Privacy-Preserving Outsourcing Decision Tree Evaluation from Homomorphic Encryption

Kexin Xu[a,b,c], Benjamin Hong Meng Tan[d], Li-Ping Wang[a,b,c,*], Khin Mi Mi Aung[d], Huaxiong Wang[e]

[a] *State Key Laboratory of Information Security, Institute of Information Engineering, CAS, Beijing, China*
[b] *State Key Laboratory of Cryptology, Beijing, China*
[c] *School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China*
[d] *Institute for Infocomm Research (I2R), Agency for Science, Technology and Research (A*STAR), Singapore*
[e] *School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore*

Abstract

A decision tree is a common algorithm in machine learning, which performs classification prediction based on a tree structure. In real-world scenarios, input attribute values may be sensitive and tree models are usually private, thus the computation potentially incurs security and privacy risks. In this paper, we focus on how to retain privacy when outsourcing decision tree evaluation. Leveraging homomorphic encryption, our construction achieves data confidentiality against semi-honest adversaries while enabling both the client (who provides the attributes) and the model holder to be offline during evaluation. Based on an unbalanced-computational-ability two-server model, we achieve single-branch evaluation, which is exponentially less computation than previous schemes that evaluated the entire decision tree. A proof-of-concept implementation demonstrates this. Additionally, with multi-key encryption and joint decryption, our protocol easily supports multi-client scenarios.

Keywords: Decision Tree, Outsourcing Computation, Cloud Computing, Homomorphic Encryption, Privacy Preserving.

1. Introduction

As the cloud services industry grows, Machine Learning as a Service (MLaaS) is becoming increasingly popular due to its versatility. Typical cases include medical diagnosis, face recognition, email classification, etc. In these scenarios, machine learning algorithms require access to sensitive raw data that, if leaked, might be used maliciously for targeted ad delivery or even extortion and threats. Therefore, it is critical to protect the data used in machine learning applications. Fully homomorphic encryption (FHE) [1, 2, 3, 4, 5] supports evaluation on ciphertexts without decryption, unlocking an option to outsource computation of sensitive information to an untrusted environment securely (e.g., the cloud). It is exactly suited for the application scenario of MLaaS.

A decision tree is a classical classification algorithm in machine learning that makes decisions based on a pre-trained tree structure. In private decision tree evaluation [6, 7, 8] from homomorphic encryption, a client sends an encrypted set of attributes to a model holder, who performs homomorphic operations according to its undisclosed tree model to provide privacy-preserving predictions and then returns the encrypted classification result to the client. Encrypting the data ensures its confidentiality, and the model holder handles only the ciphertext throughout without access to the private information.

Several interactive approaches [9, 10, 11] based on multiparty secure computation (MPC) techniques such as garbled circuits and oblivious transfer have also been proposed for above requirements. Although better computational performance might be achieved compared to solutions based only on FHE, using MPC often raises additional issues, such as network latency and high bandwidth usage. In contrast, clients in FHE-based approaches only upload and receive ciphertexts, and remain offline during the computation phase, enabling less communication and interaction.

However, sometimes both the model holder and the client need to be offline without compromising either party's privacy. For instance, a model holder has insuf-

*Corresponding author. E-mail address: wangliping@iie.ac.cn

ficient computation power or is unwilling to be responsive at all times. To this end, Lu et al. mentioned in their work [6] to delegate the computation to a third-party cloud server, while preserving the privacy of the client and the model. But only a preliminary idea was given, and several concerns exist when trying to instantiate it. The first one is that when performing homomorphic comparisons, the cloud server needs to know the attribute index corresponding to each decision node, which should be private to the model holder. Still, the scheme does not consider how to protect such information. Secondly, the model information is supposed to be encrypted under the client's public key, making it necessary for the model holder to re-encrypt and send it to the cloud server when a new client accesses the model.

1.1. Our Contribution

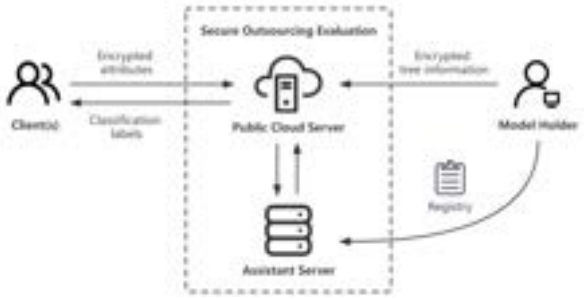

Figure 1: Our privacy-preserving outsourcing decision tree evaluation protocol.

To address these issues, in this paper we propose a privacy-preserving outsourcing decision tree evaluation (ODTE) protocol based on a two-server model. Such a model is a recent growing trend that can be seen in different security designs [12, 13, 14] tailored for specific applications. As shown in Fig. 1, a client and a model holder provide their respective encrypted data, and the decision tree evaluation is conducted between a pair of unbalanced-computational-ability servers hosted by independent cloud providers. A public cloud server is responsible for external interactions and primary computational tasks. And an assistant server participates in the process by verifying clients and facilitating cryptographic operations. Specifically, it assists the cloud server in path selection, avoiding traversing all nodes in the computation.

As a result, our protocol performs homomorphic comparison and node selection for a single branch only, reducing the computational complexity from $O(2^d)$ in previous works to $O(d)$, where d is the depth of the decision tree. Leveraging multi-key encryption and distributed decryption techniques, our construction also easily supports multi-client scenarios, that is, where the input private attributes are provided by multiple mutually untrusted parties. In addition, a registry shared with the assistant server avoids service abuse from clients who are not authorized by the model holder, while preventing the cloud server from impersonating a client. Under the assumption that there is no collusion between the public cloud server and any party, our privacy-preserving ODTE protocol is secure against semi-honest adversaries. Finally, a proof-of-concept implementation is done to demonstrate the practical performance of outsourcing computation under tree models with different parameters.

1.2. Roadmap

The rest of the paper is organized as follows. Section 2 describes several relevant works. Preliminaries are introduced in Section 3. Then, we give a novel privacy-preserving ODTE protocol as well as its security analysis in Section 4. Subsequently, the evaluation and comparison of the performance are provided in Section 5. Finally, Section 6 concludes our work.

2. Related Work

The problem of privacy-preserving for decision tree evaluation has been studied over the past years. One approach to address this issue is based on secure multi-party computation, using techniques such as secret sharing, garbled circuits or oblivious transfer to perform the protocol interactively. Another approach dedicated to improving communication overhead is based on homomorphic encryption, where the computation is done non-interactively and independently by the server.

