timing attack resistant comparison protocols [13][20].

Our PPDT also offers strong mitigations against power-based side channel attacks [3], as we use containers that offer power-based side channel protections [1], key replacement [14], and our approach can be used to split our PPDT with two non-colluding cloud providers. Finally, our PPDT protocols allow us to support load balancing to more evenly distribute more computing resources to higher levels of the PPDT, which would experience more usage and redundancy [10] in case a level-site experiences an outage.

Acknowledgements

References

[1] NIST Special Publication 800-190. Application Container Security Guide. https://csrc.nist.gov/pubs/sp/800/190/final, September 2017. [Online; accessed 29-December-2023].

[2] Furkan Aydin, Emre Karabulut, Seetal Potluri, Erdem Alkim, and Aydin Aysu. Reveal: Single-trace side-channel leakage of the seal homomorphic encryption library. In *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1527–1532. IEEE, 2022.

[3] Alessandro Barenghi, Diego Carrera, Silvia Mella, Andrea Pace, Gerardo Pelosi, and Ruggero Susella. Profiled side channel attacks against the rsa cryptosystem using neural networks. *Journal of Information Security and Applications*, 66:103122, 2022.

[4] Raphael Bost, Raluca Popa, Stephen Tu, and Shafi Goldwasser. Machine learning classification over encrypted data. In *Network and Distributed System Security Symposium (NDSS 2015)*. The Internet Society, 01 2015.

[5] Christa Cody, Vitaly Ford, and Ambareen Siraj. Decision tree learning for fraud detection in consumer energy consumption. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 1175–1179, 2015.

[6] Ivan Damgrd, Martin Geisler, and Mikkel Krigaard. Efficient and secure comparison for on-line auctions. In *Australasian conference on information security and privacy*, pages 416–430. Springer, 2007.

[7] Ivan Damgrd, Martin Geisler, and Mikkel Krigaard. Homomorphic encryption and secure comparison. *International Journal of Applied Cryptography*, 1(1):22–31, 2008.

[8] Ivan Damgård, Martin Geisler, and Mikkel Kroigard. A correction to 'efficient and secure comparison for on-line auctions'. *International Journal of Applied Cryptography*, 1(4):323–324, 2009.

[9] Fatih Emekçi, Ozgur Sahin, Divyakant Agrawal, and Amr Abbadi. Privacy preserving decision tree learning over multiple parties. *Data and Knowledge Engineering*, 63:348–361, 11 2007.

[10] Michael Hausenblas. Kubernetes Replicas: Underappreciated Workhorses. https://www.redhat.com/en/blog/kubernetes-replicas-appreciated-workhorses, July 25, 2017. [Online; accessed 29-December-2023].

[11] Health and Human Services. Health insurance portability and accountability act. https://www.hhs.gov/hipaa/for-professionals/security/laws-regulations/index.html, 1996.

[12] Xiyang Hu, Cynthia Rudin, and Margo Seltzer. Optimal sparse decision trees. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 7265–7273. Curran Associates, Inc., 2019.

[13] Marc Joye and Fariborz Salehi. Private yet efficient decision tree evaluation. In *Proceedings of DBSec 2018*, pages 243–259, 07 2018.

[14] Paul Kocher, Joshua Jaffe, Benjamin Jun, and Pankaj Rohatgi. Introduction to differential power analysis. *Journal of Cryptographic Engineering*, 1:5–27, 2011.

[15] Jimmy Lin, Chudi Zhong, Diane Hu, Cynthia Rudin, and Margo Seltzer. Generalized and scalable optimal sparse decision trees. In *International Conference on Machine Learning*, pages 6150–6160. PMLR, 2020.

[16] Yehuda Lindell and Benny Pinkas. Privacy preserving data mining. In Mihir Bellare, editor, *Advances in Cryptology - CRYPTO 2000*, pages 36–54, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[17] Yehuda Lindell and Benny Pinkas. Secure multiparty computation for privacy-preserving data mining. *Journal of Privacy and Confidentiality*, 1:197, 11 2008.

[18] Lin Liu, Rongmao Chen, Ximeng Liu, Jinshu Su, and Linbo Qiao. Towards practical privacy-preserving decision tree training and evaluation in the cloud. *IEEE Transactions on Information Forensics and Security*, 15:2914–2929, 2020.

[19] Hefeng Mo, Ting Tao, Wenbin Gao, Chunlei Fan, Ligang Wang, Xin Zhao, and Haili Wang. Application of decision tree algorithm in medical nursing and health assessment system. In *2023 3rd Asia-Pacific Conference on Communications Technology and Computer Science (ACCTCS)*, pages 55–59, 2023.

[20] Majid Nateghizad, Thijs Veugen, Zekeriya Erkin, and Reginald L Lagendijk. Secure equality testing protocols in the two-party setting. In *Proceedings of the 13th International Conference on Availability, Reliability and Security*, pages 1–10, 2018.

[21] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology—EUROCRYPT'99: International Conference on the Theory and Application of Cryptographic Techniques Prague, Czech Republic, May 2–6, 1999 Proceedings 18*, pages 223–238. Springer, 1999.

[22] European Parliament and Council of the European Union. General data protection regulation. https://gdpr.eu/, 2019.

[23] Bjørn Erik Pedersen. Horizontal Pod Autoscaling. https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/, May 5, 2018. [Online; accessed 29-December-2023].

[24] Andrew Quijano and Kemal Akkaya. Server-side fingerprint-based indoor localization using encrypted sorting. In *2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems Workshops (MASSW)*, pages 53–57, Monterrey, CA, 2019. IEEE.

[25] Rivest RL, Adleman LM, and Dertouzos ML. On data banks and privacy homomorphisms. *Foundations of Secure Computation*, 4:169–179, 01 1978.

[26] Raymond Tai, Jack Ma, Yongjun Zhao, and Sherman Chow. Privacy-preserving decision trees evaluation via linear functions. In *Computer Security - ESORICS 2017, Part II, LNCS vol. 10493*, pages 494–512. Springer, 09 2017.

[27] Stacey Truex, Ling Liu, Mehmet Gursoy, and Lei Yu. Privacy-preserving inductive learning with decision trees. In *6th International Congress on Big Data*, pages 57–64, 06 2017.

[28] Thijs Veugen. Improving the dgk comparison protocol. In *WIFS 2012 - Proceedings of the 2012 IEEE International Workshop on Information Forensics and Security*, pages 49–54, 12 2012.

[29] Thijs Veugen. Correction to "improving the dgk comparison protocol". *Cryptology ePrint Archive*, 2018.

[30] Wang Yanxia. Student information management decision system based on decision tree classification algorithm. In *2022 IEEE 5th International Conference on Information Systems and Computer Aided Education (ICISCAE)*, pages 827–831, 2022.

[31] Andrew C. Yao. Protocols for secure computations. In *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, pages 160–164. IEEE, 1982.

[32] Shuai Yuan, Hongwei Li, Xinyuan Qian, Meng Hao, Yixiao Zhai, and Xiaolong Cui. Toward efficient and end-to-end privacy-preserving distributed gradient boosting decision trees. In *ICC 2023-IEEE International Conference on Communications*, pages 1566–1571. IEEE, 2023.