2.1. Interactive Protocols

Bost et al. [9] proposed a scheme based on FHE (BGV [2]). At each decision node, the client and server perform an interactive comparison protocol to obtain the comparison result. Finally, the decision tree is represented as a polynomial where the variables are comparison results and apply SIMD for parallel computation.

To avoid using heavy FHE, Wu et al. [10] propose a protocol with security against semi-honest adversaries based on additive homomorphic encryption (the exponential variant of the ElGamal [15]) and oblivious transfer. Then, they give an extension that is robust against malicious adversaries using a new conditional OT protocol.

Using linear functions instead of higher polynomials to represent decision trees, Tai et al. [11] reduce the complexity of the protocol from exponential in tree depth to linear with the decision node number, leading to better performance for sparse decision trees with large depth.

2.2. Non-Interactive Protocols

Lu et al. [6] proposed a non-interactive and output expressive private comparison protocol using ring variants of FHE, called XCMP. This protocol requires only one homomorphic multiplication and has performance advantages for relatively small inputs (less than 16 bits). By instantiating XCMP, they provide a non-interactive variant of [11], but the communication complexity is still $O(2^d)$ (d is the depth of the tree).

Tueno et al. [7] proposed two new non-interactive decision tree evaluation protocols based on BGV and TFHE [16, 17], respectively, that significantly improve the computational overhead and reduce the communication overhead to $O(1)$. In particular, the version based on BGV and SIMD outperforms all previous work in amortized scenarios, while the TFHE-based one has less computational complexity and communication overhead, but does not support SIMD.

By approximating the step-function using a low-degree polynomial, Akavia et al. [18] use approximate homomorphic encryption (CKKS [5]) for decision tree training and prediction. The scheme achieves non-interactive prediction, lightweight client and communication in training, and rigorous privacy guarantee.

Paul et al. [19] introduced a non-interactive multi-bit comparison method based on FHE (TFHE) and a programmable bootstrapping technique. This method scales linearly with the input bit length. As an application, they developed an efficient protocol for the private evaluation of decision trees.

Recently, Cong et al. [8] presented an efficient homomorphic comparison algorithm between a plaintext and a ciphertext, as well as a decision tree homomorphic traversal technique, enabling the computational overhead of the server to be three orders of magnitude lower than existing works. In addition by using a transciphering method, they provide a version with less communication.

Azogagh et al. [20] provided a single-round protocol for computing a single branch of a decision tree using functional bootstrapping, where two newly proposed primitives are used: blind node selection and blind array access. This scheme requires only d comparisons, but its computation still involves traversing all the nodes, thus the computation overhead is still $O(2^d)$.

2.3. Outsourcing Protocols

None of the above schemes except [6] supports outsourcing computation. While in [6], although it is said that the model holder is allowed to delegate the computation to a third-party evaluator, there are still some issues in instantiation. Therefore, to the best of our knowledge, there is no privacy-preserving ODTE protocol based on homomorphic encryption now.

We observe that a recent work by Zheng et al. [14] utilizes secret sharing to perform ODTE. A two-server model is also used in their construction, but is balanced. Secret-shared multiplications are based on Beaver's triples, which usually requires the involvement of a trusted third party. Moreover, all the nodes are to be traversed and the comparison at each node takes multiple rounds of interaction between two servers. Therefore, it is vulnerable to network effects causing heavy latency.

3. Preliminaries

3.1. Notations

Throughout this paper, we denote vectors and matrices respectively in lower-case bold (e.g. $\mathbf{a}$) and upper-case bold (e.g. $\mathbf{A}$). All vectors are column vectors by default and $\mathbf{a}^\intercal$ denotes the transpose of $\mathbf{a}$. For a power-of-2 integer N, let $\mathcal{R}$ denote the cyclotomic ring $\mathbb{Z}[X]/(X^N + 1)$ and $\mathcal{R}_q := \mathcal{R}/q\mathcal{R}$, whose canonical representatives are polynomials of degree less than N with integer coefficients modulo q. For $a \in \mathcal{R}$, $a[i]$ denotes the ith coefficient of polynomial a. Given a set S, $a \leftarrow S$ signifies that a is an element of S sampled uniformly at random. For a probability distribution χ, $a \leftarrow \chi$ indicates that a is sampled from χ. I_n denotes the unit matrix of dimension n. Let λ denote the security parameter by default. The capital O is dedicated to the asymptotic notation which provides a worst-case asymptotic upper bound.

3.2. Decision Tree Evaluation

A decision tree is a common machine learning algorithm that uses a model of a tree to make decisions. Specifically, it implements a function $\mathcal{T} : (a_1, \ldots, a_w) \rightarrow \{v_1, \ldots, v_n\}$ that maps an attribute vector to a finite set of classification labels.

Suppose a (binary) decision tree with depth d consists of m decision nodes and n leaf nodes. Define the relevant variables as follows:

- $a_j (1 \leq j \leq w)$: the value of jth attribute, where the order of attributes is specified in advance.

- $t_i (1 \le i \le m)$: the threshold value at ith decision node.
- $h_i (1 \le i \le m)$: the index of the attribute that the ith decision node corresponds to.
- $v_u (1 \le u \le n)$: the label value at uth leaf node.

An example of decision tree evaluation is shown in Fig. 2. Given an attribute vector $(a_1, \ldots, a_w)$, compare the threshold value t_i and attribute value a_{h_i} corresponding to the current node from the root node, and move to the left node if $a_{h_i} > t_i$, otherwise move to the right node. Repeat the threshold comparison and node selection operations until reaching a leaf node. Output the label value of this leaf node as the classification result.

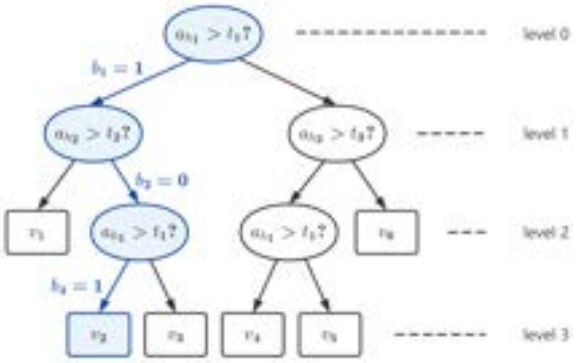

Figure 2: A decision tree evaluation example with $d = 3, m = 5, n = 6$.

In the scenario of outsourcing decision tree evaluation (ODTE), the attribute vector and decision tree information (including thresholds, attribute indexes and classification labels) are provided by a client and a model holder, respectively. The computation is then delegated to a third party (e.g., a public cloud server).

3.3. Homomorphic Encryption over Rings

Now we introduce two forms of ring-based encryption commonly found in FHE: RLWE and RGSW. They each satisfy additive homomorphism, and homomorphic multiplications are achieved by combining the two using external products [1].

[1] Actually, RGSW ciphertexts are sufficient for homomorphic multiplication [21, 22], while replacing the inner product of two RGSW ciphertexts by a simpler external product enables a significant speed-up in running.

RLWE Encryption Scheme. The Ring-Learning-with-Errors (RLWE) scheme was first described in [23] for cyclotomic rings $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$. Let $\mathcal{D}$ denote a discrete Gaussian distribution over $\mathcal{R}$ for noise sampling, and let χ be a secret key distribution. Parameterized by a message modulus p and a ciphertext modulus q, the scheme consists of the following algorithms:

- $\mathsf{RLWE.SKGen}(1^\lambda)$: Given the security parameter λ, Sample and output the secret key
$$\mathsf{sk} := s \leftarrow \chi.$$
- $\mathsf{RLWE.PKGen}(\mathsf{sk})$: See $s = \mathsf{sk}$. Pick $a \leftarrow \mathcal{R}_q$ uniformly at random and sample $e \leftarrow \mathcal{D}$. Output the public key
$$\mathsf{pk} := (a, b) = (a, a \cdot s + e \mod q) \in \mathcal{R}_q^2.$$
- $\mathsf{RLWE.Enc}_{\mathsf{pk}}^p(\mu)$: To encrypt $\mu \in \mathcal{R}_p$ under the public key pk, sample $u \leftarrow \chi$ and $e_0, e_1 \leftarrow \mathcal{D}$. Return the ciphertext
$$\mathsf{ct} := (c_0, c_1) = u \cdot \mathsf{pk} + (e_0, \frac{q}{p} \cdot \mu + e_1) \mod q \in \mathcal{R}_q^2.$$
- $\mathsf{RLWE.Dec}_{\mathsf{sk}}^p(\mathsf{ct})$: For $\mathsf{ct} = (c_0, c_1)$ and $\mathsf{sk} = s$, compute and recover the message
$$\mu' = \left\lfloor \frac{p}{q} \cdot (c_0 - c_1 \cdot s) \right\rceil \mod p.$$

The sum (homomorphic addition) $\boxplus$ of two RLWE ciphertexts is defined as

$$\boxplus : \mathrm{RLWE} \times \mathrm{RLWE} \to \mathrm{RLWE}$$
$$(\mathsf{ct}, \mathsf{ct}') \mapsto \mathsf{ct} \boxplus \mathsf{ct}' = \mathsf{ct} + \mathsf{ct}' \mod q.$$

RGSW Encryption Scheme. The Ring-Gentry-Sahai-Waters (RGSW) scheme is a RLWE-based variant of GSW [4]. By embedding a set of key-switching keys in the ciphertext, the scheme can support faster homomorphic multiplication without key switching [24, 2]. Consistent with RLWE, it is parameterized by $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$, a ciphertext modulus q ($\ell = \lceil \log q \rceil$), as well as distributions $\mathcal{D}$ and χ. The scheme works as follows, where the secret key $\mathsf{sk} \leftarrow \mathsf{RLWE.SKGen}(1^\lambda)$. Since decryption and homomorphic addition of RGSW ciphertexts will not be involved in this paper, we omit them here.

- $\mathsf{RGSW.PKGen}(\mathsf{sk})$: See $s = \mathsf{sk}$. Pick $\mathbf{a} \leftarrow \mathcal{R}_q^{2\ell}$ uniformly at random and sample $\mathbf{e} \leftarrow \mathcal{D}^{2\ell}$. Output the public key
$$\mathsf{PK} := (\mathbf{a}^\intercal, \mathbf{b}^\intercal) = (\mathbf{a}^\intercal, \mathbf{a}^\intercal \cdot s + \mathbf{e}^\intercal \mod q) \in \mathcal{R}_q^{2 \times 2\ell}.$$

- $\mathsf{RGSW.Enc}_{\mathsf{PK}}(\mu)$: To encrypt $\mu \in \mathcal{R}_2$ under the public key PK, sample $u \leftarrow \chi$ and $\mathbf{E} \leftarrow \mathcal{D}^{2\times 2\ell}$. Return

$$\mathsf{CT} := u \cdot \mathsf{PK} + \mathbf{E} + \mu \cdot \mathbf{G} \in \mathcal{R}_q^{2\times 2\ell},$$

where gadget matrix $\mathbf{G} = \mathbf{I}_2 \otimes (1, 2, \cdots, 2^\ell)^\intercal \in \mathcal{R}_q^{2\times 2\ell}$.

The external product [16] (homomorphic multiplication) $\boxtimes$ of an RGSW ciphertext and an RLWE ciphertext is defined as

$$\boxtimes : \mathrm{RGSW} \times \mathrm{RLWE} \to \mathrm{RLWE}$$
$$(\mathsf{CT}, \mathsf{ct}) \mapsto \mathsf{CT} \boxtimes \mathsf{ct} = \mathsf{CT} \cdot \mathbf{G}^{-1}(\mathsf{ct}) \mod q,$$

where $\mathbf{G}^{-1} : \mathcal{R}_q \to \mathcal{R}^\ell$ is an (randomized,) efficiently computable decomposition algorithm [25], such that $\mathbf{x} \leftarrow \mathbf{G}^{-1}(\mathbf{a})$ follows the discrete Gaussian distribution with standard deviation $O(1)$, and always satisfies

$$\mathbf{G} \cdot \mathbf{x} = \mathbf{a} \mod q.$$

For simplicity, we write $(\!|\mu|\!)_{\mathsf{pk}}$ (resp., $[\![\mu]\!]_{\mathsf{PK}}$) to denote an RLWE (resp., RGSW) encryption of the message μ under pk (resp., PK) hereinafter.

3.4. XCMP Homomorphic Comparison

XCMP, proposed in [6], is a non-interactive private comparison algorithm from ring-based FHE. This protocol requires only one homomorphic multiplication and has performance advantages for relatively small inputs (less than 16 bits). The solution takes as input two ciphertexts of a and b and produces one ciphertext that indicates whether $a > b$.

For adaptation to our protocol, we instantiate it with RLWE and RGSW ciphertexts here. For $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$, encode $a, b \in \mathbb{Z}_N$ in the degree of polynomials and let p, the message modulus of RLWE, be a prime number. Both input ciphertexts should be decryptable by the same secret key sk, i.e.

$$\mathsf{pk} \leftarrow \mathsf{RLWE.PKGen}(\mathsf{sk})$$
$$\mathsf{PK} \leftarrow \mathsf{RGSW.PKGen}(\mathsf{sk}).$$

The detailed comparison procedure is presented in Algorithm 1, where function $\mathsf{Neg}(\cdot)$ can be achieved by applying homomorphically an automorphism map $\mathcal{M} : X \mapsto X^{2N-1}$ as mentioned in [6]. (We refer the reader to [6, 26] for more details about that.) Note that in use, such a step could be avoided when $[\![X^{-b}]\!]_{\mathsf{PK}}$ is given directly as an input. Finally, the algorithm outputs an RLWE ciphertext. Decrypt it and extract the constant term of the plaintext polynomial, which is exactly the desired comparison result

$$c[0] = \begin{cases} 1 & \text{if } a > b, \\ 0 & \text{else.} \end{cases}$$

Algorithm 1: XCMP Homomorphic Comparison.

Input: $(\!|X^a|\!)_{\mathsf{pk}} \leftarrow \mathsf{RLWE.Enc}^p_{\mathsf{pk}}(X^a)$, $[\![X^b]\!]_{\mathsf{PK}} \leftarrow \mathsf{RGSW.Enc}_{\mathsf{PK}}(X^b)$, where $a, b \in \mathbb{Z}_N$.

Output: $(\!|c|\!)_{\mathsf{pk}}$, where $c[0]$ is the comparison result.

1 Compute $\bar{\mu} = 2^{-1} \mod p$ and $\mu = -\bar{\mu} \mod p$.
2 Set the polynomial $t = \mu(1 + X + \cdots + X^{N-1})$.
3 Sample $r \leftarrow \mathcal{R}_p$ randomly and set $r[0] = \bar{\mu}$.
4 Negate the degree $[\![X^{-b}]\!]_{\mathsf{PK}} \leftarrow \mathsf{Neg}\left([\![X^b]\!]_{\mathsf{PK}}\right)$.
5 Compute $(\!|c|\!)_{\mathsf{pk}} = t \cdot \left((\!|X^a|\!)_{\mathsf{pk}} \boxtimes [\![X^{-b}]\!]_{\mathsf{PK}}\right) \boxplus (0, r)$.

4. Our Privacy-Preserving ODTE

In this section, we started with a description of the system framework. Then, we give a novel ODTE protocol based on RLWE and RGSW, and show how to extend it to a multi-client scenario. Finally, a detailed analysis of its security is provided.

4.1. System Model

In ODTE, a client accesses the decision tree evaluation service on the server side, where the tree model is provided by a model holder. Here we adopt a two-server model which has emerged in recent years for specific scenarios, such as sharing private data between two servers to optimize computational tasks in privacy-preserving machine learning, multi-party computation, etc [12, 13, 14]. In particular, they are two unbalanced-computational-ability servers in our construction, with one having limited computational resources. A public cloud server is responsible for interacting with the client and the model holder, taking on primary computation tasks. An additional assistant server supports decryption and re-encryption of intermediate results to eliminate redundant traversals.

We assume that there is no collusion between the public cloud server and any other parties. The model holder shares a registry with the assistant server, which collects public keys of legitimately registered clients and supports real-time updates. The registry, on the one hand, prevents the cloud server from impersonating a

client and, on the other hand, allows enabling a charging model. For example, one states that only clients who have registered with the model holder and completed payment can access the decision tree evaluation service.

Like all previous work, we do not protect the following generic parameters about the decision tree: the tree depth (upper bound) d, number of input attributes w, and upper bound on the attribute and threshold values N. These parameters are assumed to be publicly available in our system. Suppose it is a complete binary tree of depth d. If not, the model provider is allowed to fill the tree with dummy nodes as many as necessary to make it complete.

4.2. Proposed Protocol

We now present our protocol for privacy-preserving ODTE. Firstly, a client and the assistant server generate their own public-private key pairs:

$$\begin{aligned}\mathsf{sk}_{\mathsf{cl}} &\leftarrow \mathsf{RLWE.SKGen}(1^\lambda),\\ \mathsf{pk}_{\mathsf{cl}} &\leftarrow \mathsf{RLWE.PKGen}(\mathsf{sk}_{\mathsf{cl}}),\\ \mathsf{sk}_{\mathsf{as}} &\leftarrow \mathsf{RLWE.SKGen}(1^\lambda),\\ \mathsf{PK}_{\mathsf{as}} &\leftarrow \mathsf{RGSW.PKGen}(\mathsf{sk}_{\mathsf{as}}).\end{aligned}$$

For the assistant server, publishing $\mathsf{PK}_{\mathsf{as}}$ is sufficient. Just take the first column of $\mathsf{PK}_{\mathsf{as}}$ as $\mathsf{pk}_{\mathsf{as}}$ when RLWE encryption is required. Unless otherwise specified, all the privacy information below is encrypted under the assistant server's public key $\mathsf{PK}_{\mathsf{as}}$ (or $\mathsf{pk}_{\mathsf{as}}$), and we omit the subscript of $[\![\cdot]\!]$ (or $(\!|\cdot|\!)$).

The protocol consists of an input preparation phase, an outsourcing computation phase, and a result reception phase. It works in the following manner.

Input Preparation Phase. The model holder and the client provide their own encrypted inputs as required.

- **Model Holder**: Encrypt private information below of a decision tree and send them to the public cloud server:
  - For $i = 1, \ldots, m$, encode the negative [2] threshold value $-t_i \in \mathbb{Z}_N$ of node i in the degree of the polynomial and encrypt it using the RGSW scheme:
  $$[\![X^{-t_i}]\!] \leftarrow \mathsf{RGSW.Enc}_{\mathsf{PK}_{\mathsf{as}}}(X^{-t_i}).$$
  - For $i = 1, \ldots, m$, represent the attribute index $h_i \in \{1, \cdots, w\}$ as a w-dimensional binary vector, where the position h_i is set to 1 and the others are 0. Then, encrypt each position separately using RGSW, and denote w ciphertexts together as H_i, where for $1 \le j \le w$,
  $$H_i[j] = \begin{cases}[\![1]\!] \leftarrow \mathsf{RGSW.Enc}_{\mathsf{PK}_{\mathsf{as}}}(1) & \text{for } j = h_i,\\ [\![0]\!] \leftarrow \mathsf{RGSW.Enc}_{\mathsf{PK}_{\mathsf{as}}}(0) & \text{for } j \ne h_i.\end{cases}$$
  - For $u = 1, \ldots, n$, encrypt the label $v_u \in \mathbb{Z}_{p'}$ corresponding to the uth leaf node using RLWE:
  $$(\!|v_u|\!) \leftarrow \mathsf{RLWE.Enc}^{p'}_{\mathsf{pk}_{\mathsf{as}}}(v_u).$$
- **Client**: For $j = 1, \ldots, w$, encode each attribute value $a_j \in \mathbb{Z}_N$ in the degree of the polynomial and encrypt it using RLWE:
  $$(\!|X^{a_j}|\!) \leftarrow \mathsf{RLWE.Enc}^{p}_{\mathsf{pk}_{\mathsf{as}}}(X^{a_j}).$$
  Send them to the public cloud server.

Outsourcing Computation Phase. With the above inputs, two servers cooperate to complete the decision tree evaluation. The computation starts from the root node.

- **Cloud Server**: Perform a homomorphic comparison of threshold and attribute values at the current node, and send the output ciphertext to the assistant server:
  - First compute the accumulated sum of all external products of $(\!|X^{a_j}|\!)$ and $H_i[j]$ ($j = 1, \cdots, w$) to blindly extract the required attribute value for decision node i:
  $$(\!|X^{a_{h_i}}|\!) \leftarrow \boxplus_{j=1}^{w} \left(H_i[j] \boxdot (\!|X^{a_j}|\!)\right)$$
  - Run XCMP algorithm:
  $$(\!|b_i|\!) \leftarrow \mathsf{XCMP}\left([\![X^{-t_i}]\!], (\!|X^{a_{h_i}}|\!)\right)$$
  - Generate a random polynomial $r_i \in \mathcal{R}_p$ and its trivial encryption $(0, \frac{q}{p} r_i)$. Add it to $(\!|b_i|\!)$ for masking:
  $$(\!|b_i + r_i|\!) \leftarrow (\!|b_i|\!) \boxplus (\!|r_i|\!).$$
- **Assistant Server**: Assist in decryption:
  $$b_i + r_i \leftarrow \mathsf{RLWE.Dec}^{p}_{\mathsf{sk}_{\mathsf{as}}}((\!|b_i + r_i|\!)).$$
  Return the constant term $b_i[0] + r_i[0]$ to the cloud server.

[2] Here we take the threshold value negative in advance to save the step of $\mathsf{Neg}(\cdot)$ in XCMP algorithm.

- **Cloud Server**: Remove the mask to recover the decision $b_i[0]$ for node selection, and move either to the left (if $b_i[0] = 1$) or the right (if $b_i[0] = 0$) leaf node.
- **Cloud Server & Assistant Server**: Repeat the above three steps until the cloud server reaches some leaf node $u' \in \{1, \cdots, n\}$.
- **Cloud Server**: Generate an encryption of a randomly chosen integer $r'_i \in \mathbb{Z}_{p'}$ trivially, and add it to $(\!|v_{u'}|\!)$:

$$(\!|v_{u'} + r'_i|\!) \leftarrow (\!|v_{u'}|\!) \boxplus (0, \frac{q}{p'} r'_i).$$

  Send the public key of the client $\mathsf{pk}_{\mathsf{cl}}$ as well as above ciphertext that hides the classification result of the decision tree to the assistant server for re-encryption.
- **Assistant Server**: Check whether $\mathsf{pk}_{\mathsf{cl}}$ is in the registry. If not, abort the process. Else, update the ciphertext to one that can be decrypted by the client and send it to the cloud server:

$$(\!|v_{u'}+r'|\!)_{\mathsf{pk}_{\mathsf{cl}}} \leftarrow \mathsf{RLWE.Enc}^{p'}_{\mathsf{pk}_{\mathsf{cl}}}(\mathsf{RLWE.Dec}^{p'}_{\mathsf{sk}_{\mathsf{as}}}((\!|v_{u'}+r'|\!))).$$

- **Cloud Server**: Homomorphically remove r' and output the final encrypted label to the client:

$$(\!|v_{u'}|\!)_{\mathsf{pk}_{\mathsf{cl}}} \leftarrow (\!|v_{u'} + r'|\!)_{\mathsf{pk}_{\mathsf{cl}}} \boxplus (0, -\frac{q}{p'} r'_i).$$

Result Reception Phase. The client receives the result which is decryptable to him only.

- **Client**: Decrypt and recover the classification result:

$$v_{u'} \leftarrow \mathsf{RLWE.Dec}^{p'}_{\mathsf{sk}_{\mathsf{cl}}}((\!|v_{u'}|\!)_{\mathsf{pk}_{\mathsf{cl}}}).$$

The above construction implements ODTE efficiently while protecting the data privacy of the client and the model holder. Despite the introduction of an additional assistant server and the need for server-to-server interaction, more impactfully, our solution has the following advantages: (i) model holder offline, (ii) single branch evaluation, and (iii) multi-client support.

Throughout the protocol, both the client and the model holder do not stay online except for sending encrypted private data at the beginning and receiving the classification result at the end, thus enabling outsourcing computation. And the encrypted model could be reused by different clients, further easing the burden of the model holder.

Meanwhile, the involvement of an assistant server enables the evaluation to be performed only on a single branch instead of traversing the whole tree, which significantly reduces the computational overhead. We provide a detailed analysis of performance in Section 5.

4.3. Multi-Client ODTE

Since the attribute values are encrypted under the same public key of the assistant server, our proposal naturally supports expansion into the case with multi-client inputs. That is to say, the input attribute values are not all provided by a single client, but provided separately by multiple clients that are not trusted by each other. To this end, we apply multi-key RLWE encryption [27] and threshold decryption based on noise flooding (a.k.a. smudging) technique [28]. Outsourcing computation is almost unaffected.

Let k be the number of parties and $\mathcal{D}_{sm}$ be a smudging noise distribution. See [28] for parameter choice and security proof. Following is the definition of relevant algorithms.

- $\mathsf{RLWE.MkEnc}^{p}_{\mathsf{pk}_1,\ldots,\mathsf{pk}_k}(\mu)$: To encrypt μ under the public keys of k parties, see $\mathsf{pk}_i = (a_i, b_i)$ for $i = 1, \ldots, k$. Compute and output

$$\mathsf{ct} := (a_1 \cdot u_1 + e_1, \ldots, a_k \cdot u_k + e_k, \sum_{i=1}^{k} b_i \cdot u_i + e) \mod q,$$

  where $u_i \leftarrow \chi$ and $e_i, e \leftarrow \mathcal{D}$.
- $\mathsf{RLWE.PartDec}_{\mathsf{sk}_i}(\mathsf{ct})$: Parse the multi-key ciphertext as $\mathsf{ct} = (a'_1, \ldots, a'_k, b)$. Sample $e'_i \leftarrow \mathcal{D}_{sm}$. For $\mathsf{sk}_i = s_i$, return

$$c_i = a'_i \cdot s_i + e'_i \mod q.$$

- $\mathsf{RLWE.FinDec}^{p}(c_1, \ldots, c_k, b)$: To recover the final result from partial decryptions, compute

$$\mu' = \left\lfloor \frac{p}{q} \cdot (b - \sum_{i=1}^{k} c_i) \right\rceil \mod p.$$

Now to implement ODTE in a multi-client scenario, replace the re-encryption algorithm performed by the assistant server (at the end of the outsourcing computation phase) with $\mathsf{RLWE.MkEnc}^{p}_{\mathsf{pk}_1,\ldots,\mathsf{pk}_k}(\cdot)$. In result reception phase, each client first performs $\mathsf{RLWE.PartDec}_{\mathsf{sk}_i}(\cdot)$ to get a partial decryption, and then the classification result could be recovered by an arbitrary party using the final decryption algorithm $\mathsf{RLWE.FinDec}^{p}(\cdot)$.

4.4. Threat Model and Security

We consider a semi-honest (honest-but-curious) adversary model. All the participants including servers are considered to follow the protocol specification honestly but wish to gain as much information as possible. Or we say that these parties may be corrupted by a semi-honest adversary. Such a model is widely used in existing works [11, 7, 20, 8] on privacy-preserving decision tree evaluation, and related works that prove security under malicious adversaries relies on additional cryptographic primitives, such as proofs of knowledge and conditional oblivious transfer in [10].

Our privacy-preserving ODTE protocol is secure under semi-honest adversaries as well as the assumption that there is no collusion between the public cloud server and any party. In the protocol, all inputs and outputs are encrypted by a semantically secure (IND-CPA) RLWE or RGSW scheme. The entire computation process, including the intermediate results, are not accessible to the client and the model holder. Input ciphertexts are under the assistant server's public key and the two servers do not collude, thus ensuring the information privacy of the client and model holder. At each decision node, attribute selection and homomorphic comparison are performed blindly without decryption. The comparison results are sent to the assistant server after randomization so that the latter does not know the actual comparison results. One thing to emphasize is that although the cloud server is aware of the path selection at each node, he has no knowledge of the attributes and thresholds, etc., and is therefore harmless. It is different from previous schemes in which homomorphic computations are performed by the model holder. The resulting label is also randomized and then re-encrypted under the client's public key (or the joint public key of multiple clients), ensuring that the result is not accessible to anyone other than the client. Besides, by introducing a registry, our protocol further prevents the public cloud server from impersonating a client to obtain classification results and inferring the label information of the decision tree.

5. Performance

We analyze asymptotic complexity in depth d for our design and compare it with state-of-the-art works in Table 1. The 'external' column denotes the number of interaction rounds with clients, and the 'internal' denotes that between servers. The 'outsourcing' column shows whether the protocol supports outsourcing computation, where 'partial support' means that the scheme only preliminarily conceives the delegated computation but has issues in instantiation.

Table 1: Comparison of non-interactive privacy-preserving decision tree evaluation protocols, where d is the tree depth.

Scheme	Rounds		Computation	Communication	Outsourcing
	external	internal			
[6]	1	/	$O(2^d)$	$O(2^d)$	not support
[7]	1	/	$O(2^d)$	$O(1)$	not support
[8]	1	/	$O(2^d)$	$O(1)$	not support
[20]	1	/	$O(2^d)$	$O(1)$	not support
[14]	1	$O(2^d)$	$O(2^d)$	$O(2^d)$	support
Ours	1	$d+1$	$O(d)$	$O(2^d)$	support

Our protocol requires a single round of external interaction (between the cloud server and the client, model holder) and $d+1$ round of internal interaction (between the cloud server and the assistant server). We still consider this a non-interactive one since the focus is on whether the client and the model holder must stay online during the computation.

In computation, the cloud server only accesses one decision node at each level, where $w+1$ external products are performed. That is, the computation of the cloud server is about $(w+1)d$ homomorphic multiplications. In addition, the assistant server takes a total of $d+1$ decryptions and one encryption. Overall, the computation complexity of our design is $O(d)$, while that of other existing non-interactive privacy-preserving decision tree evaluation schemes is $O(2^d)$.

In communication, the client sends w RLWE ciphertexts, and the model holder sends $m(w+1)$ RGSW ciphertexts as well as n RLWE ciphertexts. The final output is one RLWE ciphertext. Namely, the communication complexity is $O(2^d)$. Despite the high communication from the model holder, the transferred model is reusable in the cloud server. Actually, it is the minimum communication for ODTE since the model holder has to send information of all nodes to the cloud server, and other schemes that achieve $O(1)$ communication complexity are not able to support outsourcing computation.

To corroborate our theoretical analysis, we developed a proof-of-concept demo, which is implemented in C++ using TFHE library [29] and supports 130 bits of security. We simulated the decision tree model with different parameters, running on a laptop equipped with an

Table 2: Computation performance of different schemes, where w, d, and m denote the number of attributes, the depth of a decision tree, and the number of decision nodes, respectively.

Tree Model			Schemes				
w	d	m	[6]	[7]	[8]	[14]	Ours
13	3	5	590 ms	940 ms	10.5 ms	0.13 ms	6.5 ms
16	10	500	56370 ms	22390 ms	1045 ms	-	20.3 ms
13	13	92	10270 ms	6300 ms	190 ms	127 ms	22.7 ms
57	16	58	6880 ms	3660 ms	115 ms	6651 ms	86.7 ms

Intel Core(TM) i7-7500U CPU @ 2.70 GHz and 8 GB memory. The runtimes are given in Table 2, and the data provided in existing works are listed for reference. ([20] does not provide actual results, hence not in the table) Our implementation adopts 10-bit inputs, constrained by TFHE library. Actually, our protocol has no such limitation itself. XCMP is efficient for inputs up to 16 bits, and one can replace it with a multi-bit version [30] to support arbitrary inputs. Such a comparison may not be perfectly fair (the same problem exists in other works) since different processors and threads are used and input data size varies. However, it is already apparent that the computation cost of our proposal depends only on w and d linearly (also seen in Fig. 3), while the others in addition strongly depend on m, which is $O(2^d)$. All in all, our scheme is quite suitable for decision trees with large depths and especially numerous decision nodes.

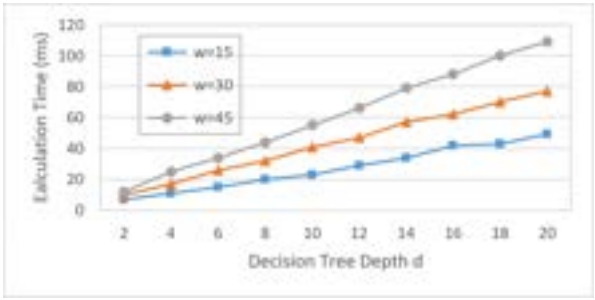

Figure 3: Outsourcing evaluation time of our ODTE with different d and w.

6. Conclusion

In this paper, we present a new privacy-preserving protocol for outsourcing decision tree evaluation. In our protocol, the private data of a client (or multiple clients) and a model holder are encrypted by RLWE or RGSW, and two servers with unbalanced computational capabilities perform the computation, including homomorphic comparison and node selection. Our design supports both the client(s) and the model holder offline but requires $d+1$ internal interactions between two servers, where d is the depth of the decision tree. Importantly, the computational complexity is improved from $O(2^d)$ in the previous scheme to $O(d)$, which is also shown in our experimental results.

Acknowledgments

Khin Mi Mi Aung, Benjamin Hong Meng Tan and Huaxiong Wang are supported by A*STAR, Singapore under research grant SERC A19E3b0099. Huaxiong Wang is also supported by Singapore Ministry of Education under Research Grant MOE2019-T2-2-083.

References

[1] C. Gentry, Fully homomorphic encryption using ideal lattices, in: M. Mitzenmacher (Ed.), Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009, ACM, 2009, pp. 169–178. doi:10.1145/1536414.1536440.
URL https://doi.org/10.1145/1536414.1536440

[2] Z. Brakerski, C. Gentry, V. Vaikuntanathan, (leveled) fully homomorphic encryption without bootstrapping, in: S. Goldwasser (Ed.), Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012, ACM, 2012, pp. 309–325. doi:10.1145/2090236.2090262.
URL https://doi.org/10.1145/2090236.2090262

[3] J. Fan, F. Vercauteren, Somewhat practical fully homomorphic encryption, IACR Cryptol. ePrint Arch. (2012) 144.
URL http://eprint.iacr.org/2012/144

[4] C. Gentry, A. Sahai, B. Waters, Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based, in: R. Canetti, J. A. Garay (Eds.), Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I, Vol. 8042 of Lecture Notes in Computer Science, Springer, 2013, pp. 75–92. doi:10.1007/978-3-642-40041-4_5.
URL https://doi.org/10.1007/978-3-642-40041-4_5

[5] J. H. Cheon, A. Kim, M. Kim, Y. S. Song, Homomorphic encryption for arithmetic of approximate numbers, in: T. Takagi, T. Peyrin (Eds.), Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I, Vol. 10624 of Lecture Notes in Computer Science, Springer, 2017, pp. 409–437. doi:10.1007/978-3-319-70694-8_15.
URL https://doi.org/10.1007/978-3-319-70694-8_15

[6] W. Lu, J. Zhou, J. Sakuma, Non-interactive and output expressive private comparison from homomorphic encryption, in: J. Kim, G. Ahn, S. Kim, Y. Kim, J. López, T. Kim (Eds.),

Proceedings of the 2018 on Asia Conference on Computer and Communications Security, AsiaCCS 2018, Incheon, Republic of Korea, June 04-08, 2018, ACM, 2018, pp. 67–74. doi:10.1145/3196494.3196503.
URL https://doi.org/10.1145/3196494.3196503

[7] A. Tueno, Y. Boev, F. Kerschbaum, Non-interactive private decision tree evaluation, in: A. Singhal, J. Vaidya (Eds.), Data and Applications Security and Privacy XXXIV - 34th Annual IFIP WG 11.3 Conference, DBSec 2020, Regensburg, Germany, June 25-26, 2020, Proceedings, Vol. 12122 of Lecture Notes in Computer Science, Springer, 2020, pp. 174–194. doi:10.1007/978-3-030-49669-2_10.
URL https://doi.org/10.1007/978-3-030-49669-2_10

[8] K. Cong, D. Das, J. Park, H. V. L. Pereira, Sortinghat: Efficient private decision tree evaluation via homomorphic encryption and transciphering, in: H. Yin, A. Stavrou, C. Cremers, E. Shi (Eds.), Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022, ACM, 2022, pp. 563–577. doi:10.1145/3548606.3560702.
URL https://doi.org/10.1145/3548606.3560702

[9] R. Bost, R. A. Popa, S. Tu, S. Goldwasser, Machine learning classification over encrypted data, in: 22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2015, The Internet Society, 2015.
URL https://www.ndss-symposium.org/ndss2015/machine-learning-classification-over-encrypted-data

[10] D. J. Wu, T. Feng, M. Naehrig, K. E. Lauter, Privately evaluating decision trees and random forests, Proc. Priv. Enhancing Technol. 2016 (4) (2016) 335–355. doi:10.1515/popets-2016-0043.
URL https://doi.org/10.1515/popets-2016-0043

[11] R. K. H. Tai, J. P. K. Ma, Y. Zhao, S. S. M. Chow, Privacy-preserving decision trees evaluation via linear functions, in: S. N. Foley, D. Gollmann, E. Snekkenes (Eds.), Computer Security - ESORICS 2017 - 22nd European Symposium on Research in Computer Security, Oslo, Norway, September 11-15, 2017, Proceedings, Part II, Vol. 10493 of Lecture Notes in Computer Science, Springer, 2017, pp. 494–512. doi:10.1007/978-3-319-66399-9_27.
URL https://doi.org/10.1007/978-3-319-66399-9_27

[12] G. Xu, H. Li, Y. Zhang, S. Xu, J. Ning, R. H. Deng, Privacy-preserving federated deep learning with irregular users, IEEE Trans. Dependable Secur. Comput. 19 (2) (2022) 1364–1381. doi:10.1109/TDSC.2020.3005909.
URL https://doi.org/10.1109/TDSC.2020.3005909

[13] B. Jiang, Multi-key FHE without ciphertext-expansion in two-server model, Frontiers Comput. Sci. 16 (3) (2022) 161809. doi:10.1007/s11704-021-0479-5.
URL https://doi.org/10.1007/s11704-021-0479-5

[14] Y. Zheng, H. Duan, C. Wang, R. Wang, S. Nepal, Securely and efficiently outsourcing decision tree inference, IEEE Trans. Dependable Secur. Comput. 19 (3) (2022) 1841–1855. doi:10.1109/TDSC.2020.3040012.
URL https://doi.org/10.1109/TDSC.2020.3040012

[15] R. Cramer, R. Gennaro, B. Schoenmakers, A secure and optimally efficient multi-authority election scheme, in: W. Fumy (Ed.), Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding, Vol. 1233 of Lecture Notes in Computer Science, Springer, 1997, pp. 103–118. doi:10.1007/3-540-69053-0_9.
URL https://doi.org/10.1007/3-540-69053-0_9

[16] I. Chillotti, N. Gama, M. Georgieva, M. Izabachène, Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds, in: J. H. Cheon, T. Takagi (Eds.), Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I, Vol. 10031 of Lecture Notes in Computer Science, 2016, pp. 3–33. doi:10.1007/978-3-662-53887-6_1.
URL https://doi.org/10.1007/978-3-662-53887-6_1

[17] I. Chillotti, N. Gama, M. Georgieva, M. Izabachène, TFHE: fast fully homomorphic encryption over the torus, J. Cryptol. 33 (1) (2020) 34–91. doi:10.1007/s00145-019-09319-x.
URL https://doi.org/10.1007/s00145-019-09319-x

[18] A. Akavia, M. Leibovich, Y. S. Resheff, R. Ron, M. Shahar, M. Vald, Privacy-preserving decision trees training and prediction, in: F. Hutter, K. Kersting, J. Lijffijt, I. Valera (Eds.), Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2020, Ghent, Belgium, September 14-18, 2020, Proceedings, Part I, Vol. 12457 of Lecture Notes in Computer Science, Springer, 2020, pp. 145–161. doi:10.1007/978-3-030-67658-2_9.
URL https://doi.org/10.1007/978-3-030-67658-2_9

[19] J. Paul, B. H. M. Tan, B. Veeravalli, K. M. M. Aung, Non-interactive decision trees and applications with multi-bit TFHE, Algorithms 15 (9) (2022) 333. doi:10.3390/a15090333.
URL https://doi.org/10.3390/a15090333

[20] S. Azogagh, V. Delfour, S. Gambs, M. Killijian, PROBONITE: private one-branch-only non-interactive decision tree evaluation, in: M. Brenner, A. Costache, K. Rohloff (Eds.), Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography, Los Angeles, CA, USA, 7 November 2022, ACM, 2022, pp. 23–33. doi:10.1145/3560827.3563377.
URL https://doi.org/10.1145/3560827.3563377

[21] J. Alperin-Sheriff, C. Peikert, Faster bootstrapping with polynomial error, in: J. A. Garay, R. Gennaro (Eds.), Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I, Vol. 8616 of Lecture Notes in Computer Science, Springer, 2014, pp. 297–314. doi:10.1007/978-3-662-44371-2_17.
URL https://doi.org/10.1007/978-3-662-44371-2_17

[22] L. Ducas, D. Micciancio, FHEW: bootstrapping homomorphic encryption in less than a second, in: E. Oswald, M. Fischlin (Eds.), Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I, Vol. 9056 of Lecture Notes in Computer Science, Springer, 2015, pp. 617–640. doi:10.1007/978-3-662-46800-5_24.
URL https://doi.org/10.1007/978-3-662-46800-5_24

[23] V. Lyubashevsky, C. Peikert, O. Regev, On ideal lattices and learning with errors over rings, in: H. Gilbert (Ed.), Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings, Vol. 6110 of Lecture Notes in Computer Science, Springer, 2010, pp. 1–23. doi:10.1007/978-3-642-13190-5_1.
URL https://doi.org/10.1007/978-3-642-13190-5_1

[24] Z. Brakerski, V. Vaikuntanathan, Efficient fully homomor-

phic encryption from (standard) LWE, in: R. Ostrovsky (Ed.), IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011, IEEE Computer Society, 2011, pp. 97–106. doi:10.1109/FOCS.2011.12.
URL https://doi.org/10.1109/FOCS.2011.12

[25] D. Micciancio, C. Peikert, Trapdoors for lattices: Simpler, tighter, faster, smaller, in: D. Pointcheval, T. Johansson (Eds.), Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings, Vol. 7237 of Lecture Notes in Computer Science, Springer, 2012, pp. 700–718. doi:10.1007/978-3-642-29011-4_41.
URL https://doi.org/10.1007/978-3-642-29011-4_41

[26] C. Gentry, S. Halevi, N. P. Smart, Fully homomorphic encryption with polylog overhead, in: D. Pointcheval, T. Johansson (Eds.), Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings, Vol. 7237 of Lecture Notes in Computer Science, Springer, 2012, pp. 465–482. doi:10.1007/978-3-642-29011-4_28.
URL https://doi.org/10.1007/978-3-642-29011-4_28

[27] L. Chen, Z. Zhang, X. Wang, Batched multi-hop multi-key FHE from ring-lwe with compact ciphertext extension, in: Y. Kalai, L. Reyzin (Eds.), Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part II, Vol. 10678 of Lecture Notes in Computer Science, Springer, 2017, pp. 597–627. doi:10.1007/978-3-319-70503-3_20.
URL https://doi.org/10.1007/978-3-319-70503-3_20

[28] G. Asharov, A. Jain, A. López-Alt, E. Tromer, V. Vaikuntanathan, D. Wichs, Multiparty computation with low communication, computation and interaction via threshold FHE, in: D. Pointcheval, T. Johansson (Eds.), Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings, Vol. 7237 of Lecture Notes in Computer Science, Springer, 2012, pp. 483–501. doi:10.1007/978-3-642-29011-4_29.
URL https://doi.org/10.1007/978-3-642-29011-4_29

[29] I. Chillotti, N. Gama, M. Georgieva, M. Izabachène, Fast fully homomorphic encryption library over the torus, https://github.com/tfhe/tfhe (2016).

[30] Y. Ishimaki, H. Yamana, Non-interactive and fully output expressive private comparison, in: D. Chakraborty, T. Iwata (Eds.), Progress in Cryptology - INDOCRYPT 2018 - 19th International Conference on Cryptology in India, New Delhi, India, December 9-12, 2018, Proceedings, Vol. 11356 of Lecture Notes in Computer Science, Springer, 2018, pp. 355–374. doi:10.1007/978-3-030-05378-9_19.
URL https://doi.org/10.1007/978-3-030-05378-9_19